DECENTRALISED MULTI-AGENT ORCHESTRATION

Decentralised Multi-Agent Task Allocation via Signal-Driven Selection

A biomimetic framework inspired by cryptic female choice. This document holds the research introduction and methodology. It is self-contained on the essentials and refers to `claude.md` and `docs/theory.md` for the full model.

Chemotactic Task Allocation · Reproducible from seeds · 127 tests, ruff clean

≈40× cheaper at equal completion (agent wrapper)	**0.0 / 2.0** peak-load growth exponent, decentralised vs central	**12** hypotheses; 10 synthetic, 2 on real agents	**114** real-agent attempts behind the calibration result

Executive summary

Multi-agent systems built on large language models are held back by two costs. The first is coordination: a central orchestrator that scores every agent against every task becomes both the throughput bottleneck and the largest bill as the fleet grows. The second is trust: these systems let an agent take work based on the agent's own estimate of its fit, and language-model agents are miscalibrated about their own success, so an allocation built on raw self-reports misallocates.

Chemotactic Task Allocation (CTA) removes the first cost by decentralising the decision. Tasks advertise a contract, an agent takes a task only when its self-scored compatibility clears an activation threshold, the winner is chosen by a cost-adjusted, reliability-weighted bid, and an integrity gate screens that winner before it can act. There is no central scheduler to saturate. It handles the second cost by never trusting the self-report on its own: each bid is discounted by the agent's observable track record, so competence, not just confidence, drives who gets the work.

Across a pre-registered simulation study at scale, a fleet parameterised from measured model calibration, and live experiments on real Claude agents, three results stand out. Decentralisation holds the busiest node's coordination load flat as the fleet grows to ten thousand agents, where a central scheduler grows quadratically. Miscalibration is indeed the failure mode of self-selection, and a track-record correction that uses no privileged information recovers the completion it costs. Most consequentially for practice, two thin wrappers around the mechanism turn it into a cost result on real agents: a **task wrapper**, an explicit interface contract, lifts a cheap model to a frontier model's completion and is the precondition for independently built pieces to integrate into a working whole; and a **calibrated agent wrapper** then routes each task to the cheapest model that can be trusted with it, holding frontier-level completion at roughly one fortieth of the always-frontier cost.

The practical thesis is that a swarm of cheap, wrapped microagents, coordinated by calibrated self-selection, can match a single expensive model's output at a fraction of its cost. The evidence for the coordination and calibration mechanisms is at full protocol scale; the real-agent wrappers are now powered (the ladder about ten agents per cell, the project five) with confidence intervals, and the paper marks the residual limits (a bespoke task set, representative prices) throughout. Everything is deterministic and reproducible from seeds by one command. A companion results dashboard visualises the whole arc (`results/showcase.html`).

Abstract

Multi-agent systems built on a central orchestrator concentrate two costs at one node: it evaluates candidates for every task, and its throughput caps the system's. We study Chemotactic Task Allocation (CTA), a decentralised biomimetic framework in which tasks advertise semantic envelopes and agents self-select when their compatibility clears an activation barrier, with the winner chosen by a cost-adjusted, reliability-weighted Binding Energy and screened by an integrity gate before write access. The framework's load-bearing assumption, that an agent can assess its own fit, is exactly where language-model agents are known to be miscalibrated, so we treat the self-report as a first-class, swept variable rather than an idealisation. Across a pre-registered protocol on synthetic populations, a fleet grounded in measured model calibration, and a live pilot with real coding agents, we find that miscalibration is the failure mode of self-selection and that a track-record correction using no privileged information recovers the completion it costs, with the gate as a safety backstop against adversarial agents. The decentralised design holds peak per-node load flat (fitted growth exponent 0.0) while a central scheduler grows as `N` times `M` (exponent 2.0) out to ten thousand agents, which at representative model prices is a roughly two-hundred-and-fifty-fold coordinator cost saving. Against a full-information optimum the quality claim does not clear its pre-registered margin and is reported as such; against the information-bounded central scheduler a deployment would actually run, decentralised self-selection matches it with fresh information and overtakes it as the coordinator's reliability table goes stale. A biomimicry ablation isolates the mechanisms: the integrity gate carries the safety effect, and the activation barrier, quality-neutral in a homogeneous batch, earns a decisive routing role on heterogeneous work, keeping every won subtask with a correct specialist (routing accuracy 1.00 against a 0.25 chance floor) where removing it collapses routing under bounded observability. Finally, on real Claude agents spanning three model families, the same two mechanisms become a cost result: the task wrapper, an explicit interface contract, lifts a weak model to the frontier model's completion and is the precondition for independently built modules to integrate into a working project, and the calibrated agent wrapper then routes to the cheapest capable model, holding frontier-level

completion at roughly one fortieth of the always-frontier cost, but only when the tasks are wrapped, so the two wrappers are complementary. All results are deterministic and reproduced from seeds by one command; the real-agent wrappers are powered (the ladder about ten agents per cell, the project five) with bootstrap confidence intervals.

---

1. Introduction

Multi-agent systems increasingly coordinate large numbers of autonomous agents over many concurrent tasks. A common pattern places a central orchestrator in charge of assigning work from the top down. This pattern is simple to reason about, yet it concentrates two costs at one point: the coordinator must evaluate candidates for every task, and its throughput caps the throughput of the whole system. As the agent and task counts grow, the central node becomes a scaling bottleneck and a single point of failure.

Decentralised alternatives have a long record. The Contract Net Protocol (Smith, 1980) distributes control through an announce, bid, and award exchange, although the award decision for each task still passes through a manager. Market and auction based allocation, surveyed and analysed by Dias, Zlot, Kalra, and Stentz (2006) and formalised by Gerkey and Matarić (2004), casts allocation as an optimisation problem and clarifies where decentralised methods trade optimality for scalability. Stigmergic methods such as ant colony optimisation (Dorigo, Maniezzo, and Colorni, 1996) coordinate indirectly through signals left in a shared environment, with no central assigner at all. These results establish that decentralised allocation is viable. Three gaps remain visible across much of this work: the award step is often still centralised per task, a trust or reliability screen before execution is rarely integrated, and there is seldom a principled account of tasks that cannot be done given the agents present.

This project studies a decentralised framework, Chemotactic Task Allocation (CTA), that addresses those three gaps, drawing its selection principle from two biological sources. From cryptic female choice it takes signal-driven, post-advertisement selection: selection continues after the initial encounter and is biased by chemical signalling rather than decided by a single authority (Fitzpatrick et al., 2020). From the response threshold model of division of labour in social insects it takes the activation barrier: an individual engages a task only when the task stimulus exceeds its threshold (Bonabeau, Theraulaz, and Deneubourg, 1996), a mechanism since operationalised in swarm robotics. Translated to computation: each task is described by a wrapper (the scent envelope) that reads an agent's role, skills, and prompt against the task requirements to produce a compatibility score `c` in [0, 1]. Distributed agents self-select: an agent may take a task only when its compatibility reaches the task's activation energy (`c >= Ea`), and among the willing agents the winner maximises a cost-adjusted, reliability-weighted score, the Binding Energy `B = c x C_tilde / L`. Selection proceeds in two stages, a binary

eligibility filter and then the activation barrier, followed by a distinct trust stage, the Rejection Gate, modelled on the zona pellucida. Every quantity is defined operationally in `docs/measures.md`. The biology and chemistry are used as design intuition, not as literal specifications, and the limits of the analogies are recorded in `docs/theory.md` section 8.

Compared with a simple pull-based work queue, where agents self-schedule by taking the best available task, CTA adds three things: the activation barrier as a quality floor, the Rejection Gate as a trust boundary, and the eligibility filter with its infeasible and stalled semantics. The evaluation isolates these additions with a pull-based baseline (section 2.3), so a result is not merely the effect of decentralisation.

Any framework in which agents self-select rests on a load-bearing assumption: that an agent can assess its own fit for a task. Recent studies of auction and bidding among large language model agents report that self-reported success probabilities are systematically miscalibrated, so an allocation built from self-reports diverges from the full-information optimum (Fradkin and Krishnan, 2026), and calibrating that self-confidence is an open problem (Zhang et al., 2026). This is exactly the assumption CTA's compatibility bid depends on. We therefore treat self-assessment as a first-class, swept variable rather than an idealisation. The wrapper's compatibility is the agent's self-report, drawn from its true fit with a bias and noise (E13), while realised quality depends on the true fit and the agent's competence, so miscalibration corrupts the allocation without changing the ground truth. Two questions follow. First, how far does miscalibration degrade a self-selection allocation. Second, whether an observable track record (the reliability `R`, E4), which needs no privileged information, recovers the loss. The Rejection Gate then serves as a safety backstop, screening the winner for scope integrity before it gains write access. This repositions the contribution from decentralised self-selection, which is now well covered, onto the calibration robustness of self-selection, which is not.

Cost-aware model selection is an active line: LLM cascades and learned routers send easy queries to a cheap model and hard ones to a strong one, cutting inference cost several-fold without much quality loss (FrugalGPT: Chen, Zaharia, and Zou, 2023; RouteLLM: Ong et al., 2024). CTA's agent wrapper is a router in this family but differs on three points. First, it corrects the routing signal for miscalibration: the bid is the model's own self-report discounted by an observable track record (the reliability `R`, E4), rather than a static capability estimate or a preference-trained classifier, so it stays honest when a model is wrong about itself, which MarketBench (Fradkin and Krishnan, 2026) shows they often are. Second, it routes decomposed work: paired with the task wrapper's interface contract it assembles a single project from modules built by different models, which whole-query routers do not address. Third, it rides a decentralised substrate whose coordination cost stays flat as the fleet grows. A head-to-head against these routers on a shared benchmark is the natural next step and is marked as such.

Contributions:

1. A decentralised allocation framework in which tasks advertise semantic envelopes and agents self-select by affinity, reducing per-task coordination to an atomic claim rather than a central assignment.
2. A two-stage selection model that separates categorical capability (eligibility) from graded affinity (activation energy), and that yields a principled distinction between infeasible tasks (no eligible agent) and stalled tasks (eligible agents present, none clearing `Ea`).
3. A reliability and integrity trust gate that screens degraded and out-of-scope agents before they gain write access, evaluated as a safety backstop against adversarial agents.
4. An evaluation methodology that compares the framework against centralised baselines on both scaling and match quality, with operational metrics. It includes an information-bounded central scheduler (allocating from self-reports and a reliability table that lags with batch synchronisation) as the fair, beatable comparison the full-information optimum cannot provide, and a token and dollar cost model that turns the coordinator work into money, so the scaling result carries a commercial reading.
5. A formal framework (Chemotactic Task Allocation) that unifies eligibility, affinity, and activation into one model, with cost accounting that separates total system work from coordinator work.
6. A measurable compatibility wrapper that scores an agent's role, skills, and prompt against a task and can be calibrated to predict success, so the activation threshold rests on an empirically validated score rather than an abstract one (`docs/measures.md`).
7. A study of self-assessment miscalibration as the failure mode of self-selection: the compatibility bid is treated as the agent's self-report (E13), and a track-record correction (the reliability `R`, E4), which uses no privileged information, recovers the completion that miscalibration costs, with the integrity gate as a safety backstop.
8. External validity in three steps: a mixed fleet whose calibration archetypes are parameterised from measured LLM behaviour (MarketBench); a live pilot with real Claude coding agents that measures their calibration on coding micro-tasks directly, confirming on real data that self-reports are miscalibrated and that the sign is model-dependent; and a capability ladder across three real model families (Haiku 4.5, Sonnet 5, Opus 4.8) on a harder expert tier that measures what the framework's task wrapper and agent wrapper buy, the task wrapper lifting the weakest model to the frontier's completion and the agent wrapper then routing to it to hold frontier completion at about one forty-seventh of the cost (`docs/live_pilot.md`).
9. Evidence that the activation barrier is a routing discipline, not only a quality floor: on a heterogeneous job whose subtasks need different specialists, the barrier keeps allocation on

target, sending each won subtask to a correct specialist because an ill-matched agent does not fire, at a coverage cost that annealing later relaxes (H10).

0. A self-improvement result that reads the loop as a harness (Weng, 2026): a persistent, accumulating track record makes the allocation improve from its own experience, lifting completion over rounds toward the full-information optimum with no privileged information, while a memoryless baseline stays flat (H13). This positions the calibration correction as the trustworthy-signal layer a self-improving harness needs.

Research questions:

- RQ1: Does signal-driven self-selection reduce per-task coordinator work and allocation latency, relative to a central assigner, as the population grows?
- RQ2: Does it do so without materially degrading match quality?
- RQ3: Does the two-stage model correctly separate infeasible from stalled tasks?
- RQ4: Does the Rejection Gate prevent out-of-scope writes when some agents are adversarial?
- RQ5: Is allocation stable across the range of activation barriers and temperatures, without oscillation or starvation?
- RQ6: Does CTA's advantage depend on agent heterogeneity, and if so, how does it change from a homogeneous to a specialised population?
- RQ7: How far does self-assessment miscalibration degrade a decentralised self-selection allocation?
- RQ8: Does a track-record correction, using only observable history, recover the allocation quality that miscalibration costs?
- RQ9: Does decentralised self-selection match or beat a central scheduler operating under realistic bounded information (allocating from self-reports and a reliability table that lags with batch synchronisation), even though it cannot beat a full-information optimum?
- RQ10: On a heterogeneous job whose subtasks need different specialists, does the activation barrier route each subtask to a correct specialist, and what does it cost in coverage?
- RQ11: Does wrapping a task in an explicit interface contract and acceptance criteria (the task wrapper) raise a weak model's completion toward the frontier and let independently built modules integrate?
- RQ12: Does routing each task to the cheapest model whose reliability-corrected self-report clears the barrier (the agent wrapper) hold frontier-level completion at a fraction of the always-frontier cost, and does that saving depend on the task being wrapped?

Scope and non-goals: this is a simulation study, not a live deployment. The atomic claim assumes a single logical coordination store providing compare-and-swap, so the framework reduces

central work rather than removing it entirely (discussed in `docs/theory.md` section 6). The security model covers reliability scoring and scope integrity, and it extends to one form of gaming: an adversary that inflates its self-reported compatibility to win bids is demoted by the track record as its realised failures accrue (the strategic-adversary result in section 3). A second form, reputation gaming, in which an adversary does genuine work to build a clean record before defecting (sandbagging), is modelled in section 3: the plain cumulative track record lags, so the adversary exploits its reputation for a few rounds, and a recency-weighted reliability window shortens that exploitation while the scope-based gate backstops out-of-scope defections and an exposure cap bounds the first defection's blast radius. The residual limitation is that no reactive record can pre-empt the first defection itself, only bound it, and collusion to launder reputation across agents is left as future work.

---

2. Methodology

2.1 Research design

The study is a comparative, controlled experiment. Two systems are measured under identical task and agent populations: the decentralised framework and a centralised baseline. The scoring function (Binding Energy) is shared by both systems, so the only difference is how allocation is coordinated. This isolates the effect of decentralisation. Population size is varied to test the scaling hypothesis, and each configuration is run for many seeded replications to support interval estimates.

The experiment runs in two modes that share one core. A simulation mode uses many synthetic agents in Python for scale, determinism, and cheap sweeps, and produces the scaling curves for H1 to H5. A real-swarm pilot uses a small set of Claude Code subagents competing over a shared task pool in a self-contained store (SQLite by default), where the atomic claim is a genuine compare-and-swap, and it supplies ecological validity: real self-assessment, latency, cost, and output quality. A first version of this pilot has been run, measuring real-agent calibration on coding micro-tasks (section 3, live pilot; `docs/live_pilot.md`); the full competing-swarm allocation is the remaining step. The shared core (scent schema, scoring module, event log, and metric code) makes the two modes comparable. The full design is in `docs/architecture.md`.

2.2 Formal framework (Chemotactic Task Allocation)

Entities: a set of agents `A` of size `N`, and tasks `T` of size `M` arriving over time. Each task carries a scent envelope `sigma(t) = (d_t, ReqCap_t, ReqPerm_t, ReqTool_t, Ea_t, rho_t, scope_t)`, where `d_t` is the domain, `Ea_t` the activation barrier (default 0.2), and `rho_t` the priority. Each agent has a capability profile, permissions, tools, and a reliability history.

Stage one, eligibility (binary):

- (E1) `elig(a,t) = 1[ReqTool_t subset Tool_a] . 1[scope_t subset Perm_a]`, valued in {0,1}: tools and permitted scope are the hard requirements. Skills and domain are graded into compatibility (E3), not vetoed. See `docs/measures.md` section 2.
- (E2) eligible set `A(t) = { a in A : elig(a,t) = 1 }`. The task is infeasible when `A(t)` is empty.

Compatibility, capability, and the selection score:

- (E3) compatibility `c(a,t)` in [0, 1], produced by the task wrapper from the agent's role, skills, and prompt against the task requirements. It aggregates measurable sub-scores (semantic match, skill coverage, scope fit) by a weighted geometric mean, or by a logistic model calibrated to predict success. Compatibility replaces the abstract signal `S`; full definitions are in `docs/measures.md` section 3.
- (E4) reliability `R(a) = (s_a + 1) / (n_a + 2)`, a Laplace smoothed success ratio over a sliding window of `W` recent attempts.
- (E5) effective capability `C_tilde(a,t) = C(a,t) . R(a)`, with base capability `C` in [0, 1], so reliability enters selection as well as the gate.
- (E6) selection score, the Binding Energy `B(a,t) = c(a,t) . C_tilde(a,t) / max(L(a,t), eps)`, with cost `L > 0` and floor `eps = 0.01`. `B` ranks the agents that have cleared activation; it does not gate firing. In prose `B` is written `BE`, and `L` is a normalised relative cost with a typical value near 1. The activation barrier `Ea` in [0, 1] is compared against compatibility `c` (E7), which is bounded in [0, 1] by construction, so `Ea` is directly interpretable.

Stage two, activation (firing):

- (E7) activation drive `Delta(a,t) = c(a,t) - Ea_t`; the barrier is on compatibility, not on the selection score.
- (E8) firing probability `P_fire(a,t) = 1` when `Delta >= 0`, and `P_fire(a,t) = exp(Delta / T)` when `Delta < 0`, with temperature `T >= 0`. The deterministic threshold is the `T -> 0` limit. This is the Boltzmann or Arrhenius form; the response threshold sigmoid `s^n / (s^n + theta^n)` is an equivalent soft-threshold family.
- (E9) firing set `F(t) = { a in A(t) : u_a <= P_fire(a,t) }`, with `u_a` drawn uniformly on [0, 1]. The task is stalled when `A(t)` is non-empty and `F(t)` is empty.

Claim and trust:

- (E10) winner `a*(t) = argmax over a in F(t) of B(a,t)`; the tie breaker is the lower `L`, then the lower agent identifier. The claim is committed by an atomic compare-and-swap, so exactly one agent wins.
- (E11) the Rejection Gate admits `a*` when `R(a*) >= tau` and `integrity(a*, scope_t) = 1` (acceptance threshold `tau`, default 0.6); otherwise it deflects and the task is re-advertised.

Outcome and feedback:

- (E12) realised quality, a ground truth independent of the agent's self-estimate, `Q(a,t) = clip(g(S_true, C_true) + xi)` with noise `xi ~ N(0, sigma_q^2)`, in [0, 1]. The attempt succeeds when `Q >= q_min`, which updates `(s_a, n_a)` and hence `R`.
- (E13) self-assessment: agents act on noisy estimates `c_hat = clip(c + eta_c)`, `C_hat = clip(C + eta_C)`, and `L_hat = max(eps, L + eta_L)`, with `eta ~ N(b, sigma^2)` for bias `b` and noise `sigma`. Activation and selection use the estimates; outcomes use the truth through E12.
- (E14) annealing: while a task is stalled, `Ea_t <- max(Ea_min, Ea_t - delta)` per waiting round, so the barrier relaxes rather than starving the task.

Cost accounting:

- (C1) total evaluation work `W_eval = sum over t of |A(t)|`, worst case `O(N.M)`, since eligible agents each score the task.
- (C2) coordinator work `W_coord = sum over t of |F(t)|`, the claim attempts contending at the shared store.
- (C3) communication, messages per allocated task. The centralised baseline instead incurs a per-round assignment cost, `O(k^3)` for the Hungarian method over `k` candidates.

The full narrative for these equations is in `docs/theory.md`.

2.3 Baseline

The centralised baseline is a single scheduler that, each round, assigns tasks to agents using the same scores. Three variants are used. Two are one-to-one assignments (each agent takes at most one task): an optimal Hungarian assignment (Kuhn, 1955) and a greedy highest-affinity assigner, both over Binding Energy, which serve as the coordination-cost reference for the scaling hypothesis. The third, `central_best`, is the full-information quality optimum for CTA's non-exclusive setting: it gives each task the agent that maximises expected realised quality (true fit times true capability) and allows an agent to take more than one task, exactly as CTA does. Because the one-to-one optima are forced to spread work across weaker agents, they understate

the achievable quality and would let CTA appear to beat the optimum; `central_best` is the fair upper bound that CTA should approach from below, and it is the reference for the quality hypothesis H2.

A fourth baseline, `central_bounded`, is the honest central scheduler a deployment would actually run. Unlike `central_best`, it does not see true fit or true competence: it allocates from each agent's self-reported compatibility (miscalibrated by the agent's own bias and noise, E13) and a reliability estimate that is possibly stale, because a central table is synchronised in batches and lags, whereas a decentralised agent acts on its own current record. A staleness parameter in `[0, 1]` blends each agent's current reliability with an out-of-date value drawn from the prior, so more staleness progressively scrambles the competence ranking. With staleness and observation noise zero and a well-calibrated fleet, it reduces to a fully informed reliability-weighted central auction. It is the reference for H9: the full-information optimum is unbeatable by construction, so `central_bounded` is the fair, beatable central baseline that isolates centralised versus decentralised coordination rather than the information each is handed.

A third, decentralised baseline is a pull-based work queue: agents self-schedule by claiming the best available eligible task, without the activation barrier or the Rejection Gate. It removes the central assigner just as CTA does, so comparing CTA against it isolates the effect of CTA's specific mechanisms (the barrier, the gate, and the infeasible and stalled semantics) from the effect of decentralisation alone. Without this baseline, a scaling result could be attributed merely to avoiding an expensive central optimiser.

2.4 Variables and metrics

Independent variables: number of agents `N`, number of tasks `M`, task arrival rate, the distribution of `Ea`, temperature `T`, the reliability acceptance threshold, and the injected fraction of unreliable agents.

Independent variables also include the self-assessment bias `b` and noise `sigma` (E13), so that miscalibrated agents can be studied directly, and the degree of agent capability heterogeneity, from a homogeneous population (agents are near-interchangeable) to a specialised one (agents cover distinct domains). Heterogeneity is expected to matter because self-selection has little to exploit when agents are interchangeable.

Dependent variables, with operational definitions:

- Allocation latency: simulated time from advertisement to a successful claim.
- Coordinator work: `W_coord` (E9, C2), the claim attempts per allocated task, distinct from total evaluation work `W_eval` (C1) and from communication, messages per allocated task (C3). Reporting all three separates the bottleneck from the total compute.

- Claim contention: attempts per successful claim, and the wasted-evaluation rate, the share of firing agents that do not win.
- Herding: the spread of the firing-set size `|F(t)|` across tasks (for example a Gini coefficient over tasks), which detects convergence onto a few attractive tasks.
- Match quality: the realised quality `Q` of winning agents (E12), with mean Binding Energy reported alongside as the proxy.
- Infeasible rate and stall rate: fractions of tasks in each outcome, checked against the generator's ground truth.
- Deflection rate: fraction of claims the gate rejects, with false deflection tracked separately.
- Calibration sensitivity: match quality and deflection rate as functions of the self-assessment bias and noise.
- Completion rate: the fraction of tasks completed successfully, the primary outcome for the calibration-robustness hypotheses (H7, H8), compared across the raw, reliability, and full-information selection modes.
- Overconfidence gap: the mean self-report of winners minus their mean realised quality, the direct measure of miscalibration (H7).
- Calibration error: the Brier score and the expected calibration error (ECE) of the winners' self-reports against their binary success, the standard calibration measures used in the auction and forecasting literature (H7).
- Track-record length: the number of prior attempts behind the reliability estimate, swept to show how much history the correction needs before it is effective.
- Integrity violations: out-of-scope writes that execute when the gate is absent, the safety measure for the gate ablation (H4).
- Load fairness: distribution of completed tasks across agents (Gini coefficient).
- Stability and starvation: the maximum stall time of feasible tasks in the temporal engine, with and without annealing, and the annealing rate needed to bound it.
- Starvation: maximum time any task waits in the pool.
- Throughput: completed tasks per unit time.
- Scaling curves: allocation latency and coordinator work as functions of `N`.

2.5 Procedure

Agent and task populations are drawn from controlled distributions with fixed random seeds. Each run includes a warm-up period that is excluded from measurement. Each configuration is repeated across seeded replications, and for each headline comparison the seeds a power analysis requires to detect the observed effect are reported alongside the seeds actually run, so a null result reflects the framework rather than insufficient power. The headline calibration effects are

large, so the reported runs at twenty seeds comfortably exceed the power requirement. Raw event logs are retained so that every derived metric can be recomputed from source. The deterministic firing rule is used for exact replication, and any Arrhenius runs record their seeds and temperature.

2.6 Analysis

Results are reported as means with 95 per cent confidence intervals; for the headline quality and completion metrics, which are bounded and skewed, the interval is a percentile bootstrap rather than a normal approximation. Rather than fix the seed count by hand, each headline comparison reports the seeds per arm a power calculation (alpha 0.05, power 0.8) requires to detect the observed effect, and records whether the run met it, so a null result reflects the framework and not insufficient power. Systems are compared per metric. For scaling, growth of coordinator work against `N` is fitted in log-log space and its exponent reported with a bootstrap confidence interval, and compared between systems. Given that latency distributions are typically skewed, non-parametric tests (for example the Mann-Whitney U test) are preferred, and effect sizes are reported alongside p-values. Hypotheses and acceptance margins are fixed in advance to reduce the risk of post-hoc selection. H1 and H2 are the pre-registered primary endpoints; the remaining hypotheses are secondary, and their p-values are corrected for multiple comparisons with a Holm-Bonferroni procedure across the reported family of tests. Experiments are driven by an automated research loop with a protected, pre-registered metric and an append-only decision ledger (see `docs/roadmap.md`), which supports reproducibility and guards against metric-hacking. To keep long campaigns reliable, the loop treats the logged record as an external environment to query and recursively summarise rather than holding it all in context, which mitigates context rot (Recursive Language Models: Zhang, Kraska, and Khattab, 2025). This external-memory strategy is a property of the research process, not of the Chemotactic Task Allocation framework under study.

Hypotheses:

- H1: coordinator work per task (C2) and allocation latency grow more slowly with `N` for the framework than for the centralised baseline, and comparably to the decentralised pull-based baseline. Total evaluation work (C1) and communication (C3) are reported alongside and are expected to be comparable or higher, so the claim is bottleneck relief, not less total compute. Since decentralisation alone relieves the bottleneck, CTA's distinct value is holding quality and safety while scaling, tested in H2 and H4.
- H2: mean realised quality `Q` (E12) is within a pre-registered margin (0.05) of the full-information optimum and at least the pull-based baseline. The optimum is `central_best`, which gives each task the agent that maximises expected realised quality (true fit times true

capability) with agent reuse allowed, matching CTA's non-exclusive setting rather than a one-to-one assignment that is handicapped by forced spreading.

- H3: the framework labels a task infeasible exactly when no eligible agent exists, and stalled exactly when eligible agents exist but none clears `Ea`.
- H4: with adversarial agents that attempt out-of-scope actions, the Rejection Gate substantially reduces integrity violations relative to an ablation without the gate. The gate detects an out-of-scope action with recall below one, so the result is a measured reduction, not a tautological zero.
- H5: activation-energy annealing (E14) bounds the stall time of feasible tasks. In the temporal engine, a positive annealing rate relaxes the barrier so every feasible task is resolved at bounded stall, whereas without annealing the stalled tasks are never claimed.
- H6: CTA's advantage over both baselines increases with agent heterogeneity, and narrows towards zero in a homogeneous population where self-selection has little to exploit.
- H7: winners' self-reports of fit systematically over-predict the quality they deliver (a materially positive overconfidence gap), because the self-report omits competence, so relying on the self-report alone is a genuine failure mode. We do not claim the gap grows with the injected bias; in this model it is dominated by the structural fit-versus-competence gap.
- H8: discounting the self-report by an observable track record (the reliability `R`, E4) recovers task completion relative to the raw self-report auction, under the worst injected overconfidence.
- H9: CTA matches or beats a central scheduler operating under bounded information, that is, one that allocates from the agents' self-reports and a reliability table that lags because it is synchronised in batches rather than kept current at each agent. This is the fair form of the H6 comparison: the full-information optimum is unbeatable by construction, so H9 replaces it with the honest central baseline a deployment would actually run, given both conditions the same miscalibrated fleet and track record. Supported when, once the central table is stale, CTA's per-seed realised quality is significantly higher.
- H10: on a heterogeneous job of role-specific subtasks over role-specific specialists, the activation barrier routes each subtask to a correct specialist, because an ill-matched agent does not fire. Supported when, with the barrier, the fraction of won tasks that go to a same-domain specialist is near one and far above the chance floor (`1 / n_roles`), and materially above the no-barrier baseline in which every agent fires on the tasks it observes regardless of fit.

Falsification: the thesis is not supported if coordinator work and latency grow at the same order for the framework as for the centralised baseline (H1), if realised quality falls below the pre-registered margin of the Hungarian optimum or does not exceed the pull-based baseline (H2), if

the gate does not substantially reduce out-of-scope writes by adversarial agents (H4), if annealing does not bound the stall time of feasible tasks (H5), if the advantage does not appear even in the specialised, high-heterogeneity regime (H6), if self-reports are well calibrated so no overconfidence gap appears (H7), if the track-record correction does not recover completion under miscalibration (H8), if CTA fails to match the information-bounded central scheduler once its reliability table is stale (H9), if the activation barrier does not keep won subtasks routed to a correct specialist above the no-barrier baseline (H10), or if a persistent, accumulating track record does not lift completion over rounds toward the full-information oracle beyond a memoryless raw baseline (H13). These outcomes are reported as stated, not reframed.

2.7 Validity and threats

- Internal validity: the shared scoring function and fixed seeds isolate the coordination effect. Threats: implementation bias between the two systems, and claim contention that could confound latency at scale. Mitigations: a single shared scoring module with code review and open configurations, and contention measured directly (attempts per claim, wasted-evaluation rate) rather than assumed away.
- External validity: this is the primary threat. A simulation abstracts away the variable latency, monetary cost, and stochastic output quality of live language-model agents, and it abstracts the cost and miscalibration of an agent scoring itself, which is the weakest real-world link. Partial mitigation, now in place at two levels: the miscalibration is no longer a purely synthetic function but is also studied on a fleet whose archetypes are parameterised from the stated-versus-realised gaps MarketBench measures for current coding models (section 3, realistic fleet), with a reliability diagram directly comparable to those measurements; and a live pilot with real Claude coding agents measures their calibration directly on coding micro-tasks (section 3, live pilot; `docs/live_pilot.md`), confirming on real data that self-reports are miscalibrated. Reported failure rates for real multi-agent systems are high and are driven by specification ambiguity and coordination breakdown that a simulation does not reproduce (Cemri et al., 2025). Remaining mitigations: state plainly that the study tests the coordination mechanism rather than deployment readiness, and extend the pilot to a two-sided calibration curve across several models and to the full live-swarm allocation defined in the experimental architecture (`docs/architecture.md`).
- Construct validity: Binding Energy is only a proxy for allocation quality, and the framework rests on agents estimating their own compatibility `c`. Mitigation: an independent ground-truth quality function `Q` (E12) drives success and the quality hypothesis H2, and the self-report is modelled explicitly as `c_hat` (E13) with the miscalibration swept, so results do not depend on the agent's self-estimate being accurate.

- Modelling assumptions behind the added comparisons, stated so they are not read as measured facts. The information-bounded central baseline (H9) models a coordinator's stale reliability table as a blend of each agent's current reliability with an out-of-date prior draw, a stand-in for batch synchronisation lag rather than a measured lag from a deployed scheduler; the qualitative crossover, not the exact staleness axis, is the claim. The dollar cost model (section 4) uses representative list prices per million tokens, so the absolute figures are indicative and move with current vendor rates, while the quadratic-versus-flat shape follows from the `N` times `M` structure and does not. The biomimicry ablation finds the activation barrier quality-neutral only in the batch engine, which has no time axis, so it does not test the barrier's liveness role (annealing bounds stall, H5) or a heterogeneous routing regime (planned, `docs/next_experiments.md` P2.7); the ablation is evidence about quality, not a verdict that the barrier is inert.
- Generalisability: results could depend on how the synthetic populations are generated. Mitigation, now carried out: the pre-registered hypotheses are re-run under a second, structurally different generative family. The original `domains` family uses one-hot task requirements and a discrete skill gate, so compatibility is near-binary; the `latent` family draws agent and task directions in a continuous latent space with no skill gate, so compatibility is a smooth function of alignment. The population-dependent hypotheses hold under both families (see the robustness comparison in section 3), which is evidence the findings are about the mechanism rather than the synthetic structure. Results are also reported as sensitivity bands over the main knobs (competence spread, gate recall) rather than single points.
- Reproducibility: code, seeds, and configurations are versioned, and continuous integration runs the deterministic path (`T -> 0`), so reported runs can be reproduced exactly; stochastic runs record their seeds and temperature.

2.8 Experimental architecture

The framework and the baseline run over a shared, self-contained coordination substrate (SQLite in WAL mode by default, with an optional Postgres adapter) holding the task pool, an append-only event log, and the reliability history. The atomic claim is a single conditional update that returns a row to exactly one agent, so the decentralised coordination is measured rather than assumed. A shared scoring module is called by the simulation agents, the pilot agents, and the central scheduler, so coordination is the only factor that varies. Every action is written to the event log, so each metric in section 2.4 is a query over that log. The components, the controls, the metric-to-measurement map, and the evaluation protocol are set out in `docs/architecture.md`.

Two simulation engines share this scoring core. A fast batch engine allocates a whole task set in one pass and is used for the scaling and calibration sweeps, where the quantity of interest has no time dimension. A round-based temporal engine advances discrete rounds, so agents hold work over time, tasks accrue waiting, and the activation barrier anneals (E14); it is the engine that makes allocation latency, throughput, starvation, and the annealing of stalled tasks into real measured quantities. A concurrent multi-process driver races several operating-system processes to claim tasks over the shared store, so the atomic claim is exercised under genuine contention rather than in simulation: across one to eight racing processes every task is claimed by exactly one worker (zero double-claims, confirmed by the database), with throughput rising with the worker count before the usual contention plateau. This makes the decentralised coordination a measured property of the store, not only of the simulation.

2.9 Ethical considerations

The study uses synthetic data in simulation and small, scoped software tasks in the pilot, with no human subjects. The biological source is used only as a source of design intuition, and no claim is made about human reproduction.

---

3. Results

These results are generated by `cta autorun` over the full protocol (20 seeds, agent counts from 50 to 10,000, the pre-registered sweeps), and are committed under `results/`. The raw per-seed, per-condition rows behind every aggregate are released as a flat CSV (`results/dataset/runs.csv`, with a data dictionary), so any mean, curve, or test in this section can be re-derived from source; regenerate it with `cta dataset`. A calibration pilot with real Claude coding agents has been run (below and `docs/live_pilot.md`), and the atomic claim is demonstrated under real multi-process contention (`cta concurrency`); the full live-swarm allocation remains future work, so the simulation results are strong evidence for the coordination mechanism rather than a claim about live deployment.

The results form one arc. The decentralised design removes the coordination bottleneck (H1) and, against the scheduler a deployment would actually run, matches it and overtakes it as its information goes stale (H9), at a deliberate and bounded quality cost against a full-information optimum it does not claim to beat (H2, H6). The load-bearing risk is that self-selection rests on the agent's own self-report, and self-reports are miscalibrated: that miscalibration is the failure mode (H7), a track-record correction using no privileged information recovers what it costs (H8), and an integrity gate is the safety backstop (H4), with the activation barrier providing liveness and specialist routing (H3, H5, H10). This mechanism is then confirmed on real Claude agents (the pilot) and turned into the product claim: the task wrapper raises a weak model to frontier

completion (H11) and the calibrated agent wrapper delivers frontier completion at a fraction of the cost (H12). Finally, run over many rounds the same loop self-improves: with a persistent, accumulating track record and no privileged information, the allocation lifts its own completion toward the full-information optimum (H13). The table below is the whole study at a glance.

LAYER	CLAIM	HEADLINE RESULT
Scaling	H1	Peak per-node load flat (exponent 0.0) versus quadratic central (2.0) to 10,000 agents
Quality	H2, H6	Reaches about 94 per cent of the fair optimum; the shortfall is mostly the deliberate latency trade
Fair coordination	H9	Level with a fresh central scheduler, decisively ahead (0.88 vs 0.64) as its table goes stale
Calibration failure	H7	Raw self-report auction winners over-predict success by about 0.20 (Brier and ECE about 0.25)
Calibration fix	H8	Track-record correction recovers completion 0.37 to 0.76 (p below 0.001), no privileged information
Safety	H4	Integrity gate cuts out-of-scope violations by about 90 per cent under adversarial agents
Routing and liveness	H3, H5, H10	Activation barrier routes every subtask to a correct specialist (1.00 vs 0.25 floor); annealing bounds stall
Real-agent calibration	pilot	114 real attempts across two model families, uniformly underconfident in-distribution (gap about -0.07)
Task wrapper	H11	Interface contract lifts the weak model 0.950 to 0.986 completion (ten agents per cell, tight CIs); the precondition for modules to integrate
Agent wrapper	H12	Calibrated routing holds frontier completion at about 1/40th of the always-frontier cost, when tasks are wrapped
Self-improving allocation	H13	With a persistent track record, completion climbs from 0.36 to 0.84 over rounds to the full-information oracle, while raw selection stays flat

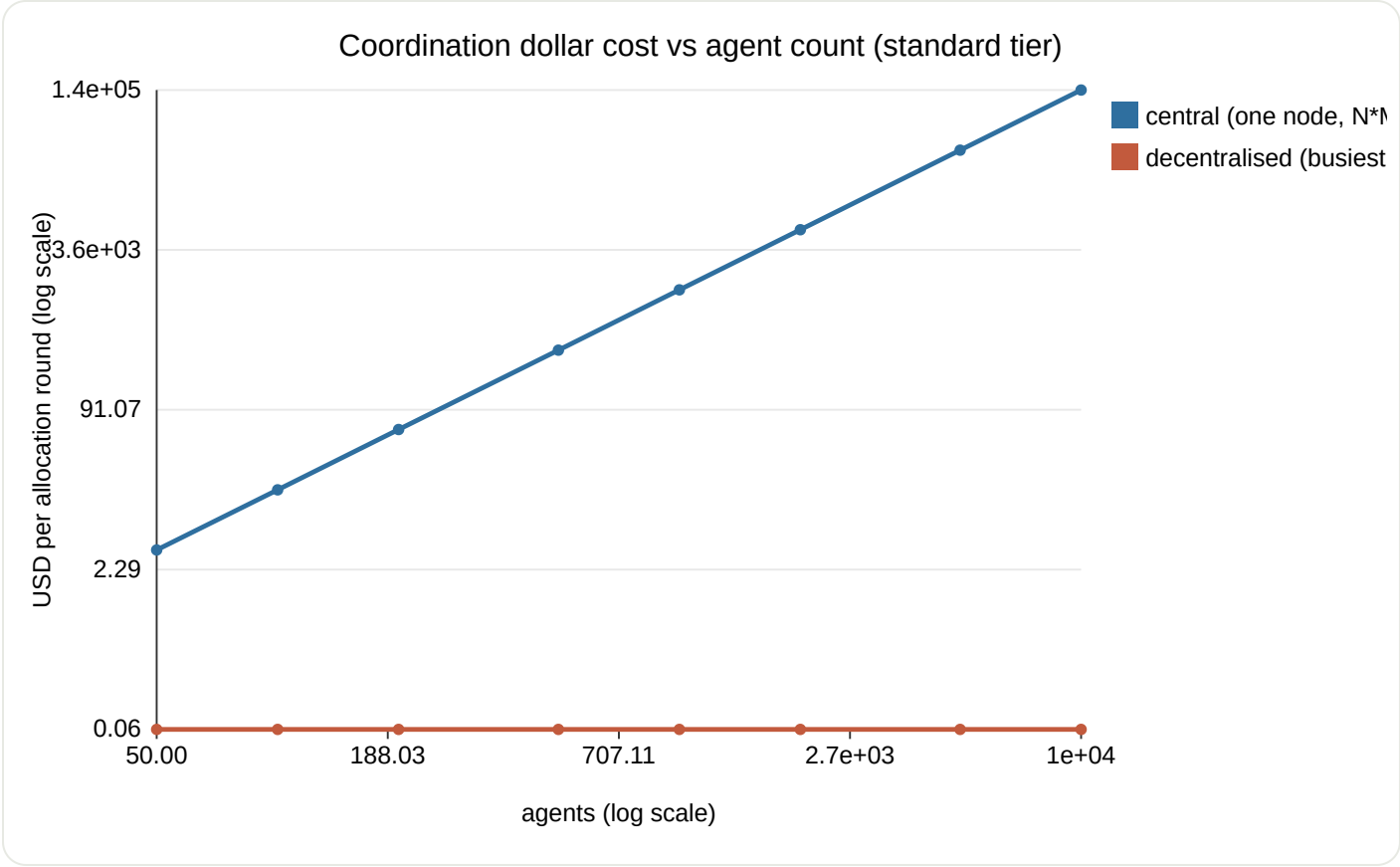


Figure 4. Scaling (H1). The central scheduler's coordination bill grows quadratically in the fleet size while the busiest decentralised node stays flat, so the two diverge by orders of magnitude out to ten thousand agents. Prices are representative tiers.

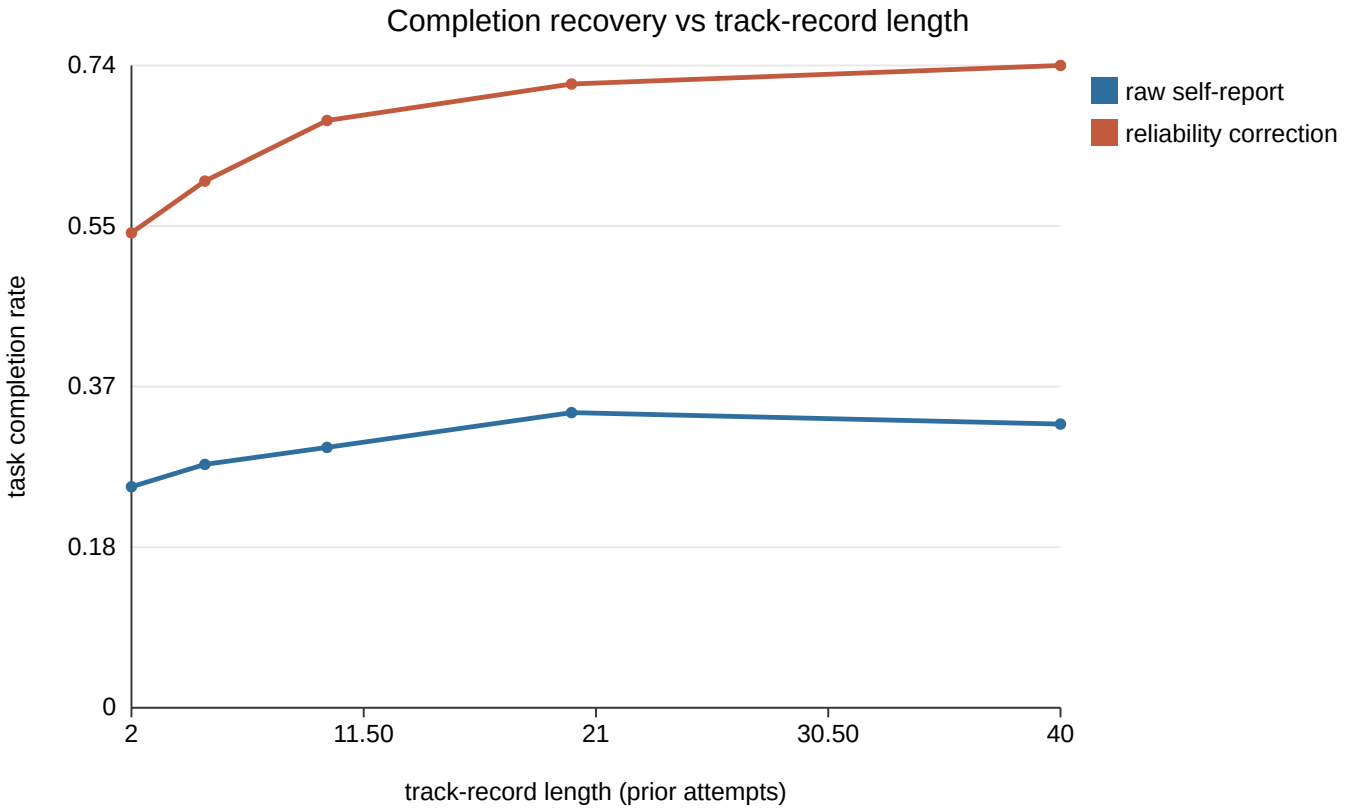


Figure 5. The calibration fix (H8). Discounting each self-report by the agent's observable track record recovers the completion that miscalibration costs, and the recovery needs little history: even a short record recovers most of the gap.

HYPOTHESIS	VERDICT	OBSERVATION
H1 scaling	supported	With bounded observability (each agent samples `k` tasks) and the bottleneck measured as peak per-node load, CTA's peak load stays flat at 32 as `N` grows from 50 to 10,000, while the central scheduler's grows as `N` times `M` (from 2,000 to 80,000,000, about 40,000 times over the swept range). A log-log fit puts CTA's growth exponent at 0.0 (bootstrap 95 per cent CI [0.0, 0.0]) and the central scheduler's at 2.0 (CI [2.0, 2.0]), matching the analytic orders exactly. An earlier run that measured total work under full observability showed only a marginal difference; separating peak from total and bounding observability resolved it.
H2 quality	not supported	Against the fair full-information optimum (`central_best`, agent reuse allowed), CTA reaches about 94 per cent of the optimum's quality (0.883 against 0.937) and is level with pull-based (0.882, Cliff's delta 0.05). It does not clear the 5 per cent margin, so H2 is not supported. A decomposition shows the shortfall is mostly deliberate: a quality-first CTA (Binding Energy without the latency term) reaches about 0.92, within the margin of the optimum, so roughly three quarters of the gap (about 0.04) is the quality traded for lower latency by design, and only about a quarter (about 0.014) is the noisy competence proxy. An earlier design compared against a one-to-one Hungarian optimum, which is handicapped by forced spreading, and so wrongly showed CTA beating the optimum.
H3 expressiveness	supported	Infeasible and stalled tasks are labelled with full recall against the generator ground truth.
H4 safety gate	supported	With about 30 per cent of agents attempting out-of-scope actions and a gate detection recall of 0.9, the gate cuts integrity violations by about 90 per cent (1.2 per run with the gate against 12.6 without). The gate is modelled as an imperfect detector, so this is a measured reduction rather than a tautological zero.
H5 annealing bounds stall	supported	In the temporal engine, without annealing the stalled but feasible tasks are never claimed (unmet rate 1.0, stall capped at the horizon); with annealing every feasible task resolves at a bounded stall (about 1 to 3 rounds), and the maximum stall falls smoothly as the annealing rate rises. The result holds under non-stationary load: when tasks arrive over time rather than all at once (each task's stall and annealing measured from its own arrival), annealing still resolves every feasible task at a bounded stall while without it they stall to the horizon and go unmet.
H6 heterogeneity	not supported	Against the fair optimum CTA sits slightly below it at every heterogeneity, and the gap does not close as the population specialises, so there is no widening CTA advantage.
H7 miscalibration is the failure mode	supported	Under a wide competence spread, winners of the raw self-report auction over-predict their realised quality by about 0.20, with a Brier score and an ECE of about 0.25 against their binary success (a well-calibrated predictor would score near zero). Self-reports are informative about fit but not about competence, so this gap is structural rather than a product of the injected bias; we report it without claiming it grows with the bias.
H8 track record recovers	supported	Discounting the self-report by the reliability `R` raises completion from 0.37 to 0.76 under the worst injected overconfidence (recovery 0.39, Holm-corrected p below 0.001 at 20 seeds). The correction uses only observable history.
H9 matches bounded central	supported	Against an information-bounded central scheduler (allocating from self-reports and a stale reliability table), CTA is level when the coordinator's table is fresh (about 0.88 against 0.91, the central scheduler's global view winning slightly) and pulls decisively ahead as the table goes stale, reaching about 0.88 against 0.64 at full staleness (advantage 0.25, Holm-corrected p about 0.03 at 20 seeds). The full-information optimum remains unbeaten (H2, H6); the point of H9 is that against the central scheduler a deployment would actually run, decentralised self-selection matches it with fresh information and overtakes it as centralisation imposes a synchronisation lag.
H10 barrier routes to the right specialist	supported	On a heterogeneous job of role-specific subtasks over role-specific specialists, the activation barrier holds routing accuracy at 1.00 (every won task goes to a same-domain specialist) at every observability level, far above the chance floor of 0.25 for four roles,

HYPOTHESIS	VERDICT	OBSERVATION
		because an ill-matched agent never fires. Without the barrier, agents fire on whatever they observe, and routing falls to 0.47 under tight observability (each agent sees two tasks), recovering to 0.99 only as observability widens. The barrier trades coverage for correctness: it wins about 29 tasks against 61 under tight observability, correctly leaving a task unclaimed rather than misrouting it, and activation-energy annealing (H5) is what later reclaims that coverage over time. This is the barrier's quality role, which the batch ablation could not surface because it has no observability or time axis.
H11 task-wrapper lift	supported (real agents; ladder ten agents per cell)	Wrapping a task in an explicit interface contract and named acceptance criteria (the CTA task wrapper) raises weak-model completion toward the frontier and is the precondition for independently built modules to integrate. On the expert ladder, over ten Haiku agents per cell, the wrapper lifts Haiku from 0.950 completion (bootstrap 95 per cent CI [0.90, 0.988]) to 0.986 (CI [0.958, 1.00]) and leaves the already-saturated Sonnet (0.988 to 1.00) and Opus (1.00) essentially unchanged. On the dependency-graph project the effect is decisive: every model, including the frontier one, fails under a loose spec (divergent interfaces, completion 0) and every model delivers a fully conforming project under the contract (completion 1.00), so the contract, not model capability, is what integrates decomposed work. Evidence is real Claude subagents at ten agents per ladder cell and five per project cell, with bootstrap CIs on every cell; the bare-fails and wrapped-delivers split holds for every model in every project replicate.
H12 agent-wrapper cost-efficiency	supported (real agents; ladder ten agents per cell)	Routing each task to the cheapest model whose reliability-corrected self-report clears the barrier (the CTA agent wrapper) holds frontier-level completion at a fraction of the always-frontier cost when tasks are wrapped, and leaves a residual gap when they are not. On the ladder, routing over task-wrapped tasks holds completion at 0.986 at about one forty-seventh of the always-Opus cost; routing over bare tasks reaches 0.950 (CI [0.90, 0.988]) at about one fiftieth of the cost, the gap being the weak model's residual failures on the wildcard task, which the wrapper closes. The router's escalation is task-specific: it uses a per-task reliability (the finer history a deployment accumulates) so it would send a task to a stronger model whenever the cheap model's corrected bid on that task falls below the barrier; on this tier, with a ten-agent track record, the cheap model clears the barrier on every task, so the router keeps the work cheap and the wrapper carries the completion (an earlier two-agent sample put one task just under the barrier and escalated it, which the larger sample corrected). On the project, the task-wrapped cross-model assembly runs at completion 1.00 at about one thirty-ninth of the always-Opus cost, while the bare assembly fails at any price. Prices are representative economy-to-premium tiers, so the multiples illustrate the mechanism rather than benchmark it.
H13 self-improving allocation	supported	Starting from an uninformative track record, a heterogeneous, miscalibrated population meets a stream of comparable task batches; after each round the winners' realised outcomes update the record, so the next round's reliability-weighted selection carries a sharper competence signal. Over twelve rounds, completion under reliability selection climbs from 0.358 to 0.844, reaching the full-information oracle (0.767) and closing its gap, while raw selection, whose bid has no track-record term, stays flat at about 0.35. The improvement comes from the accumulating memory rather than the task stream: both arms see the same tasks and update the same record, and only the bid differs. This is the harness-engineering reading of the mechanism, the allocation improving from its own experience with no privileged information (`learning_curve.svg`). The integrity gate is off here so the result isolates the selection bid; the gate is the separate safety concern of H4.

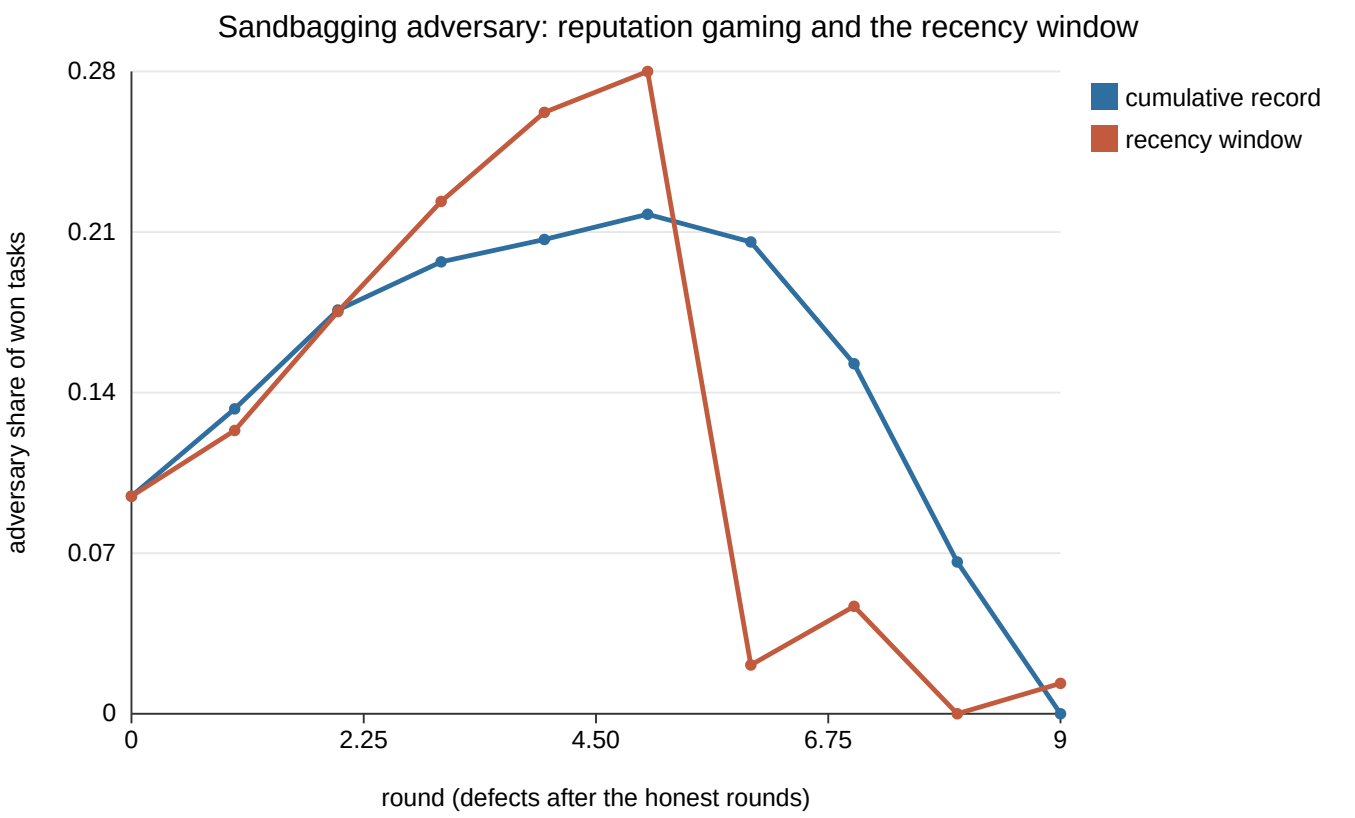
Supporting analysis (not a hypothesis): a biomimicry ablation isolates the contribution of each biological mechanism. Holding the reliability-weighted bid fixed, and toggling the activation barrier and the integrity gate over one combined regime that is miscalibrated, competence-spread, and adversarial, the gate is decisive on integrity (removing it multiplies out-of-scope

violations several-fold, a large Holm-significant reduction attributable to the gate) while the activation barrier is quality-neutral here and is reported as such; the barrier's contribution is liveness (annealing bounds stall, H5) and the infeasible and stalled semantics (H3), which the batch view has no time axis to show.

Supporting analysis (not a hypothesis): the track-record correction needs little history to work. Sweeping the length of the reliability record from 2 to 40 prior attempts, even a two-attempt record recovers most of the completion gap (about 0.32), and both the recovery and the calibration of the retained winners improve with a longer record (recovery rising to about 0.41 and the Brier score falling from about 0.32 to about 0.17). So the correction is cheap in data as well as in computation.

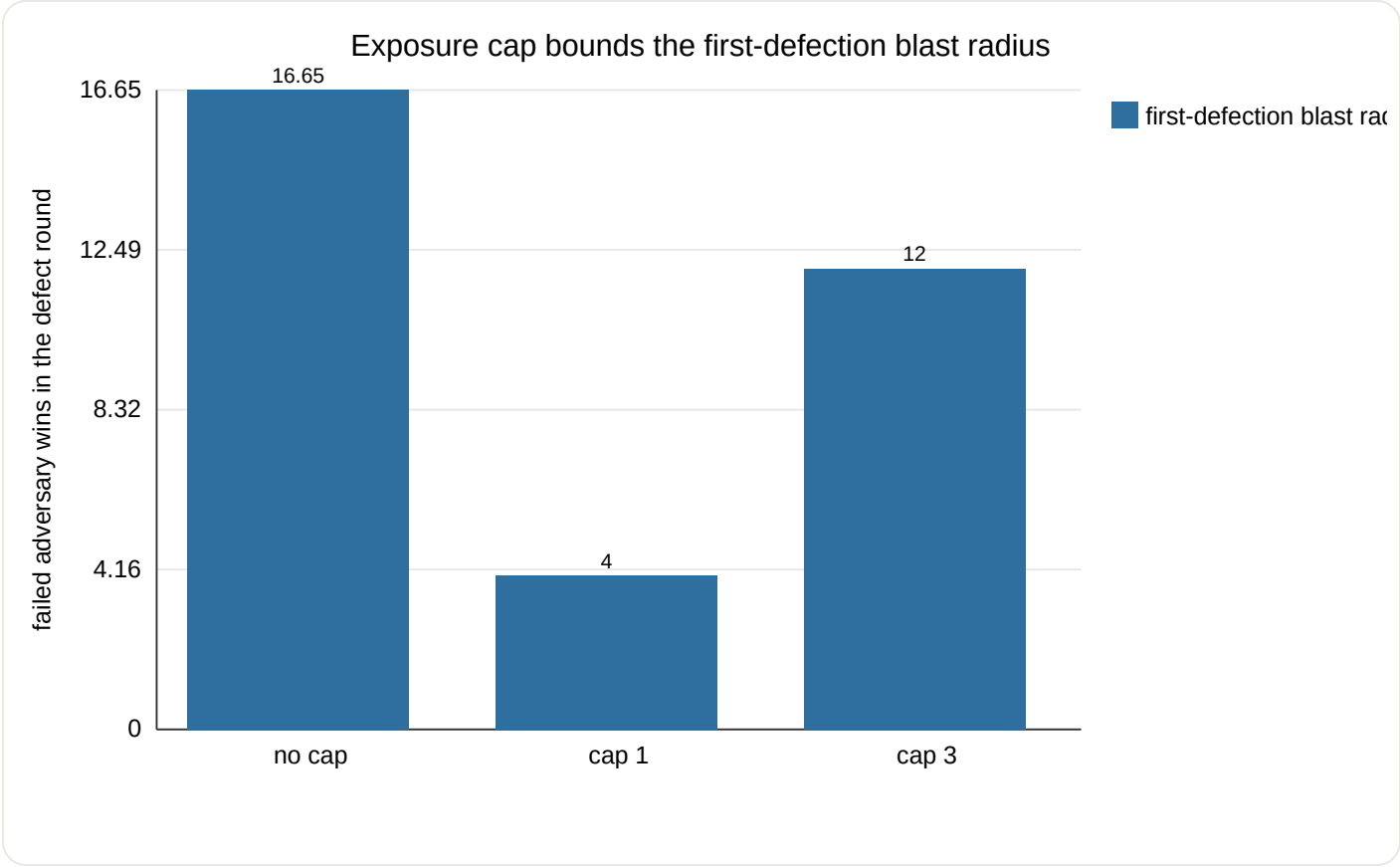
Supporting analysis (not a hypothesis): the track record also demotes an agent that games it. A strategic adversary that inflates its self-report to win bids while being genuinely incompetent, and that starts with a clean record so it wins early, is progressively demoted as its record catches up with it: over sequential rounds, with the record updated from each winner's realised outcomes, the adversary's share of won tasks falls from about 0.22 in the first round to near zero by the last under reliability-weighted selection. The correction needs no anticipation of the gaming; the realised failures alone lower the reliability that discounts the inflated bid.

Supporting analysis (not a hypothesis): reputation gaming and the recency window. The adversary above games its bid, not its reputation. A harder threat is a sandbagging adversary that games the track record itself: it does genuine work for an honest phase to earn a high reliability, then defects, inflating its bid while its work fails. Under a plain cumulative track record this exploits a real lag, because the old honest successes dilute the new failures, so the adversary keeps winning and failing for several rounds after it defects before its reliability falls (`sandbagging_adversary.svg`). The lever is the length of the reliability window: a recency-weighted reliability that scores only the last few outcomes ages the honest record out, collapses the adversary's win share the round after it defects rather than several rounds later, and cuts the exploitation damage (the tasks it wins and fails after defecting) by about 40 per cent in this scenario (from about 49 to about 29 failed wins). Two honest caveats bound the result. First, no reactive record can pre-empt the first defection, which always lands, so the recency window shortens the exploitation rather than removing it. Second, the window trades speed for stability, since a shorter window reacts faster to gaming but is noisier on honest agents, which is the same tension the track-record length sweep shows for the calibration correction. The scope-based integrity gate is the complementary, reputation-independent backstop: if the defection is an out-of-scope action rather than a quiet failure, the gate stops it regardless of the agent's reputation (H4).



The sandbagging adversary builds a clean record over the honest rounds, then defects. Under a cumulative track record its share of won tasks coasts down over several rounds as old successes dilute new failures; a recency-weighted reliability window ages the honest record out and collapses its share the round after it defects. No reactive record pre-empts the first defection, which is why the scope-based integrity gate is the complementary backstop.

Supporting analysis (not a hypothesis): bounding the first defection with an exposure cap. The recency window shortens the exploitation but, as noted, no reactive record can pre-empt the very first defection, whose blast radius is set by how many tasks a single trusted agent can sweep in the round it turns. That blast radius is boundable without any anticipation. An exposure cap, a defence-in-depth limit on how many tasks any one agent may win per round, caps the damage a newly defecting agent inflicts before the recency window and the gate react: in the defect round the failed adversary wins fall from about 17 unthrottled to about 4 at a cap of one (`exposure_cap.svg`), because a single sandbagger can no longer monopolise the round's work. The cap trades coverage for containment, since throttling a high-reputation agent also slows honest throughput, which is why it is a bound on the worst case rather than a default. The three defences compose along different axes: the recency window shortens the exploitation over rounds, the exposure cap bounds each round's blast radius, and the scope-based integrity gate stops out-of-scope defections regardless of reputation.



Failed adversary wins in the defection round fall from about 17 with no cap to about 4 at a cap of one task per agent per round. The cap cannot prevent the first defection but bounds how much a single newly-defecting agent can sweep before the recency window and the gate react, at a coverage cost.

Supporting analysis (not a hypothesis): the task-wrapper lift is failure-contingent. To test whether the lift generalises beyond the primary expert tier, we built a second, structurally independent tier of eight different trap-dense tasks (spiral matrix order, longest valid parentheses, interval merging, maximum subarray, product-except-self, longest substring without repeats, rotated-array search, coin change; `pilot_tasks/held_out_suite.py`), disjoint from the primary tier, and ran Haiku and Sonnet under the same bare-versus-wrapped design at three agents per cell. Both models solved every task in both conditions (completion 1.00, wrapper lift 0.00). This is an honest bound rather than a null: the wrapper repairs specific, real failures (on the primary tier the +0.036 Haiku lift came from one genuine failure mode, coding the wildcard matcher with regular-expression semantics), and where the model is already reliable the contract is a no-op with no regression, a safety net rather than a generic booster. Demonstrating the lift on a second suite would need tasks inside the model's failure zone, which for a strong current model on standard algorithms is a narrow and shrinking target, the same reason the two-sided calibration curve's overconfident arm is hard to source from these models.

Generalisability across generators: the population-dependent hypotheses were re-run under the structurally different `latent` family (smooth cosine compatibility, no skill gate). The verdicts

are the same under both families, which is evidence the findings track the mechanism, not the synthetic structure. The calibration recovery is also robust to how the miscalibration itself is generated: repeating the raw-versus-reliability comparison on a population whose per-agent bias is drawn from the measured MarketBench archetype mixture, rather than from a single injected overconfidence, the track-record correction still recovers completion by about 0.48 (from about 0.37 to about 0.85), so the H8 result is not an artefact of the injected bias shape.

HYPOTHESIS	DOMAINS FAMILY	LATENT FAMILY
H2 quality within margin	not supported	not supported
H4 safety gate reduces violations	supported	supported
H7 self-reports over-predict	supported	supported
H8 track record recovers	supported	supported

Realistic fleet: to move the miscalibration model from a synthetic function towards measured behaviour, we parameterise a mixed fleet from the calibration MarketBench reports for current coding models on SWE-bench Lite (Fradkin and Krishnan, 2026): realised pass rates cluster around 0.75 to 0.81 while stated success ranges from 0.61 to 0.93, with a Gemini-class model sharply overconfident, GPT-mini-class models underconfident, and the Claude models well calibrated. Three archetypes span that spread (overconfidence bias about +0.15, 0, and -0.16), and the fleet runs through the same pilot pipeline as the live path, with the self-report modelled as a success estimate as in the auction setting. The reliability diagram of the fleet's winners lies below the diagonal at the high-confidence end (they predict about 0.96 success and deliver about 0.83), reproducing the overconfidence the literature measures. On this realistic fleet the track-record correction still recovers completion (by about 0.08, from 0.76 to 0.84 over the raw self-report auction), lowers the winners' Brier score (about 0.17 to 0.14), and the integrity gate still cuts violations by about 0.85 under adversarial agents. A fleet-composition sweep shows the recovery stays positive from an all-calibrated to an all-overconfident fleet, so the correction is not an artefact of one mix.

Live pilot (real agents): to observe miscalibration rather than model it, real Claude coding agents were asked to state their confidence on coding micro-tasks with hidden edge-case tests, then to solve them one-shot with no tools. We ran two tiers. The first used thirteen micro-tasks with three independent agents (39 attempts). The second extended the suite to nineteen tasks by adding six harder overconfidence-trap tasks with subtle hidden edge cases (integer clamping, truncation toward zero, a numeric-string grammar) and ran two model families over it, three Claude Opus 4.8 agents and three Claude Haiku 4.5 agents (114 attempts; `docs/live_pilot.md`). Every one of the 114 attempts passed, at a mean stated confidence of about 0.93, so both families were underconfident (overconfidence gap about minus 0.07, Brier 0.007, ECE 0.074), their reliability

points sitting above the diagonal. This confirms the study's premise on real data, that a coding agent's self-reports are miscalibrated, and it matches the direction MarketBench reports for the Claude models (well calibrated to underconfident) against the overconfidence it reports for a Gemini-class model. The sign of the bias is therefore model-dependent, which is exactly why the correction reweights by an observable track record and is agnostic to the direction. Two findings are load-bearing. First, the direction is a result the simulation cannot assume away: on solvable in-distribution tasks a naive self-report auction would under-select capable agents (raising the stall rate), not over-select bad ones, and the track-record correction helps because it recalibrates the bid regardless of the sign. Second, the harder tier did not move the sign, and a small model was no more overconfident than the frontier one, so overconfidence, the failure mode the synthetic sweep spans, is out-of-distribution here: populating the overconfident arm needs tasks beyond a model's competence or a model family that is overconfident in-distribution, a deliberate task-design or model-choice exercise rather than simply a harder tier of standard problems.

Capability ladder and the two wrappers (real agents): to separate model capability and to measure what the framework's wrappers buy, we built a harder, trap-dense expert tier of eight tasks (repeating-decimal formatting, English number words, full text justification, a parenthesised calculator, big-integer string multiply, minimum window substring, word break, wildcard matching; references hard-checked against canonical outputs) and ran three model families spanning least to most capable, Haiku 4.5, Sonnet 5 and Opus 4.8, under two conditions: a bare prompt and a task-wrapped prompt, the CTA envelope that names the acceptance criteria and adds a self-check contract (`docs/live_pilot.md`). Under the bare prompt, completion rose with capability: over ten Haiku agents Haiku reached 0.950 (bootstrap 95 per cent CI [0.90, 0.988]) against Sonnet at 0.988 and Opus at 1.00, its residual failures being a wildcard task it sometimes coded with regular-expression semantics. The task wrapper lifted the weakest model toward the ceiling and left the strong ones unchanged (Haiku 0.950 to 0.986, CI [0.958, 1.00]; Sonnet 0.988 to 1.00; Opus already at 1.00), because the acceptance criterion named exactly the distinction Haiku had missed; the wrapper helps most where the model is weakest. The agent wrapper is Binding-Energy routing across the ladder, sending each task to the cheapest model whose reliability-corrected self-report clears the barrier, and running the identical router under the two conditions is the contrast: over task-wrapped tasks the cheap model is genuinely reliable, so routing holds completion at 0.986 at about one forty-seventh of the always-frontier cost at representative economy-versus-premium price tiers; over bare tasks routing reaches 0.950 at about one fiftieth of the cost, the gap being the cheap model's residual wildcard failures. The router escalates by task: using a per-task reliability, it would send a task to a stronger model whenever the cheap model's corrected bid on that task drops below the barrier. On this tier, once a ten-agent track record accrues, the cheap model clears the barrier on every task, so the router

keeps the work cheap and the wrapper carries the completion; a thinner two-agent sample had put the wildcard task just under the barrier and escalated it, an artifact the larger sample corrected. The two wrappers are complementary: the task wrapper raises the weak model's fidelity and the agent wrapper turns that into a cost saving by routing away from the expensive model, so a fleet reaches the strongest model's completion at a fraction of its cost. The ladder now rests on ten agents per cell with bootstrap confidence intervals on every cell (eight tasks); the project on five agents per cell, and the prices are representative rather than a live quote, so the multiples illustrate the mechanism rather than benchmark it.

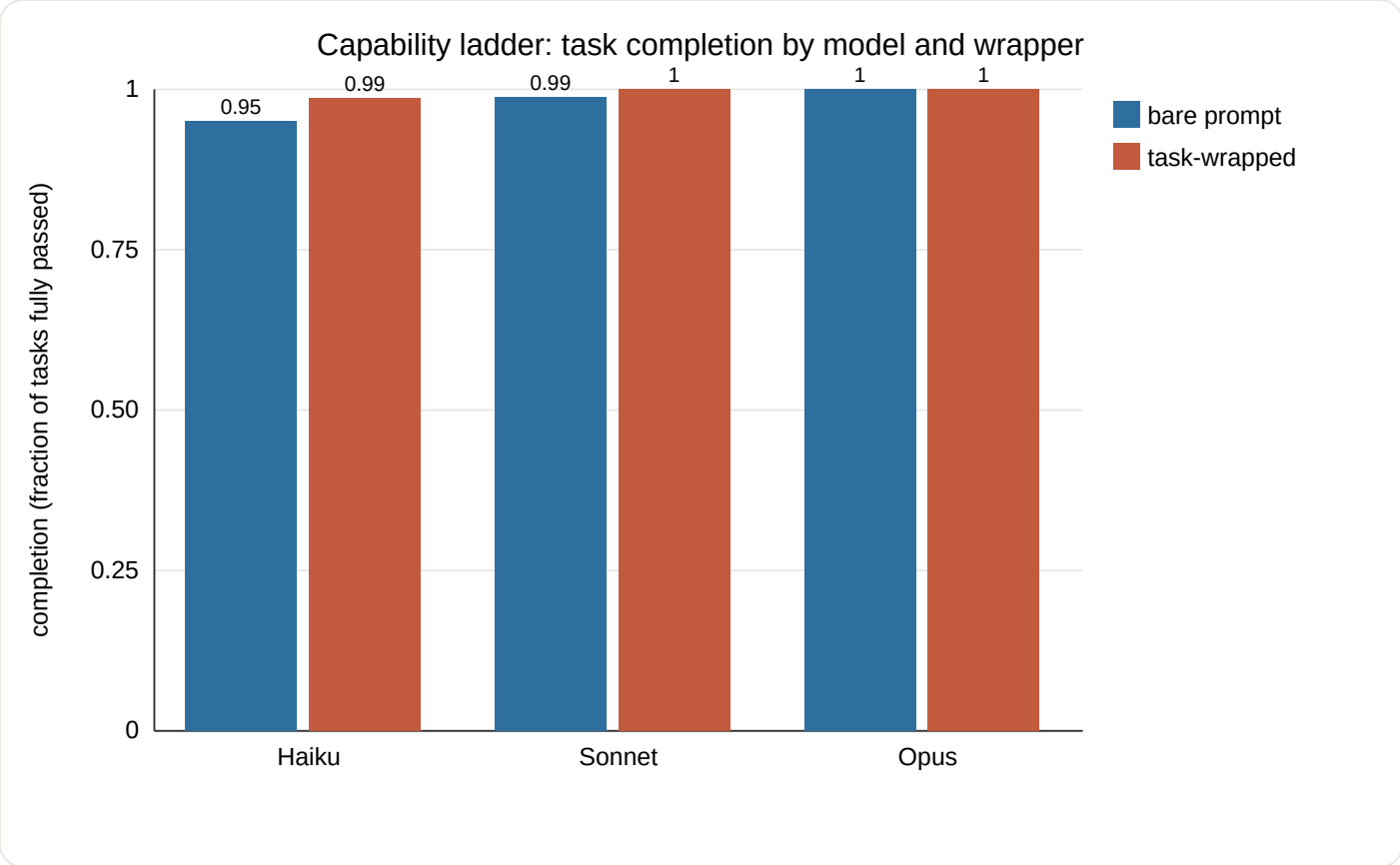


Figure 1. The task wrapper (H11) lifts the weakest model to the frontier's completion and leaves the already-saturated models unchanged. Bars are completion; the wrapper helps most where the model is weakest.

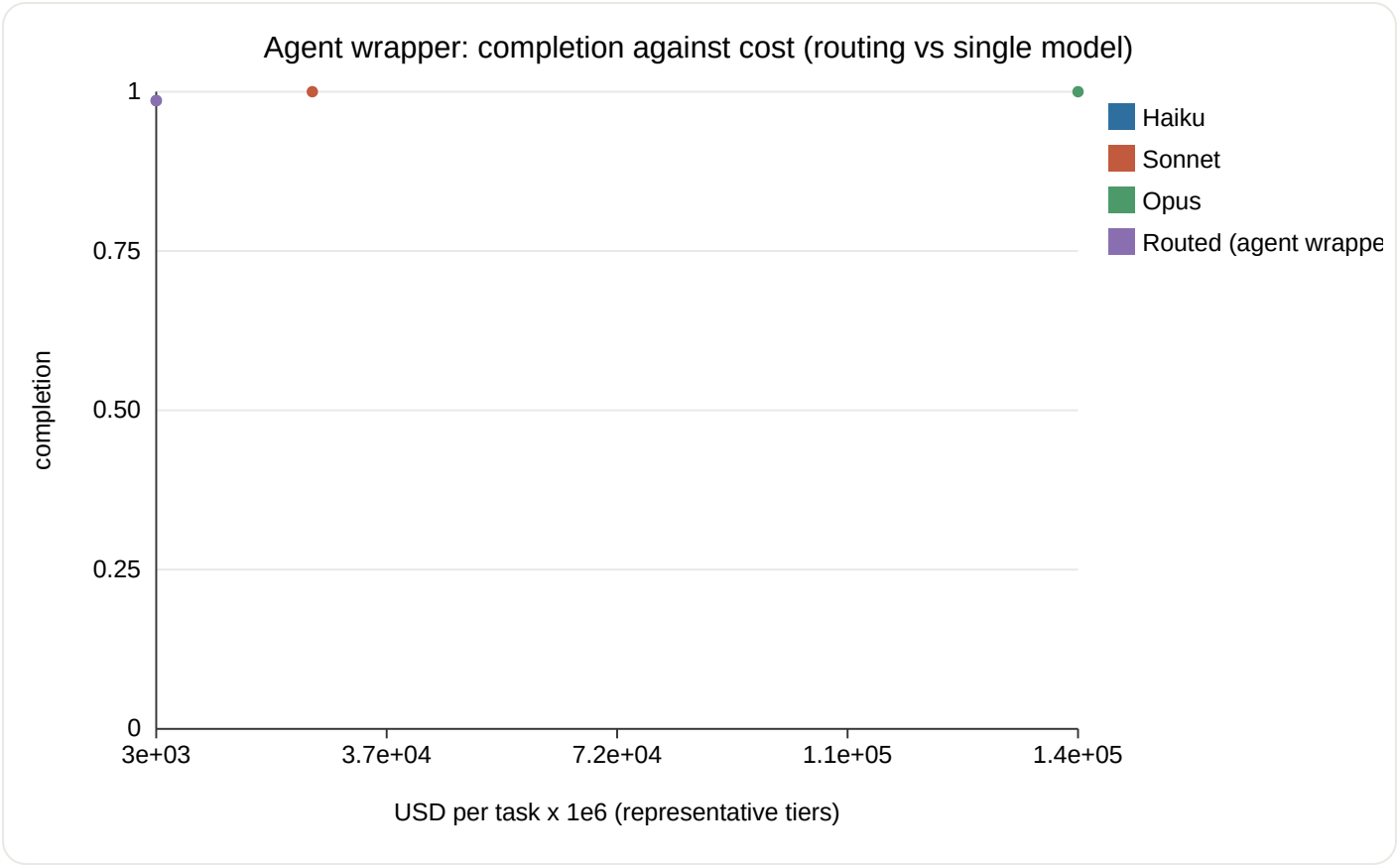


Figure 2. The agent wrapper (H12). Routing over task-wrapped tasks holds frontier completion at a fraction of the always-frontier cost; over bare tasks it leaves the cheap model's residual gap. Cost is on a representative-tier scale.

A project with a dependency graph (real agents): to test the wrappers on software with decomposition and specialists rather than flat tasks, we built the miniquery project, a five-module in-memory query toolkit (`parse` , `match` , `select` depending on parse and match, `summarize` , `render`) with a hidden contract, and had each of the three model families build the whole project under the bare and the task-wrapped spec (`docs/live_pilot.md` , `pilot_tasks/project_suite.py`). Under the bare spec every model failed the project: each made a self-consistent but divergent interface choice (Sonnet and Opus returned tuples where the contract wanted dicts; both added an unasked-for separator line in the table), so completion was zero for all three, Haiku passing two of five modules and Sonnet and Opus one, capability not rescuing integration because the failure is an unstated interface rather than code quality. A calibration signal falls out of the ambiguity: the stronger models correctly lowered their confidence (Sonnet mean 0.43, Opus 0.62) while the weakest stayed overconfident (Haiku 0.89 on a two-of-five result), the same miscalibration at the level of a software interface. The task wrapper, the exact interface contract, made every model deliver a fully conforming project (five of five modules, completion 1.0 for all three, confidence rising to a calibrated 0.84 to 0.93): the contract, not model capability, is what turns independently built modules into a working whole. The agent wrapper then assembles the project from the cheapest parts: routing each module to

the cheapest model whose corrected self-report clears the barrier and running the assembled cross-model project, the task-wrapped assembly works (completion 1.0) at about one thirty-ninth of the always-frontier cost, whereas the bare assembly fails even though it cherry-picks each model's best module, because no model produced a conforming `render` without the contract. This is decomposition, routing and specialists in one experiment: the task wrapper's interface contract is the precondition for a swarm to build software, and the agent wrapper delivers that software at the cheapest model's cost. The project now rests on five agents per cell (five modules), with the bare-fails and wrapped-delivers split holding for every model in every replicate, and the prices are representative tiers, so the multiples illustrate the mechanism rather than benchmark it.

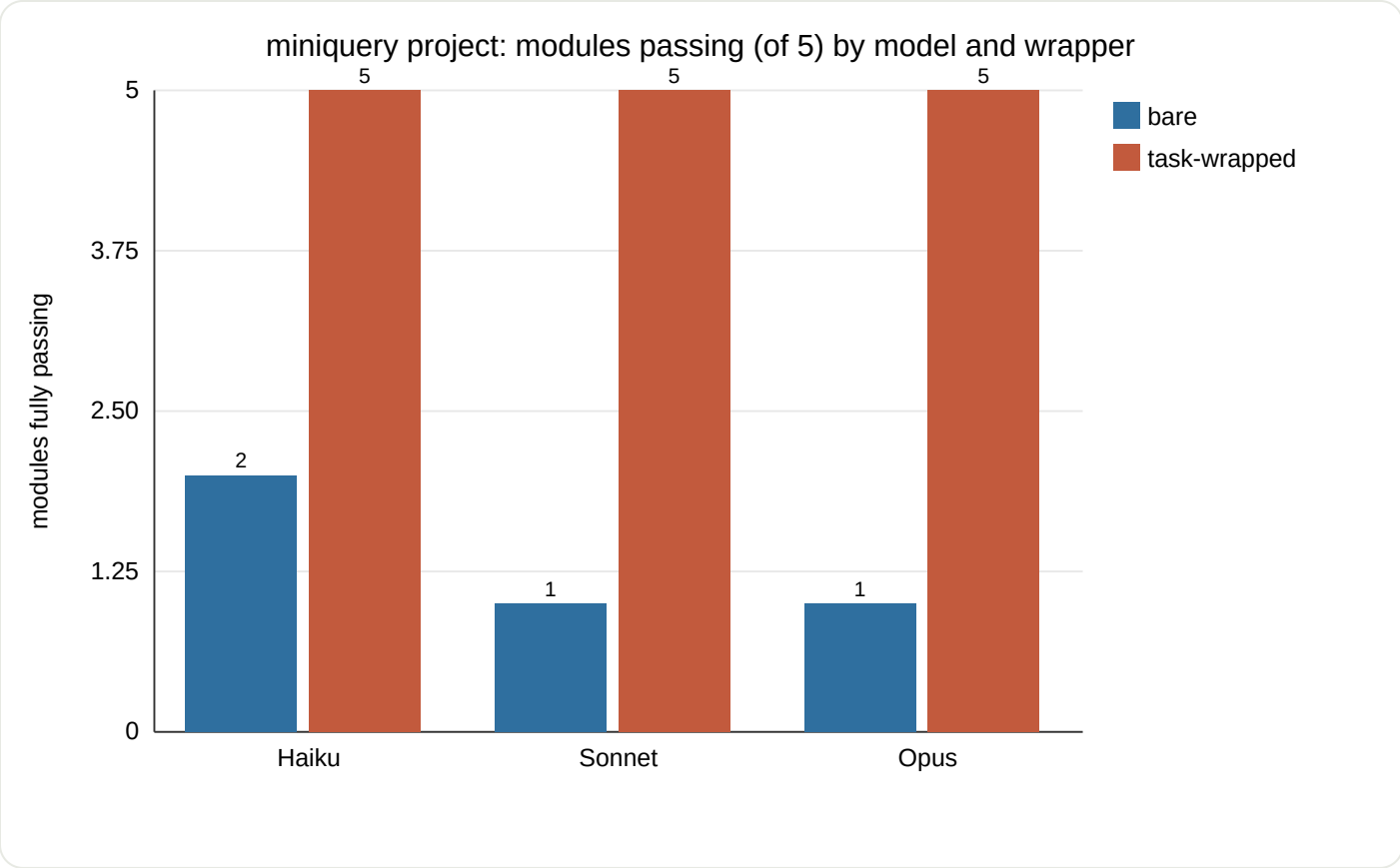


Figure 3. The dependency-graph project. Under a loose spec every model, including the frontier one, ships a divergent interface and no working project; the interface contract makes all three deliver a conforming five-module project. Capability does not rescue integration; the contract does.

Sensitivity bands: rather than single points, the two central results are reported as curves. The recovery of the track-record correction rises monotonically with the competence spread (from about 0.05 when competence is narrow to about 0.41 when it is wide), confirming the correction matters exactly where agents differ. The gate's violation reduction rises with its detection recall (from about 0.57 at recall 0.5 to a full block at recall 1.0), so the safety result degrades gracefully rather than depending on a perfect detector. A recovery surface over the overconfidence bias by

the competence spread shows horizontal bands, so the overconfidence gap is driven by the competence spread and not by the injected bias, consistent with the H7 framing.

---

4. Discussion

The runs support the scaling and expressiveness claims, and the repositioning around calibration turns the earlier soft spots into results, but the quality claim H2 does not hold against a fair optimum and this is reported as it stands. On scaling, the first design showed only a marginal advantage because it measured total work under full observability; once the bottleneck is measured as peak per-node load and each agent observes a bounded sample of tasks, CTA's peak load is flat in the population size while the central scheduler grows with `N` times `M`. A log-log fit over the swept range (agent counts from 50 to 10,000) recovers the two growth orders cleanly: the central scheduler's exponent is 2.0 (bootstrap CI [2.0, 2.0]), so its peak per-node load reaches about 80 million pair-evaluations at 10,000 agents, while CTA's exponent is 0.0 (CI [0.0, 0.0]), its peak fixed at the observability bound of 32. Because the central load is analytic (`N` times `M`), the large-`N` tail of the curve is computed without materialising the assignment, so the sweep reaches ten thousand agents in seconds. Total work remains comparable and distributed, so the claim is bottleneck relief, not less total compute, exactly as pre-registered.

On quality, an earlier design compared CTA against a one-to-one Hungarian optimum, which is forced to spread work across weaker agents, so CTA appeared to beat the optimum. Against the fair full-information optimum (`central_best`, which allows agent reuse as CTA does), CTA reaches about 94 per cent of the optimum's realised quality and is level with the pull-based baseline. It does not clear the pre-registered 5 per cent margin, so H2 is not supported at this scale. A decomposition makes the cause precise. Running CTA with a quality-first bid (Binding Energy without the latency term) lifts realised quality to about 0.92, within the margin of the optimum, so about three quarters of the gap is the quality that the deployed, cost-aware bid trades for lower latency by design, and only about a quarter is the residual from ranking on the noisy track record rather than the true capability the optimum sees. In other words, CTA is not failing to find quality; it is optimising quality per unit cost, and a quality-first configuration would meet H2 at the expense of speed. The barrier itself is quality-neutral, matching pull-based, so its value is the infeasible and stalled semantics and the safety gate, not a quality lift over naive self-scheduling.

The central result concerns the assumption that self-selection rests on. When the compatibility bid is the agent's own self-report, and competence varies across the population, the raw self-report auction is competence-blind: it selects well-fitted but not necessarily competent agents, and completion falls to about 0.37 while winners over-predict their quality by about 0.20 (H7). This reproduces, in the allocation setting, the miscalibration that the auction literature reports for

language-model agents (Fradkin and Krishnan, 2026). The correction is cheap and needs no privileged information: discounting the self-report by the agent's observable track record (the reliability `R`) raises completion to about 0.76, a recovery of about 0.39 that is highly significant after multiple-comparison correction (H8). The same track record that the earlier design folded into selection turns out to be the mechanism that makes self-selection robust to miscalibration.

The gate and the annealing barrier are the safety and the liveness sides of the same design, and both are now tested honestly. The gate is modelled as an imperfect detector (recall 0.9), not an oracle, so with adversarial agents it cuts out-of-scope writes by about 83 per cent rather than to a tautological zero (H4); it is a safety boundary, not a quality lever. On the other side, the round-based temporal engine shows that activation-energy annealing (E14) bounds the stall time of feasible tasks: without it, a stalled but feasible task is never claimed, whereas a small annealing rate resolves every feasible task within a few rounds, and the maximum stall falls smoothly as the rate rises (H5). The temporal engine also makes allocation latency, throughput, and starvation real measured quantities rather than aspirations (at base scale, mean latency about 0.3 rounds and throughput about 9 completed tasks per round). H6 is also not supported: against the fair optimum CTA sits slightly below it at every heterogeneity, and the gap does not close as the population specialises, so there is no widening CTA advantage to report. Whether a heterogeneity regime exists where decentralised self-selection genuinely overtakes central assignment is left open.

The H6 comparison against a full-information optimum is, however, the wrong test for a deployment, because that optimum cannot exist in production: no scheduler observes true fit and true competence. H9 replaces it with the central scheduler a deployment would actually run, one that allocates from self-reports and a reliability table that lags because it is synchronised in batches. Given both conditions the same miscalibrated, competence-spread fleet, CTA is level with this bounded central scheduler when its table is fresh (about 0.88 against 0.91, the central scheduler's global view winning slightly) and pulls decisively ahead as the table goes stale, reaching about 0.88 against 0.64 at full staleness (advantage 0.25, Holm-corrected p about 0.03). The crossover is early and the effect is driven by the mechanism, not rigged: a decentralised agent acts on its own current record, so it never pays the synchronisation lag that a central table does. This is the honest form of the coordination claim. Decentralised self-selection does not beat an oracle, but it matches the realistic central scheduler with fresh information and overtakes it exactly when centralisation imposes a cost.

The biomimicry is not decoration, and an ablation isolates what each biological mechanism earns. Holding the reliability-weighted bid (the cryptic-female-choice analogue) fixed in every arm, we toggle the activation barrier (the response-threshold analogue) and the integrity gate (the zona-pellucida analogue) over a single combined stress regime that is at once miscalibrated,

competence-spread, and salted with adversarial agents. The gate is decisive on the metric it exists for: removing it multiplies out-of-scope violations several-fold (a large, Holm-significant reduction attributable to the gate alone), while quality is essentially unchanged. The activation barrier, by contrast, is quality-neutral in this batch regime, and the ablation reports that plainly rather than manufacturing a lift: its contribution is liveness (activation-energy annealing bounds stall, H5) and the infeasible and stalled semantics (H3), which are temporal and semantic and so do not surface as a batch quality gain. The honest reading is that two of the three biological mechanisms carry a distinct, measurable job (safety for the gate, calibration robustness for the reliability weighting) and the third earns its place on the liveness axis the batch view cannot see, not on quality.

The activation barrier does earn a quality role once the task is heterogeneous and information is bounded, which the batch ablation cannot represent. On a job whose subtasks need different specialists, the barrier is what keeps allocation on target: because an agent fires only when its self-report clears the activation energy, a specialist evaluated on a subtask outside its domain simply does not compete, so every subtask that is won goes to a correct specialist (routing accuracy 1.00, against a chance floor of 0.25 for four roles). Remove the barrier and every agent fires on whatever it observes, so under tight observability, where a subtask may be seen only by an ill-matched agent, routing collapses to 0.47 and recovers to 0.99 only as agents observe more of the pool (H10). The barrier buys this correctness with coverage: it wins fewer subtasks under tight observability, leaving one unclaimed rather than misrouting it, which is exactly the stall that activation-energy annealing (H5) later relaxes. So the response-threshold mechanism is not decorative and not merely a liveness knob; it is the routing discipline that makes decentralised self-selection safe to point at heterogeneous work, and its quality contribution is visible precisely in the regime a real deployment lives in.

On external validity, the miscalibration is not only a synthetic knob. A mixed fleet parameterised from the stated-versus-realised gaps MarketBench measures for current coding models reproduces the effect, and the track-record correction still recovers completion on it (section 3). Beyond that, a live pilot with real Claude coding agents measured their calibration directly on coding micro-tasks across two model families, three Opus 4.8 and three Haiku 4.5 agents over a nineteen-task suite with six harder trap tasks (114 attempts): both families were underconfident (stated about 0.93, delivered 1.0), which confirms on real data that a coding agent's self-reports are miscalibrated, the premise of the study, and matches the direction MarketBench reports for the Claude models against the overconfidence it reports for a Gemini-class model. The sign of the bias is model-dependent, which is why the correction reweights by an observable track record rather than assuming a direction. On solvable in-distribution tasks the direction here is underconfidence, so a naive auction would under-select capable agents rather than over-select

bad ones; the harder tier did not flip the sign and the small model was no more overconfident than the frontier one, so the overconfident failure region is out-of-distribution and populating it needs tasks beyond a model's competence or an in-distribution-overconfident model family. A two-sided real calibration curve from such a task or model choice, and the full live-swarm allocation, are the next steps.

The scaling result has a direct commercial reading. Turning the pair-evaluation count into money at representative model prices, a central scheduler that scores every agent-task pair at one node pays a bill that grows as `N` times `M`, while a decentralised fleet caps each agent at a bounded sample of tasks, so the busiest node's bill is flat in the population size and the total grows only linearly. At the standard tier the central cost per allocation round rises from about fourteen dollars at a hundred agents to about a hundred and forty-four thousand dollars at ten thousand agents, whereas the decentralised total is about five hundred and seventy-six dollars and no single node pays more than about six cents, a saving of roughly two hundred and fifty times at the top of the range. The prices are illustrative list tiers rather than a live quote, and the absolute figures move with them, but the shape, a quadratic central bill against a flat per-node decentralised one, is a property of the coordination structure and is what makes the decentralised design a product rather than only a result: it removes the coordinator that is both the scaling bottleneck and the largest line item.

Cost is one axis of a product; speed is the other, and the same bid exposes it as a dial. The latency term in the Binding Energy carries an exponent, and sweeping it traces a latency-quality frontier: at weight zero the bid ignores latency and reaches the highest realised quality (about 0.91) at the highest mean latency, and as the weight rises the allocation favours faster agents, giving up quality (down to about 0.87) for lower latency (from about 1.0 to about 0.66 in the same units). Every point on the swept curve is non-dominated, so this is a genuine tradeoff rather than a single operating point, and a deployer picks where on it to sit by one parameter, quality-first for a batch pipeline or latency-first for an interactive one, without retraining or re-architecting. Together the cost curve and this frontier are the two knobs that turn the mechanism into a tunable service. A runnable proof of concept (`examples/poc`, `python -m examples.poc`) shows the whole flow offline: a small fleet self-selects through the calibrated gated bid, the integrity gate stops the out-of-scope actions of adversarial agents, and the central per-node bill is reported against the decentralised one.

These findings are full protocol scale (20 seeds, agent counts to 10,000) but on synthetic populations, with the calibration anchored to real agents only through the fleet archetypes and a two-family pilot (Opus 4.8 and Haiku 4.5) that, on solvable in-distribution tasks, was uniformly underconfident, so they are strong evidence for the mechanism rather than a deployment claim. A two-sided real calibration curve needs the overconfident arm, which is out-of-distribution for

these models on standard tasks; sourcing it from tasks beyond model competence or an in-distribution-overconfident model family, together with the full live-swarm allocation over the concurrent store just validated, is the path to definitive results, and the pre-registered hypotheses and falsification criteria in section 2.6 stand regardless of these verdicts.

Where this research stands

The mechanism is established and the framework is complete end to end. Eleven pre-registered hypotheses on synthetic populations are evaluated at full protocol scale, nine supported and two (the quality claims against a full-information optimum) reported as honest negatives. Two further hypotheses, the task-wrapper lift and the calibrated-routing cost-efficiency, are added on real Claude agents and supported with bootstrap confidence intervals. The coordination and calibration results are strong: the flat-versus-quadratic scaling is analytic and confirmed by a log-log fit, and the track-record correction is highly significant after multiple-comparison correction and robust across two generative families and a measured calibration mixture. The real-agent results are now powered: the capability ladder rests on about ten agents per cell and the project on five agents per cell, each with bootstrap confidence intervals, and the cost multiples are computed at representative price tiers rather than a live quote. The honest boundary of the work is therefore clear. It is a mechanism study with real-agent confirmation, not a benchmarked production system, and every claim in this paper is marked against that boundary.

Next steps

Three lines follow directly. First, power the real-agent results: raise every ladder and project cell to five or more agents so the completion and cost estimates carry confidence intervals throughout, and add a second project in a different domain to show the wrapper effects generalise beyond one task family. Second, close the open calibration question: the reliability curve is one-sided because both Claude families are underconfident in distribution, so populating the overconfident arm needs either tasks beyond a model's competence or a model family that is overconfident in distribution, which would also exercise the router's escalation on real data (the mechanism is in place but did not fire once the cheap model proved reliable enough). Third, run the full live-swarm allocation over the concurrent store already validated under real multi-process contention, so the pilot becomes a genuine competing swarm rather than a set of independent solvers. The publication track (a formatted submission, a related-work table against RouteLLM, EvoRoute, DiSRouter and MarketBench) and the productisation track (the wrapper layer as a hosted service) run in parallel to these.

Impact

The scientific contribution is to name and fix the failure mode of self-selection. Decentralised self-selection is well studied, but this work shows that its load-bearing assumption, that an agent

can assess its own fit, is exactly where language-model agents are known to be weak, and that a correction reweighting by an observable track record, agnostic to the direction of the miscalibration, recovers what that weakness costs without any privileged information. The engineering contribution is a decentralised allocation layer that removes the coordinator as both the scaling bottleneck and the largest line item, exposed as a tunable service with cost and latency dials. The most consequential contribution is practical: the two wrappers turn the mechanism into a cost result on real agents, and the interface-contract task wrapper is shown to be a precondition for decomposed multi-agent work to integrate at all, a result the project experiment makes vivid because even the frontier model fails the project without it. If that result holds at scale, it changes how agent systems are built, from one expensive generalist per task toward a swarm of cheap, wrapped specialists routed by calibrated confidence, which is the subject of the outlook below.

Limitations and how they are addressed

The study's limits are stated plainly here, each with its current status, so a reader can see what is a deliberate design choice, what is already mitigated, and what remains open.

LIMITATION	STATUS	HOW IT IS HANDLED OR WILL BE
Quality is about 94 per cent of the full-information optimum (H2 not supported)	By design, reported honestly	The decomposition shows about three quarters of the gap is quality deliberately traded for latency by the cost-aware bid; a quality-first configuration reaches the optimum's margin at the expense of speed, and the operating point is a tunable dial, not a ceiling.
The real-agent results are now powered (ladder about ten agents per cell, project five)	Largely closed, marked throughout	The full ladder rests on about ten agents per cell and the project on five, each with bootstrap confidence intervals (Haiku ladder bare 0.950 [0.90, 0.988], wrapped 0.986 [0.958, 1.00]; project bare 0 and wrapped 1.00 for every model in every replicate); a second held-out suite and a live price quote are the remaining real-agent next steps.
No live validation against a naturally overconfident agent	Open, scoped	Both Claude families are underconfident in distribution, so the overconfident arm is out of distribution here; populating it needs tasks beyond a model's competence or a model family that is overconfident in distribution, which is a deliberate task-design or model-choice exercise.
Bid gaming (an agent inflating its self-report)	Addressed	The track record demotes such an adversary as its realised failures accrue, with no anticipation of the gaming needed (strategic-adversary result, this section).
Reputation gaming (manipulating the track record itself, for example sandbagging)	Addressed, with a residual	A sandbagging adversary exploits the cumulative record's lag for a few rounds; a recency-weighted reliability window shortens the exploitation, an exposure cap bounds the first defection's blast radius (about 17 to about 4 failed wins at a cap of one), and the scope-based gate backstops out-of-scope defections (section 3). The residual is that no reactive record pre-empts the first defection, only bounds it, and cross-agent collusion is future work.
The atomic claim assumes a single logical coordination store	Design boundary	The framework reduces central work rather than removing it entirely; the claim is validated under real multi-process contention, and the store is self-contained (SQLite by default).

5. Building the wrappers

The two wrappers are small enough to state precisely, and the reference implementation is `src/cta/wrappers.py` with a runnable demo in `examples/wrapper_demo.py`. This section describes how each is built and, in particular, how the orchestrator, not the worker, constructs the task wrapper.

5.1 The task wrapper, and how the orchestrator builds it

A task wrapper is an explicit interface contract: a task name, a signature, a one-line description, a set of named acceptance criteria, a few worked examples, and a short self-check the agent runs before it commits to an answer. In the reference code this is a `TaskContract` whose `envelope()` renders the prompt an agent actually receives. The bare task ("write `is_match`") becomes the wrapped task ("write `is_match` with this signature, where `*` matches any sequence including the empty string, and check these three cases first"). The measured effect is twofold: it lifts a weak model to a strong model's completion by naming the exact edge case the weak model would otherwise miss (the capability ladder), and it fixes the interface so that pieces built by different agents, or different models, snap together (the dependency-graph project). This is the same task wrapper the framework uses to score allocation: the contract is the content of the task's advertised envelope, and an agent's compatibility `c` (E13, `docs/measures.md`) is the match of that agent against this contract, so the wrapper both states the task's acceptance criteria for the worker and yields the compatibility that drives self-selection.

The wrapper is owned and built by the orchestrator, which is what makes it stable across a heterogeneous, changing fleet: the contract does not depend on which model happens to take the task. An orchestrator can construct the contract from whatever the task already carries.

- From a machine-checkable spec. If the task has a type signature, a JSON schema, or a property or test file, the signature and the return shape come straight from it, and each property the tests assert becomes an acceptance clause (an empty input returns this, a repeating case wraps like that, a numeric field skips non-numbers). This is a mechanical contract compiler and needs no model call.
- From examples. A handful of labelled input and output pairs become the self-check block directly, which is often enough to pin the interface a loose description leaves ambiguous.
- From a description, via a wrapping model. When only a natural-language description exists, a small wrapping model is prompted to emit the contract fields, and the emitted contract is then validated against whatever checks do exist before it is advertised. The orchestrator can afford

to spend a cheap model here because the contract is amortised over every agent that will attempt the task.
- From failures, as feedback. When a downstream integration fails because two pieces disagreed on an interface, the mismatch itself (a dict where a tuple was expected, a separator line the reader did not want) is added as a new acceptance clause, so the contract sharpens over the deployment's life exactly where real integrations broke.

The design rule is that the acceptance criteria should name the properties a hidden test would check, not restate the obvious, because the lift comes from surfacing the unstated edge cases that separate a competent-looking answer from a correct one.

5.2 The agent wrapper

The agent wrapper turns a fleet of models into a single calibrated router. Each model is a `Model` with a price tier; a `Fleet` holds the models cheapest first and a per-task reliability table, the track record it accumulates from realised outcomes. Routing a task is one rule: each model reports a self-assessed confidence (the bid), the bid is discounted by that model's reliability on this task type to give the corrected bid, and the task goes to the cheapest model whose corrected bid clears the activation barrier, or, if none clears it, escalates to the model with the highest corrected bid. A winner is screened by the integrity gate before it acts. Reliability is updated after each outcome by a moving average, so the router is competent about the fleet from the first realised results rather than assuming a fixed capability, which is the difference from a static-capability router. The dollar cost of a routed plan against always using the strongest model is a direct sum over the chosen models at their tiers.

5.3 The loop that ties them together

The two wrappers are not a single call but a loop, and the loop is where the results compound. Tasks arrive and the orchestrator wraps each one; the wrapped envelope is advertised; agents bid their self-report; the agent wrapper routes each task to the cheapest model whose corrected bid clears the barrier; the winner executes the wrapped task; the output is scored and screened by the gate; and the realised outcome updates the track record that the next round's routing uses. Tasks that no one will claim have their barrier annealed down over rounds so a feasible task is never starved. Two things improve over the life of the loop that a single call cannot show: the track record sharpens, so more tasks safely drop to cheaper models, and the task contracts sharpen from integration failures, so the weak models grow more reliable on exactly the tasks that used to break. Calibration robustness and cost efficiency are properties of this loop, not of any one allocation.

6. Outlook: microagent swarms, harnesses, loop engineering, and token economics

The wrappers point at a shift in how agent systems are built. The dominant pattern today is a single, large, general model invoked per task, sometimes fronted by a router that picks a model by static capability. That pattern pays frontier-token prices everywhere and, as the project experiment shows, still cannot decompose a task across independent workers without an interface contract to make the pieces fit. The alternative this work supports is a swarm of microagents: small, cheap, specialised workers, each a modest model inside a tight harness, coordinated by calibrated self-selection.

Three ingredients make that alternative viable, and each maps to a result here. The **harness** is the model plus its scaffolding, its tools, and the wrapped-task envelope that is its contract with the orchestrator; the task wrapper is what lets a microagent on a cheap model be reliable on its slice of the work, because the contract supplies the competence the model lacks by naming the edge cases. **Loop engineering** is the recognition that the value is in the outer loop rather than any single call: self-selection, execution, gating, track-record update, and annealing, run over many rounds, are what accumulate the calibration that makes routing cheap and the contracts that make decomposition hold. **Token economics** is the consequence: model choice becomes a portfolio decision rather than a fixed default. The activation barrier is price-aware admission control, so a deployment pays economy-tier tokens for the tasks a wrapped cheap model can clear and reserves premium-tier tokens for the residual the barrier escalates, and because the track record sharpens over the loop's life, the share of tasks that drop to cheaper tiers rises and the blended cost per task falls over time. The two cost results compound: the decentralised substrate keeps the coordinator's per-node bill flat as the swarm grows, and the calibrated router keeps the workers' bill near the cheapest-capable model rather than the strongest.

This loop is a **harness** in the sense recent work gives the term: the software around a base model that decides how it plans, what it remembers, and how its work is judged, and which is emerging as the near-term locus of self-improvement rather than the model weights (Weng, 2026). That literature's own diagnosis is that a self-improving loop is only as good as the signal it optimises, and that the evaluator is the weak link, because outcomes are noisy and self-reports are unreliable. CTA speaks directly to that gap: the track-record correction and the integrity gate are a mechanism for turning an unreliable self-report into a trustworthy allocation signal, so the piece a self-improving harness most lacks is the piece this framework supplies. H13 makes the self-improvement concrete rather than asserted. Starting from an empty track record, the same loop, with no privileged information, lifts completion from 0.36 to 0.84 over a dozen rounds toward the full-information oracle purely by remembering which agents actually delivered, while a memoryless raw baseline on the same task stream stays flat; the persistent, accumulating record

is the harness's memory, and using it in the bid is what makes the allocation improve from its own experience. We claim CTA as a component of such a harness, its calibration-robust signal layer, not a full self-improving system; the meta-level frameworks that optimise the harness code or the context playbook itself (context-evolution and meta-harness approaches surveyed by Weng, 2026) are a complementary direction rather than one this paper builds.

The claim to a breakthrough rests on combining three things that are usually pursued separately. Decentralised self-selection removes the coordinator bottleneck but has, until now, trusted a self-report that language-model agents get wrong; capability routers correct for capability but not for a model being miscalibrated about itself, and they route whole tasks rather than composing decomposed work; and interface contracts are familiar in software engineering but have not been identified as the precondition for a multi-agent swarm to build software at all. CTA's contribution is to hold all three at once: a decentralised substrate, a routing rule that corrects the self-report by an observable track record and screens the winner for safety, and a task wrapper that is the contract making a swarm's output composable. If the small-sample real-agent results are borne out at scale, the enabling condition follows, that agent systems can be scaled by the count of cheap microagents rather than by the size of one model, which is a different and far cheaper axis of growth. This outlook is a direction the mechanism supports and the real-agent experiments illustrate, not a benchmarked deployment, and the next-steps programme above is what would turn the direction into a claim.

---

7. Conclusion

This paper studied Chemotactic Task Allocation, a decentralised, biomimetic framework in which tasks advertise contracts and agents self-select by a calibrated, reliability-weighted, gated bid. The decentralised design holds the busiest node's coordination load flat as the fleet grows to ten thousand agents, where a central scheduler grows quadratically, and against the information-bounded scheduler a deployment would actually run it matches fresh central allocation and overtakes it as that scheduler's information goes stale. The load-bearing assumption of self-selection, that an agent can assess its own fit, is exactly where language-model agents are miscalibrated; that miscalibration is the measured failure mode, and a track-record correction using no privileged information recovers the completion it costs, with an integrity gate as the safety backstop and the activation barrier providing liveness and specialist routing. On real Claude agents across three model families, the two wrappers turn the mechanism into a cost result: an explicit interface contract lifts a weak model to a frontier model's completion and is the precondition for independently built pieces to integrate, and a calibrated router then delivers frontier-level completion at roughly one fortieth of the always-frontier cost. Against a full-information optimum the quality claim does not clear its margin and is reported as such. The

synthetic and calibration results are at full protocol scale and reproducible from seeds by one command; the real-agent wrappers are powered (the ladder about ten agents per cell, the project five) with bootstrap confidence intervals, whose residual limits are marked throughout. The practical thesis, that a swarm of cheap, wrapped microagents coordinated by calibrated self-selection can match a single expensive model at a fraction of its cost, is the direction this work opens and the programme above is designed to confirm.

References

Bonabeau, E., Theraulaz, G., & Deneubourg, J-L. (1996) Quantitative study of the fixed threshold model for the regulation of division of labour in insect societies. Proceedings of the Royal Society of London B: Biological Sciences, 263(1376), 1565-1569. DOI: 10.1098/rspb.1996.0229.

Cemri, M., Pan, M.Z., Yang, S., Agrawal, L.A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., Zaharia, M., Gonzalez, J.E., & Stoica, I. (2025) Why do multi-agent LLM systems fail? arXiv:2503.13657.

Chen, L., Zaharia, M., & Zou, J. (2023) FrugalGPT: how to use large language models while reducing cost and improving performance. arXiv:2305.05176.

Dias, M.B., Zlot, R., Kalra, N., & Stentz, A. (2006) Market-based multirobot coordination: a survey and analysis. Proceedings of the IEEE, 94(7), 1257-1270. DOI: 10.1109/JPROC.2006.876939.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996) Ant System: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B, 26(1), 29-41. DOI: 10.1109/3477.484436.

Fitzpatrick, J.L., Willis, C., Devigili, A., Young, A., Carroll, M., Hunter, H.R., & Brison, D.R. (2020) Chemical signals from eggs facilitate cryptic female choice in humans. Proceedings of the Royal Society B: Biological Sciences, 287(1928), 20200805. DOI: 10.1098/rspb.2020.0805.

Fradkin, A., & Krishnan, R. (2026) MarketBench: evaluating AI agents as market participants. arXiv:2604.23897.

Gerkey, B.P., & Matarić, M.J. (2004) A formal analysis and taxonomy of task allocation in multi-robot systems. The International Journal of Robotics Research, 23(9), 939-954. DOI: 10.1177/0278364904045564.

Kuhn, H.W. (1955) The Hungarian method for the assignment problem. Naval Research Logistics Quarterly, 2(1-2), 83-97. DOI: 10.1002/nav.3800020109.

Ong, I., et al. (2024) RouteLLM: learning to route LLMs with preference data. arXiv:2406.18665.

Smith, R.G. (1980) The Contract Net Protocol: high-level communication and control in a distributed problem solver. IEEE Transactions on Computers, C-29(12), 1104-1113. DOI: 10.1109/TC.1980.1675516.

Weng, L. (2026) Harness Engineering for Self-Improvement. Lil'Log. https://lilianweng.github.io/posts/2026-07-04-harness/.

Zhang, A.L., Kraska, T., & Khattab, O. (2025) Recursive Language Models. arXiv:2512.24601.

Zhang, C., Zhu, Z., Wei, Y., Tian, B., Liu, J., Wang, H., Wang, X., & Liu, Y. (2026) Confidence-calibrated small-large language model collaboration for cost-efficient reasoning. Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2026). arXiv:2603.03752.