CONTENTS

RAFT.AI

Multi-Model LLM Coordination versus Single-Model Routing on Verifiable Coding Tasks: A Contamination-Audited Benchmark

HIGHER FLOOR, LOWER CEILING: NINE COORDINATION METHODS, NINE MODELS, 49 TASKS

Guilherme Nigri, ATO agentictool.ai *Technical report. Draft, 2026-07-08, revised 2026-07-10 to add the reviewer-requested claude-fable-5 baseline, a per-model contamination audit, and a receipt-level gate audit.* *© 2026 ATO.*

Abstract

Multi-model coordination, meaning running several language models together by voting, debating, drafting and reviewing, escalating, or delegating, is widely promoted as a way to exceed any single model, and it now ships as closed, hosted products. We test that premise directly and report the result neutrally. On a set of 49 post-release LiveCodeBench problems, graded by sandboxed code execution against hidden tests and contamination-audited per model, we benchmark nine single models and nine coordination recipes across four coordination classes, with per-token cost drawn from metered API receipts. Our headline hypothesis, that a group of models solves more problems than the best single model, is not supported, and it weakened further under review. Reviewers flagged that our escalation recipes used claude-fable-5 as the strong closer without benchmarking it solo. Among contamination-clean models the best single, gemini-3-flash at 41 of 49, beats every recipe (38 of 49 best); the added fable baseline scored 43 of 49, the best of any system in the study, but is contamination-exposed and so serves as the mechanism check rather than the headline. The added baseline also exposed a mechanism. The flagship cascade is strictly dominated by always calling its own closer: across 49 tasks the cascade never solved a problem fable missed and lost five that fable solves. On four of the five the cause is the gate: the cascade escalates only on public-test failure, and code that passed the public tests while failing the hidden ones never escalated. On the fifth, escalation fired and the closer's answer still failed on a sandbox-environment mismatch. The binding constraint on asymmetric coordination is chiefly the verifier that decides when to escalate, not the strength of the closer. The floor result stands. A cheap model wrapped in a coordination recipe gains 9 to 10 problems (deepseek-chat rises from 28 to 37 or 38) and reaches roughly 90 percent of the best contamination-clean model, and 86 percent of the best model overall, with no model selection required, while a five-model shared-context war room lands about 2 problems below its own best member and, on three specific tasks, loses problems that member solved alone. Which effect dominates depends on the shape of the coordination. Asymmetric patterns such as cascades and advisors raise the floor while largely preserving the ceiling, and can be cheaper than a single strong model on long-horizon tasks (consistent with Anthropic's advisor results on SWE-bench Pro). Symmetric war rooms trade peak accuracy for consistency. We are explicit that at n = 49 the pass-rate differences are directional, not individually significant, and that our coordination costs reflect a short single-shot regime and are not the pattern's floor. Every score, cost, and token count derives from a stored per-task receipt; all metered costs are reproducible with the reader's own API keys, and the two runs that used a subscription channel are flagged as such where they appear.

1. Introduction

Two design philosophies compete in applied LLM systems. The first, coordination, holds that combining models produces capability beyond any single model. Self-consistency [Wang et al., 2022], multi-agent debate [Du et al., 2023], and Mixture-of-Agents [Wang et al., 2024] are prominent examples, and the idea now ships as a product such as Sakana's Fugu [Sakana AI, 2026], a learned orchestrator behind a single closed endpoint. The second philosophy, routing, holds the opposite: most queries should go to the cheapest sufficient model, escalating only when needed. FrugalGPT's cascades [Chen et al., 2023], quality-aware routing [Ding et al., 2024], and preference-based routing [Ong et al., 2024] are examples. The two views are rarely measured against each other on the same task distribution, with the same grader and honest cost accounting.

This report supplies that comparison, and reports it without a thumb on the scale. We treat each coordination recipe as a virtual model and benchmark it against the individual models it is built from, and against the best single model overall. We test three hypotheses.

- **H1.** A coordination recipe solves more problems than the best single model.
- **H2.** Where two models disagree on a task, the disagreement is sampling noise, so repeated sampling of one model would suffice.
- **H3.** The set of unsolved tasks is an artifact of single-sample luck rather than a genuine capability ceiling.

H1 is not supported. The exploratory probes argue against H2 and H3 (Section 5.3). The value of the study is not the falsification. It is the shape of what coordination does instead. Coordination does not raise the top of the distribution. It compresses the distribution, lifting weak outcomes and capping strong ones. That trade is real, quantifiable, and useful, and it tells you precisely when to coordinate and when to route.

One result in this revision deserves its own sentence. Independent review of the first draft caught that our escalation recipes billed claude-fable-5 as the strong closer while Table 1 carried no fable row, so the recipes were never compared against the simplest alternative: always call the closer. We ran that baseline (Section 4.4). It beat every recipe and strictly dominated the flagship cascade, and the main reason turned out to be the most useful engineering lesson in the study.

2. Related Work

Coordination and ensembling. Self-consistency [Wang et al., 2022] majority-votes sampled solutions. Multi-agent debate [Du et al., 2023] and Mixture-of-Agents [Wang et al., 2024] extend this to heterogeneous models that exchange intermediate outputs. These report gains on reasoning and open-ended benchmarks. We evaluate the same design patterns on execution-verified code, where correctness is unambiguous.

Cost-aware routing and cascades. FrugalGPT [Chen et al., 2023] formalizes the LLM cascade. Hybrid LLM [Ding et al., 2024] and RouteLLM [Ong et al., 2024] learn a router that sends each query to a small or large model by predicted difficulty. Our escalation recipes are instances of this family, placed on the same scoreboard as the coordination recipes and the single models.

In-API asymmetric coordination. Providers now ship asymmetric coordination as native features. Anthropic's advisor tool [Anthropic, 2026a] lets a cheaper executor consult a stronger advisor mid-generation, billing most tokens at the executor rate. It reports raising a Sonnet executor from about 75.5 percent to about 84 percent on SWE-bench Pro, roughly 92 percent of the flagship's score at roughly 63 percent of its cost. A related coordinator pattern has a strong planner delegate to cheap parallel workers [Anthropic, 2026b]. These are the cascade and routing family delivered in-API within one vendor. We test the same asymmetric idea across vendors and, critically, on single-shot tasks, where the economics differ (Section 5.4).

Contamination-free evaluation. LiveCodeBench [Jain et al., 2024] annotates competitive-programming problems with release dates, which enables evaluation strictly on problems published after a model's training cutoff. We adopt this to neutralize memorization.

3. Methodology

Task set. 49 Python problems from LiveCodeBench [Jain et al., 2024], selected by release date to postdate the training cutoffs of the 2025-era model pool the study began with. Problems mix stdin and stdout with functional, call-based formats. Because two models added later carry 2026-01 training cutoffs that postdate the task releases, we no longer claim the set is contamination-free for every model. Instead the harness records a per-model contamination audit (Table 1): a model is *clean* when every task's release date postdates its cutoff, *exposed* when its cutoff postdates the releases, and *unverified* when the provider publishes no usable cutoff. Headline comparisons are reported both overall and restricted to clean models.

Grader. Each candidate solution runs in a network-isolated, resource-limited sandbox and is graded against the problem's full hidden test suite. A task passes only if all tests pass. Primary decoding is single attempt at temperature 0. This replaces subjective or LLM-judge scoring with a verifiable oracle.

Models and provider selection. The nine models span four labs: OpenAI (gpt-5, o3-mini), Anthropic (claude-fable-5, claude-sonnet-4-6), Google (gemini-3-flash, gemini-2.5-flash, gemini-2.5-flash-lite), and DeepSeek (deepseek-reasoner, deepseek-chat). We selected these four because they are among the strongest labs on recent public coding and reasoning evaluations, because they include both leading closed providers and a leading open-weight provider (DeepSeek), and because together they cover the full cost spectrum a cost-aware developer actually routes across, from a frontier model near ten cents per task down to a small model at a fraction of a cent. This is the realistic candidate pool a router would draw from, not an arbitrary set. Each model was called through its native provider API on the user's own keys.

One provider excluded, and why. We attempted to add z.ai (the GLM family) as a fifth lab, but in our testing we could not reconcile its cost accounting. Our metered balance drained far faster than the per-call usage the API reported: it recorded cents of usage while dollars left the account. We contacted z.ai to try to resolve the discrepancy and received no response, so we could not get it working for the coordination runs. Because this entire study rests on trustworthy per-call receipts, and a cost we cannot verify is exactly the thing we set out not to publish, we excluded z.ai rather than report a number a reader could not reproduce. We record this as our own operational experience with our setup, not a general claim about the provider.

One baseline added in revision, and how it was run. claude-fable-5 was added after independent review of the first draft, because the escalation recipes already used it as their strong closer. Its 49 tasks ran in two legs: 14 tasks through Anthropic's metered API ($1.02 total, $0.0728 per task) and, after the metered budget was exhausted, 35 tasks through the Claude subscription CLI. Grading is channel-independent (the identical sandbox executes the returned program against the same hidden tests), so pass results are comparable across legs, and every task's receipt records which channel produced it. Costs are not comparable across legs: the subscription channel bills a flat fee rather than per call, and although the harness records a token-derived estimate on every subscription receipt (35 tasks, $0.74 total, mean $0.021 per task), on this provider output-token counts exclude server-side thinking tokens, so those estimates are best read as a lower bound; the gap to the metered leg's $0.0728 average is consistent with that undercount. The fable cost cell therefore reports the metered leg only, and the full-run floor is at least $1.76 total (about $0.036 per task blended). The subscription leg also does not accept a pinned temperature, so its sampling settings are the CLI defaults rather than the temperature 0 used everywhere else; Section 7 lists this as a limitation.

Coordination recipes. Nine recipes across four classes; five completed the full n = 49 (Table 2) and four ran as 12-task pilots (Table 2b). Blind: self-consistency [Wang et al., 2022] with k = 3. Sequential: two-stage cost cascades [Chen et al., 2023] that escalate on public-test failure, plus draft-then-review. Hierarchical: a coordinator with workers, and an in-context advisor. Shared-context war room: two-way debate [Du et al., 2023], a contrarian variant, and a full N-way mesh in the spirit of Mixture-of-Agents [Wang et al., 2024].

Design choices, with reasoning. Four choices shape every number in this report.

Metered APIs, not subscriptions. Every cost-accounted run went through each provider's metered API on our own keys. A flat-rate subscription bills a fixed fee and never emits a per-call record, so it cannot tell you what any single task cost. Metered access records the exact input and output tokens for each dispatch, which is the only way to attach an auditable cost to a result. We therefore paid metered per-token rates on all 4,297 measured dispatches (26.5 million tokens, 3,727 graded answers), and every cost figure traces to a stored receipt: model, tokens in and out, latency, and sandbox result. The trade is real money for real receipts, and that is the point. A cost number the reader cannot reproduce is a marketing claim, not a measurement. Subscription channels were used only where noted, for a few fallback legs and for peer review, and those legs' costs are flagged as not comparable.

The user's own keys. Running on the reader's own keys, rather than a hosted endpoint we control, makes the study reproducible by anyone and keeps the evaluation honest. The reader re-runs the same suite on the same models and checks our numbers directly.

Single attempt at temperature 0. The primary unit is one attempt per task at temperature 0, the setting closest to deterministic. This measures typical single-shot behavior rather than best-of-many, and it bounds cost. Repeated sampling appears only in the targeted variance study (Section 5.3).

Execution grading on post-release problems. Correctness is decided by running the code against hidden tests in a sandbox, not by a model's judgment, so a pass is unambiguous. Problems were selected to postdate the training cutoffs of the original model pool, and the per-model audit in Table 1 makes the residual exposure explicit rather than assumed away. Per-task cost is provider pricing applied to recorded tokens. These estimates can drift from a provider's console and should be reconciled there for exact billing.

Author's interest, disclosed. ATO builds the open-box router harness that produced every receipt in this study, and the paper's practical conclusion (route, per task, in the open) is the author's product category. We cannot remove that interest, so we handle it the only way that survives scrutiny: every claim ships with its receipt, the result most favorable to our product is the one we most invite readers to falsify, and ATO appears in this paper as instrument, not as an evaluated system. Only the named models and recipes are scored.

Statistics. Pass-rates carry Wilson score intervals [Wilson, 1927]. Paired model comparisons use McNemar's test [McNemar, 1947]. A four-fold resample at temperature above 0 (Section 5.3) separates capability from sampling variance.

Reproducibility. Results are pinned to a dataset-revision hash and a harness-configuration hash. Two pass-rates are comparable only when both match. Receipts are retained per task, the full task-level pass/fail matrix is printed in Appendix A, and the suite is re-runnable with the reader's keys. Every published run, the coordination war rooms and the full single-model baseline table, is browsable and downloadable at app.agentictool.ai/?replays (a free account is required to view; costs and machine identifiers are stripped from the public snapshots).

4. Results

4.1 THE HEADLINE HYPOTHESIS IS NOT SUPPORTED, BUT COORDINATION IS NOT A FAILURE

Table 1 gives the single-model baselines. Table 2 gives the recipes at full n = 49. The best contamination-clean single model, gemini-3-flash at 41 of 49, and the best single model overall, claude-fable-5 at 43 of 49 (contamination-exposed), both exceed the best recipe, a cost cascade at 38 of 49. H1 is not supported under either reading, and adding the reviewer-requested fable baseline widened the gap from 3 problems to 5. No recipe beats the best single model on accuracy.

Table 1. Single models (n = 49). Contamination: *clean* = every task released after the model's training cutoff; *exposed* = cutoff postdates the task releases; *unverified* = no usable published cutoff.

Model	Solved	Cost / task	Contamination
claude-fable-5	43/49	$0.0728 †	exposed (cutoff 2026-01)
gemini-3-flash	41/49	$0.0555	clean
gpt-5	40/49	$0.1051	clean
deepseek-reasoner	39/49	$0.0163	unverified
claude-sonnet-4-6	36/49	$0.0058 ‡	exposed (cutoff 2026-01)
gemini-2.5-flash	32/49	$0.0387	clean
o3-mini	31/49	$0.0291	clean
deepseek-chat	28/49	$0.0006	unverified
gemini-2.5-flash-lite	15/49	$0.0014	clean

† Metered leg only (14 of 49 tasks); the 35 subscription-channel tasks carry no comparable per-call cost (Section 3). The metered-leg receipts predate the harness's cutoff registry and record contamination status "unknown"; the row's exposed label comes from the model's published 2026-01 cutoff, which postdates every task release, and is confirmed on all 35 subscription-leg receipts. ‡ This run went through a subscription channel; its cost cell is an estimate from recorded tokens, not a metered receipt, and on this provider such estimates exclude server-side thinking tokens. A metered re-run is planned.

Table 2. Coordination recipes with complete n = 49 runs (harness 6169df13).

Recipe (class)	Solved	Cost / task
cascade: deepseek-chat then fable (sequential)	38/49	$0.117
refine: deepseek-chat then sonnet review (sequential)	37/49	$0.021
mesh: 5-model, 2-round war room (shared)	37/49	$0.145
cascade: sonnet then fable (sequential)	35/49	$0.046
self-consistency: deepseek-chat times three (blind)	29/49	$0.0017

Table 2b. Pilot recipes (12-task subsets, earlier harness hashes; not comparable to Table 2 and excluded from every headline claim).

Recipe (class)	n	Solved	Cost / task
advisor: sonnet consulting fable (hierarchical)	12	10/12	$0.315
debate: sonnet with gemini-3-flash (shared)	12	9/12	$0.086
contrarian: deepseek-chat vs gemini-3-flash (shared)	12	9/12	$0.060
cascade: sonnet then gemini-3-flash (sequential)	12	8/12	$0.045

A hierarchical coordinator design and two frontier-model recipe variants were additionally attempted and aborted mid-run; their fragments (n of 6 or less) are retained in the receipt store but not scored.

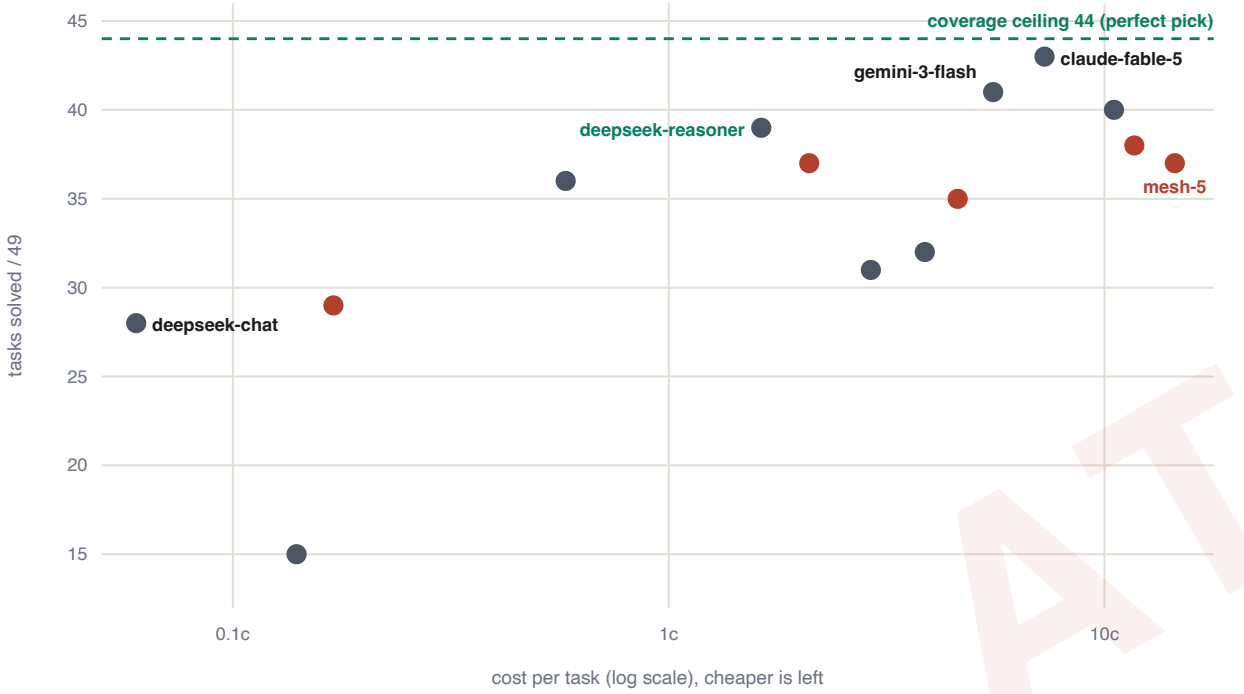

Figure 1. *Cost per task versus tasks solved, for nine single models (gray) and five coordination recipes (red), n = 49. Cheaper is left, more accurate is up. The recipes sit down and to the right of the best single models: they cost more without buying accuracy. claude-fable-5, added in revision, is the highest gray point; its cost is the metered-leg rate (Section 3). The dashed line is the coverage ceiling of 44, the accuracy a perfect per-task selector over all models would reach; no single model or recipe attains it.*

4.2 THE REAL FINDING: A HIGHER FLOOR AND A LOWER CEILING

Coordination does not move the top of the distribution. It compresses it. Two measurements point in opposite directions.

The floor rises. A cheap model wrapped in a coordination recipe gains substantially. deepseek-chat alone solves 28 of 49. Inside a cascade it solves 38 (a gain of 10), and inside draft-then-review it solves 37 (a gain of 9), a lift of about 32 percent. Framed as selection risk, a randomly chosen cheap model averages 29 of 49 (59 percent), whereas a five-model group reliably returns 37 of 49 (76 percent). That is 17 points of protection against picking the wrong model, and it reaches about 90 percent of the best contamination-clean model (37 versus 41, at $0.145 per task versus $0.0555) and 86 percent of the best model overall (37 versus 43, versus $0.0728 per task) with no model-selection skill required. The cheaper asymmetric recipes buy nearly the same floor for far less: draft-then-review reaches 37 at $0.021 per task. This mirrors Anthropic's advisor lifting a Sonnet executor by 8.5 points on SWE-bench Pro. Our lift is larger because our executor is weaker, so there is more floor to raise.

The ceiling falls. The same five-model war room, at 37 of 49, scores about 2 problems below its own best member, deepseek-reasoner at 39 of 49. At the task level, the group solved one problem that member missed alone (`3776`), but lost three the member solved alone (`3701` , `3765` , `3781`). The shared context talked a correct answer out of the group. This is the classic ensemble trade: lower variance, capped upside. We scope this to symmetric war rooms. The asymmetric recipes (cascade, refine) raise the floor without the ceiling penalty (Section 5.2). We report the task-level anchoring as observed in a single n = 49 run. A subset of the flips could be sampling noise, and confirming the exact magnitude would require repeated mesh runs.

Figure 2. The floor and ceiling trade. Top: a cheap model (deepseek-chat) gains 10 problems inside a coordination recipe, from 28 to 38. Bottom: a symmetric five-model war room scores 2 below its own best member, deepseek-reasoner, from 39 alone to 37 in the group.

The coverage ceiling. The union of problems solved by any single model, which is the accuracy a perfect per-task selector would reach, is 44 of 49, and it did not move when fable was added: fable's six misses are the five problems no model solves plus one (`3794`) that five other models already solve. The full ceiling is attained by just two models (gemini-3-flash and deepseek-reasoner before the revision; fable paired with any of five models after it). Five problems are solved by no model at all. The five-model mesh realized only 37 of the 42 its own members could collectively solve. The gap between realized (37) and attainable (44) is a selection failure, not a capability one.

4.3 STATISTICAL HONESTY

At n = 49, single attempt, the leading scores (43, 41, 40, 39, 38) fall within overlapping Wilson intervals, and no adjacent pair is individually significant by McNemar's test. claude-fable-5 at 43 of 49 carries a Wilson interval of [75.8, 94.3] percent; against gemini-3-flash the discordant tasks split 3 to 1 (exact McNemar p = 0.625), so we do not claim fable is significantly better, only that it is the best observed. Against its own cascade the split is 5 to 0 in fable's favor (exact p = 0.0625), a one-sided pattern that motivates the mechanism analysis in Section 4.4 rather than a significance claim. We therefore do not claim a significant ranking. The robust, direction-level results are the magnitude and sign of the floor and ceiling trade, and the structural findings confirmed by resampling (Section 5.3).

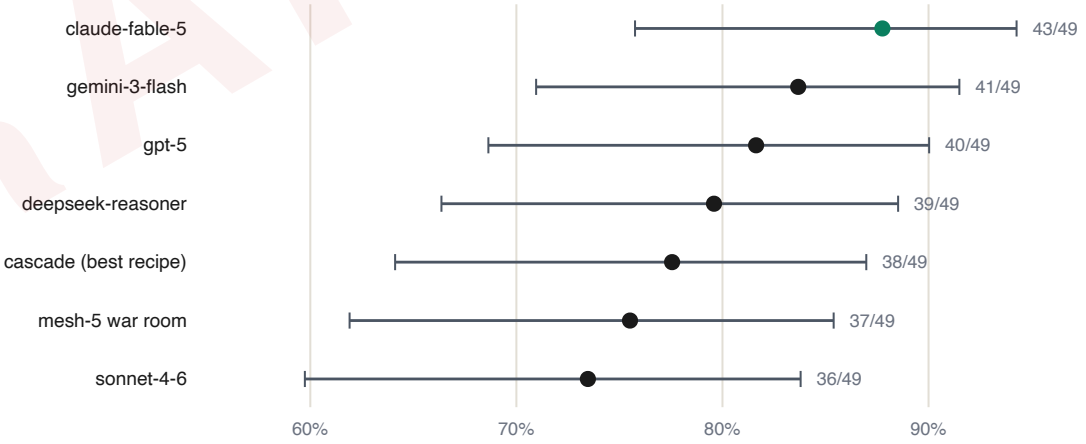

Figure 3. Pass-rate with 95% Wilson intervals for the seven leading configurations, n = 49. The intervals overlap heavily, so no adjacent pair is individually significant (claude-fable-5 versus gemini-3-flash: exact McNemar p = 0.625). The direction-level floor and ceiling trade, not the ranking, is the reproducible result.

4.4 THE REVIEWER-REQUESTED BASELINE AND WHAT IT EXPOSED

The first draft carried no claude-fable-5 row while two escalation recipes billed fable as their strong closer. Review caught it, and the fix produced the sharpest mechanism in the study. Fable solo solved 43 of 49. The flagship cascade (deepseek-chat escalating to fable on public-test failure) solved 38 while spending $0.117 per task, more than fable's $0.0728 metered rate. Task by task, the comparison is strictly one-sided: the cascade solved nothing that fable missed, and it lost five problems (`3696`, `3701`, `3764`, `3788`, `3795`) that fable solves.

Four of the five losses share one signature (the receipts record `escalated` and `primary_public_passed` per task). On `3696`, `3764`, `3788`, and `3795`, deepseek-chat produced code that passed the one-to-three public tests, so the cascade's gate declared success and never escalated, and the code then failed the hidden suite. The strong closer was available, paid for, and never called. The fifth loss, `3701`, is different and worth its own sentence: the gate worked (public tests failed, escalation fired), but the closer's answer written inside the cascade imported numpy, which the network-isolated sandbox does not provide, and died at runtime, while fable solo solved the same task without the import. So the cascade did not fail because escalation is a bad idea; it failed mostly because a weak verifier decided when to escalate, and once because escalation itself produced an environment-incompatible answer. This is direct, task-level evidence for what Section 7 had listed speculatively as the binding limitation: in asymmetric coordination, the verifier is the bottleneck far more often than the closer.

The full gate audit. The five losses are a slice of a diagnostic the receipt supports for all 49 tasks:

Gate outcome (n = 49)	Count
escalated, cascade passed	15
escalated, cascade failed	5
not escalated, cascade passed	23
not escalated, public tests passed, hidden tests failed (false trust)	6

The gate never escalated an answer that was actually correct (specificity 23 of 23) and caught 20 of the 26 tasks whose primary answer was wrong on the hidden tests (sensitivity 77 percent); the closer repaired 15 of the 20 escalations it received. The six false-trust tasks are `3696`, `3759`, `3760`, `3764`, `3788`, and `3795`. The counterfactual is also computable from receipts: an oracle gate that escalated exactly the wrong primaries, with the closer proxied by fable's solo answers on the six missed tasks (an assumption we state because in-cascade closer outputs do not exist where the gate never fired), scores 42 of 49. It rescues `3696`, `3764`, `3788`, and `3795`; tasks `3759` and `3760` are in the nobody-solves set. Against 38 observed and 43 for always calling the closer, a perfect verifier closes four of the five lost tasks, and the residual one is `3701`, the failed escalation. On this suite, verifier quality is almost the entire cascade gap, and even a perfect verifier does not quite reach always-call. A stronger gate (generated tests, cross-candidate agreement, or the user's own suite) is the highest-leverage component an escalation system can invest in, and any evaluation of cascades that reports only aggregate scores, without this task-level accounting against the closer run solo, will miss it.

5. Analysis

5.1 WHY THE DISTRIBUTION COMPRESSES

Two forces act in opposition. Error correction raises the floor. When a weak member would answer wrong, a stronger member or the vote overrides it, so catastrophic individual failures are caught. Consensus and anchoring lower the ceiling. When only the best member holds the correct answer, shared context can regress it to the majority, so the lone correct answer does not survive the group. The net is a distribution squeezed toward a reliable middle-high band.

5.2 SHAPE DECIDES THE TRADE: SYMMETRIC VERSUS ASYMMETRIC

The floor and ceiling trade is not uniform across coordination. Symmetric recipes (debate, mesh), where peers read peers, pay the full ceiling penalty. They buy consistency and lose the peak. Asymmetric recipes (cascade, advisor), where a cheap default escalates to or consults a stronger model, raise the floor while preserving access to the ceiling, because the strong model is invoked rather than averaged away. That preservation is conditional, not automatic: access to the ceiling is only as good as the gate that grants it, and Section 4.4 shows our public-test gate silently withheld the closer on four of the five tasks where it was needed (the fifth escalated and failed anyway). Asymmetric coordination is therefore the configuration that can deliver more for less, meaning near-top accuracy at a fraction of top-model cost, and it is where the practical value concentrates, provided the verifier is strong enough to know when the cheap answer is wrong.

5.3 DISAGREEMENTS ARE SPECIALIZATION, NOT NOISE (H2, H3)

We resampled a 13-task subset (the 8 tasks where gemini-3-flash and deepseek-reasoner disagreed, plus the 5 no-model-solves tasks) four times each at temperature above 0. Nine of thirteen were fully deterministic. One model reliably owns them and the other reliably fails, which is genuine specialization and argues against H2. Four showed cross-sample variance, a minority where repeated sampling helps. The five unsolvable tasks returned 0 of 4 for both models on every sample, a robust ceiling rather than bad luck, which argues against H3. We label both findings exploratory: n = 13, one resample set, tasks selected for disagreement. Incidentally, several tasks a model failed once but solved on three or four resamples, which implies single-attempt scoring slightly underestimates every model.

5.4 RECONCILIATION WITH IN-API ADVISOR RESULTS

Our results are consistent with, not contradicted by, concurrent industrial results for asymmetric coordination. On SWE-bench Pro, Anthropic's advisor tool [Anthropic, 2026a] lifts a Sonnet executor from about 75.5 percent to about 84 percent, approaching a Fable solo at about 91 percent, at roughly 63 percent of the flagship's cost. This is the same executor-plus-advisor pattern we benchmark. In our own pilot on LiveCodeBench, however, the advisor matched the flagship's accuracy but cost roughly 1.5 times more. The reconciliation is a single variable: consultation rate. On long-horizon agentic tasks the executor runs many mechanical turns and consults the advisor about once, so the bulk of tokens bill cheaply. On short single-shot puzzles the advisor is consulted on nearly every task, so the bulk of tokens bill at the advisor rate. Anthropic's own guidance states the pattern is a weak fit for single-turn work. Asymmetric routing is thus a genuine cost lever whose sign depends on task horizon. Two consequences follow for how our numbers should be read. First, our coordination costs reflect a short single-shot regime and are not the pattern's floor. The mesh is the exception, because it is inherently N times rounds expensive by design. Second, an in-API single-vendor advisor routes only within one provider, whereas the same decision made openly can place the cheapest sufficient model, from any provider, in either seat.

5.5 PRELIMINARY PROBES AND ONGOING WORK: RECOGNITION VERSUS DISSUASION

The floor-and-ceiling trade raises a mechanistic question the main study was not designed to answer. When a group beats or trails a member, is it because a model can *recognize* a correct answer it could not *generate*, and can a model also be *dissuaded* out of a correct answer it already held? We ran small probes that give a lead, not a result. Each is n = 12 on the contested tasks, a single sample, and selected for disagreement, so we report them to state the direction honestly and to scope a dedicated follow-up, not to settle the question here.

Recognition, the floor side. We showed a coordinator model three anonymized candidate solutions, from deepseek-chat, o3-mini, and gemini-2.5-flash, on the 12 tasks where those candidates disagreed, and asked it to pick one. The pick was then graded against the full hidden tests. The informative case uses the weakest model in the study, gemini-2.5-flash-lite, as the judge. It solves only 2 of the 12 itself, yet it picked the correct candidate 9 times, and on the 10 it could not solve alone it still picked correctly 7 times. Majority vote and the best single candidate each reached 7 of 12, and the oracle union is 12. Stronger judges scored higher still (gemini-3-flash 12 of 12, sonnet 7 of 10 completed), but each solves most of the set alone, so their picking is confounded with their solving. The weak judge is not, which is why it is the informative case. On this small sample, recognizing a correct answer was easier than producing one.

Dissuasion, the ceiling side. The mirror image is already visible in the main study. The symmetric five-model war room lost three problems that its own best member had solved alone (Section 4.2). A model arrived with the correct answer, saw the others, and was talked out of it. This is consistent with documented sycophancy and anchoring in multi-agent LLM debate.

The unifying hypothesis we intend to test. Exposure to peers' answers is double-edged, and the architecture sets the sign. When a model only *selects* among finished answers, exposure recovers correctness and raises the floor. When a model *revises* its own answer inside a shared debate, exposure can destroy correctness and lower the ceiling. Asymmetric coordination that picks but never overwrites should capture the first effect and avoid the second, while symmetric coordination invites both. Our main-study recipes are directionally consistent (Section 5.2): the asymmetric recipes preserved the ceiling while the symmetric war room did not.

We flag the threats plainly. n = 12, a single sample, tasks chosen for disagreement, and a picker that is itself imperfect (9 of 12, not 12). A properly powered study, across many models and tasks, with repeated sampling, and with select versus revise as an explicit manipulated factor, is future work we intend to publish separately. We include these probes here only to record the lead and to mark the question the main result opens.

6. When to use what

The floor and ceiling trade yields direct guidance.

Coordinate when you cannot run or cannot afford the top model, because a cheap group returns about 86 to 90 percent of frontier accuracy (at $0.021 to $0.145 per task in our runs, against $0.0555 to $0.1051 for the frontier models themselves, so check the arithmetic for your workload before assuming the group is the cheap option). Coordinate when you cannot identify the best model per task, because the group hedges the selection. Coordinate when worst-case reliability matters more than peak, as in production systems where a rare catastrophic answer is unacceptable. Coordinate on long-horizon tasks where asymmetric escalation is cheap.

Route to a single model when you know and can afford the best model, because that gives a higher ceiling, lower cost, and less complexity. In our study this was not a close call: always calling the strongest model beat every recipe that gated access to it, on accuracy and on cost. Route when peak accuracy on hard problems is the goal, so the ceiling is not capped. Route on short single-shot tasks where coordination overhead is high without a compensating ceiling. And if you do escalate, invest in the verifier before the closer, because a gate that cannot detect a wrong answer will not call the model you are paying it to call.

In short, coordination is insurance for when you cannot pick, and routing is knowing which single model to pick. The practical system is the one that decides, per task, which regime applies, and can show the receipt for that decision.

7. Limitations

First, n = 49, a single benchmark, a single domain (competitive programming), and a single-shot task format, which is the unfavorable regime for advisor and coordinator patterns, whose cost advantage appears on long-horizon workloads such as SWE-bench Pro [Anthropic, 2026a]. Pass-rate differences are directional, not individually significant. Second, single attempt at temperature 0 underestimates borderline tasks and does not exhaust repeated-sampling regimes. Third, a weak public-test verifier (one to three tests per problem) caps the realizable coverage of the recipes; Section 4.4 now quantifies this on the flagship cascade, where the gate cost four of the five tasks lost against the closer run solo (the fifth escalated and failed on a sandbox-environment mismatch). A stronger verifier, such as generated tests, cross-candidate consensus, or the user's own suite, would raise it, and this is the most important direction for future work. Fourth, cost estimates from recorded tokens can over-count for some providers, so reconcile against provider consoles for exact billing. Fifth, provider nondeterminism means outputs are not bit-reproducible even at temperature 0, so reproducibility is an overlapping confidence interval under a matching harness hash, not identical outputs. Sixth, the task-level anchoring result is from a single mesh run, so its exact magnitude awaits repeated runs. Seventh, the two strongest Anthropic results carry provenance caveats the reader should

weigh: claude-fable-5 and claude-sonnet-4-6 are contamination-exposed (2026-01 cutoffs postdate every task release), fable's 35 subscription-leg tasks ran without a pinned temperature, and sonnet's cost cell awaits a metered re-run. The clean-versus-overall split in Section 4.1 exists so no headline depends on an exposed model.

8. Conclusion

Across 49 execution-graded, contamination-audited coding tasks, multi-model coordination did not solve more problems than the best single model, and the margin grew when review forced the strongest closer onto the scoreboard as its own baseline. It did something more useful to characterize. It raised the floor and lowered the ceiling. A cheap model in a coordination recipe gained 9 to 10 problems and reached about 90 percent of the best clean model (86 percent of the best model overall) with no model selection, while a symmetric war room lost about 2 problems relative to its best member and, on three tasks, talked a correct answer out of the group. The trade is governed by shape, since asymmetric coordination preserves the ceiling only as far as its verifier can see (the flagship cascade lost four of its five dropped tasks to a gate that could not detect hidden-test failures, and the fifth to a failed escalation) and can be cheaper than a strong single model on long-horizon tasks, while symmetric coordination buys consistency at the cost of peak. It is also governed by task horizon, which sets the economics. None of the differences are individually significant at this scale, and we say so. The reproducible, cost-accounted method, the floor and ceiling structure, and the verifier-bottleneck finding are the contribution. The practical conclusion is neither "coordinate everything" nor "coordinate nothing," but a routing decision, made per task, and made in the open. Our next benchmark tests the regime this study deliberately excluded: long-horizon agentic work with complementary roles, one model reasoning, another generating, and a third verifying, over multi-step loops. That is where coordination is most likely to exceed the best single model, and where the honest case for it, if one exists, will be made. It will ship with the same contamination-free, receipt-backed method used here.

References

- Anthropic. (2026a). *Advisor tool* (developer documentation; beta header `advisor-tool-2026-03-01`). platform.claude.com/docs/en/agents-and-tools/tool-use/advisor-tool.
- Anthropic. (2026b). *Coordinator pattern: plan big, execute small* (Claude Cookbooks, managed agents). github.com/anthropics/claude-cookbooks.
- Chen, L., Zaharia, M., & Zou, J. (2023). *FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance.* arXiv:2305.05176. (TMLR, 2024.)
- Ding, D., Mallick, A., Wang, C., Sim, R., Mukherjee, S., Rühle, V., Lakshmanan, L. V. S., & Awadallah, A. H. (2024). *Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing.* ICLR 2024. arXiv:2404.14618.
- Du, Y., Li, S., Torralba, A., Tenenbaum, J. B., & Mordatch, I. (2023). *Improving Factuality and Reasoning in Language Models through Multiagent Debate.* arXiv:2305.14325. (ICML 2024.)
- Jain, N., Han, K., Gu, A., Li, W.-D., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K., & Stoica, I. (2024). *LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code.* arXiv:2403.07974.
- McNemar, Q. (1947). *Note on the sampling error of the difference between correlated proportions or percentages.* Psychometrika, 12(2), 153 to 157.
- Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., & Stoica, I. (2024). *RouteLLM: Learning to Route LLMs with Preference Data.* arXiv:2406.18665.
- Sakana AI. (2026). *Sakana Fugu: One Model to Command Them All* (Fugu / Fugu Ultra technical report). sakana.ai/fugu-release; arXiv:2606.21228.
- Wang, J., Wang, J., Athiwaratkun, B., Zhang, C., & Zou, J. (2024). *Mixture-of-Agents Enhances Large Language Model Capabilities.* arXiv:2406.04692.
- Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., & Zhou, D. (2022). *Self-Consistency Improves Chain of Thought Reasoning in Language Models.* arXiv:2203.11171. (ICLR 2023.)
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.* arXiv:2201.11903. (NeurIPS 2022.)

- Wilson, E. B. (1927). *Probable inference, the law of succession, and statistical inference.* Journal of the American Statistical Association, 22(158), 209 to 212.

All quantitative results derive from per-task execution receipts retained by the benchmark harness; metered results are reproducible with the reader's own API keys, and the two subscription-channel runs are flagged where they appear. Revised 2026-07-10 to add the claude-fable-5 baseline requested in review, the per-model contamination audit, and the verifier-bottleneck analysis (Section 4.4); the underlying receipts predate the revision. ATO is an open-box agentic model router. Learn more and reproduce this benchmark at agentictool.ai.

Appendix A. Task-level results

Pass (P) / fail (F) for all 49 tasks across the nine single models and the five full-run recipes (harness 6169df13; fable assembled from its two flagged legs; mesh assembled from its seven 7-task chunks). Every union, discordance, and domination number in the paper can be recomputed from this table by eye.

RAFT · AT

task	fable-5	gem-3f	gpt-5	ds-r	son-4-6	gem-2.5f	o3-mini	ds-chat	g2.5lite	cascd>f	refine	mesh-5	cascs>f	self-cons
3692	P	P	P	P	P	P	P	P	F	P	P	P	P	P
3696	P	P	P	F	P	F	F	F	F	F	F	F	P	F
3697	P	F	P	P	F	P	P	P	F	P	P	P	F	P
3701	P	P	P	P	F	F	F	F	F	F	F	F	F	P
3705	P	P	P	P	P	P	P	P	P	P	P	P	P	P
3709	P	P	P	P	P	P	P	P	P	P	P	P	P	P
3717	P	P	F	P	F	F	F	F	F	P	P	P	F	F
3722	P	P	P	P	P	P	P	P	F	P	P	P	P	F
3723	P	P	P	P	P	P	P	P	P	P	P	P	P	P
3733	P	F	P	P	P	P	P	F	F	P	P	P	P	P
3736	P	P	P	P	P	P	P	P	P	P	P	P	P	P
3743	P	P	P	P	P	P	F	P	F	P	F	P	F	P
3744	P	P	P	P	P	P	P	P	F	P	P	P	P	F
3748	P	P	P	P	P	P	P	P	F	P	P	P	P	P
3750	F	F	F	F	F	F	F	F	F	F	F	F	F	F
3753	P	P	P	P	P	P	P	P	F	P	P	P	P	P
3754	P	P	P	P	P	P	P	F	P	P	P	P	P	P
3759	F	F	F	F	F	F	F	F	F	F	F	F	F	F
3760	F	F	F	F	F	F	F	F	F	F	F	F	F	F
3762	P	P	F	F	F	F	F	F	F	P	F	F	P	F
3763	F	F	F	F	F	F	F	F	F	F	F	F	F	F
3764	P	P	P	P	P	P	P	P	P	F	P	P	P	F
3765	P	P	P	P	P	F	F	P	F	P	P	F	F	F
3768	P	P	P	P	P	P	P	P	P	P	P	P	P	P
3770	P	P	P	P	P	P	P	P	F	P	P	P	P	F
3771	P	P	P	P	P	P	F	P	F	P	P	P	P	F
3773	P	P	P	P	F	F	P	P	P	P	P	P	P	P
3776	P	P	F	F	P	P	P	F	F	P	P	P	P	P
3777	P	P	P	F	F	F	P	F	F	P	F	F	F	F
3778	P	P	P	P	P	P	P	P	P	P	P	P	P	P

DRAFT.AI

task	fable-5	gem-3f	gpt-5	ds-r	son-4-6	gem-2.5f	o3-mini	ds-chat	g2.5lite	cascd>f	refine	mesh-5	cascs>f	self-cons
3779	P	P	P	F	P	F	F	P	F	P	P	F	P	P
3781	P	P	F	P	F	F	F	F	F	P	F	F	P	F
3783	P	P	P	P	P	P	P	F	F	P	P	P	P	F
3784	F	F	F	F	F	F	F	F	F	F	F	F	F	F
3785	P	P	P	P	P	P	P	P	P	P	P	P	P	P
3786	P	P	P	P	P	P	P	P	F	P	P	P	P	P
3788	P	P	P	P	P	F	P	F	F	F	P	P	P	F
3789	P	P	P	P	P	F	F	F	F	P	P	P	F	F
3791	P	P	P	P	P	P	F	F	F	P	P	P	P	F
3793	P	P	P	P	P	P	P	P	F	P	P	P	P	P
3794	F	P	P	P	P	P	F	F	F	F	P	P	P	P
3795	P	F	P	P	F	F	P	F	F	F	F	P	F	P
3799	P	P	P	P	P	P	P	P	P	P	P	P	P	P
3801	P	P	P	P	P	P	P	P	F	P	P	P	F	P
3805	P	P	P	P	P	P	F	F	F	P	P	P	P	P
3809	P	P	P	P	P	P	P	P	P	P	P	P	P	P
3811	P	P	P	P	P	P	P	P	P	P	P	P	P	P
3817	P	P	P	P	P	P	P	P	P	P	P	P	P	P
3832	P	P	P	P	P	P	P	P	P	P	P	P	P	P
total	**43**	**41**	**40**	**39**	**36**	**32**	**31**	**28**	**15**	**38**	**37**	**37**	**35**	**29**

Key paired discordances (row wins / column wins): fable vs cascade ds>f = 5/0 (p = 0.0625); fable vs gemini-3-flash = 3/1 (p = 0.625); deepseek-reasoner vs mesh-5 = 3/1. Tasks solved by no system: 3750, 3759, 3760, 3763, 3784.

RAFT.AI