MIDIAN: METRIC-INDUCED DISTRIBUTED AGENT NETWORKS FOR SCALABLE, ADVERSARIALLY ROBUST TASK ALLOCATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent frameworks route tasks by having a supervisor model read the agents' self-descriptions and select one. This suits small, hand-assembled teams but breaks down as agent networks grow large, specialized, or dishonest. From a hundred to a hundred thousand agents, every tested framework performs poorly even with dense retrievers and task-specific retrieval instructions, and it degrades to near-random with standard retrievers. We propose MIDIAN, a hierarchical router that replaces self-description with verified competence. Small cohorts of agents probe one another on tasks with checkable outcomes, each cohort summarizes its best member per task family up the tree, and tasks descend the tree in logarithmically many steps. A small audit of peer reports keeps the estimates accurate under dishonest reporting. We introduce a live execution benchmark of LLM agents with ground-truth skill, and also evaluate on RouterBench, RouterEval, and LLMRouterBench. On the specialist populations, every framework we test improves when given MIDIAN's shortlist in place of its default retriever, and on large populations MIDIAN on its own matches or outperforms the most widely used frameworks and published routers. Its advantage grows with population size, reaching 0.97 of the oracle at a hundred thousand agents. It also grows when agents are dishonest, above all when low-skill agents collude. In that regime, the declared argmax and MIDIAN without defenses lose accuracy. Because MIDIAN makes no model calls at routing time, its cost is a one-time build, repaid after about ten tasks per agent; after that, each routing decision costs under a thousandth of a supervisor call.

1 INTRODUCTION

The multi-agent frameworks in wide use all follow one paradigm for distributing tasks: a supervisor model reads a roster of agent descriptions and picks one. AutoGen's selector, LangGraph's supervisor, CrewAI's manager, Magentic-One's orchestrator, and the handoffs of the OpenAI Agents SDK and Google ADK all work this way (Section 2). The open agentic web takes the same view: the NANDA index resolves an agent's name to a signed record of its capabilities (Raskar et al., 2025), and Agent-to-Agent (A2A) cards and Model Context Protocol (MCP) server descriptions are self-describing capability documents (A2A Project, 2026; Anthropic, 2026; Yang et al., 2025). With a few agents of similar ability, this approach performs close to an oracle. We study what happens as the population grows to thousands of specialized agents, some of them dishonest, and what should replace description-based routing.

Three failures appear, in order of the scale at which they occur. First, rosters no longer fit in a prompt: beyond about ten agents, every framework retrieves a shortlist of descriptions and lets the supervisor choose among them. But retrieval over prose ranks agents by the words they share with a task, not by how often they solve tasks of that kind, and the frameworks recover only part of the skill even a verified shortlist contains. Second, descriptions lose discriminative power: a paragraph is a coarse summary of a vector of skills, adequate when agents are uniformly strong or weak and uninformative once they specialize. Third, descriptions were never reliable: a self-description is produced by the model it describes, and models overrate their own competence, most on unfamiliar tasks (Xiong et al., 2024; Kadavath et al., 2022; Yin et al., 2023). A self-interested agent also has an incentive to overstate. Self-advertised metadata lets an agent make unverifiable claims (Raskar et al., 2025), and

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

attractive metadata alone redirects tool selection in most settings (Mo et al., 2025; Wang et al., 2026). Once the description is untrustworthy, a natural alternative is to ask peers, but a peer's report can be falsified as well (Kamvar et al., 2003; Douceur, 2002).

Verified outcomes are the alternative signal; two literatures already route on them. LLM routers learn a policy from labeled prompt–model pairs (Hu et al., 2024; Ong et al., 2025; Feng et al., 2025; Zhuang et al., 2025), but they train centrally on fixed pools of two to about a hundred models and hundreds of thousands of labels, and often fail to beat a best-single-model baseline (Li et al., 2026). Crowdsourcing and best-arm identification aggregate labels and probe outcomes collected by a trusted observer (Section 2); neither was built for a population that is decentralized and partly dishonest. Peer prediction supplies the missing piece: a little ground truth checked against individual reports buys incentives that agreement alone cannot provide (Gao et al., 2019).

MIDIAN (Metric-Induced Distributed Agent Networks) combines these pieces. Agents form cohorts of $r{=}10$. Cohort members probe one another on tasks with checkable outcomes and report what they saw; each member's estimated skill per task family is a trimmed mean of its peers' reports, so a minority of false reports cannot change it. Each cohort passes its best member per family up a tree, and a task descends the tree in $\lceil \log_r n \rceil$ steps with no model call. A five-percent audit of the reports, excluding any reporter caught misreporting twice, protects the estimates from colluding reporters. Together, re-verifying each candidate before promotion and caching the root's pick halve the cost of a route. The tree is built once and then actively maintained at low cost: every routed outcome updates the path it took, and audits continue at the same rate. We evaluate it live on populations of a hundred to a hundred thousand LLM agents executing verifier-checked tasks, so every routing decision is scored on a real outcome. We also evaluate on three of the routing literature's own benchmarks and in calibrated simulation to ten million agents, against ten deployed frameworks, eighteen routers and bandits, and a production retrieval stack.

Contributions. We make five claims. **(i) Frameworks are adequate in small, homogeneous populations and fail at scale.** On a homogeneous pool of ten real LLMs every framework we test is within 0.03 of the oracle; at a thousand LLM agents the mean framework trails a plain argmax over the agents' declared skills, and at ten thousand, on the default retriever, it trails a random pick (Section 4.2). **(ii) On large populations MIDIAN matches or outperforms every baseline, with a lead that grows with population size.** On live LLM populations its paired advantage over a flat scan with the same probe budget and no tree grows from $+0.001$ at a hundred agents to $+0.09$ at a hundred thousand, where MIDIAN reaches 0.97 of the oracle. The ordering holds at a million replayed RouterBench agents and at ten million agents in simulation. On the two RouterEval-family pools, which are small or dense, a method that learns from outcomes is more accurate at $b{=}3$ (Section 4.3). **(iii) The advantage is largest under the cartel.** A cartel of low-skill agents that inflate their claims and endorse one another costs MIDIAN at most 0.014 on any live population, while the declared argmax loses 0.04–0.13 live and 0.08–0.16 of the oracle elsewhere, MIDIAN without defenses 0.03–0.05 and 0.17, and the report- and bid-based schemes of Table 1 up to 0.36 (Section 4.4). **(iv) MIDIAN is a shortlist provider for any existing framework.** On every live specialist population the MIDIAN shortlist improves every framework we run over its default retriever, and nine of ten over its best text retriever at 10^5 (Section 4.5). At a hundred thousand agents, mean framework success rises from 0.38 on the default retriever (0.48 on the best text retriever) to 0.57 when agents are honest and 0.56 under the cartel. **(v) MIDIAN routes at a small fraction of a framework's cost.** MIDIAN routing makes no LLM calls. It costs $r\lceil \log_r n \rceil$ comparisons and 5–9 mJ from a thousand to ten million agents, versus n comparisons for every scan and 20.6–220 J per task for the frameworks. The one-time build is repaid after ten tasks per agent versus the cheapest framework and one versus the costliest. Every routing decision costs under a thousandth of a supervisor call (Section 4.6).

2 Related work

Orchestration frameworks. AutoGen (Wu et al., 2024), MetaGPT (Hong et al., 2024), CAMEL and its Workforce (Li et al., 2023; Hu et al., 2025), Magentic-One (Fourney et al., 2024), and the supervisor, handoff, and manager primitives of LangGraph, CrewAI, the OpenAI Agents SDK, Google ADK, LlamaIndex, smolagents, and the Microsoft Agent Framework (LangChain, Inc., 2023; CrewAI, Inc., 2023; OpenAI, 2025; Google, 2025; LlamaIndex, 2022; Hugging Face, 2024; Microsoft, 2025) all delegate by having a language model read a roster of descriptions, except MetaGPT, which routes work through a fixed pipeline of roles. Recent benchmarks study multi-agent systems on

small teams: performance is governed by orchestration architecture, not model capability alone (Orogat et al., 2027), and by the fit between task and agent capabilities (Ke et al., 2026). A team's communication graph, likewise, can be learned or rewired per task rather than fixed (Zhuge et al., 2024; Liu et al., 2024). AgentsNet evaluates distributed protocols over up to a hundred agents but contains no routing decision (Grötschla et al., 2025). To our knowledge, ours is the first benchmark to run these frameworks unmodified over populations of 10^2 to 10^5 agents with known skill and a controllable fraction of liars.

LLM routing. RouterBench (Hu et al., 2024), RouteLLM (Ong et al., 2025), GraphRouter (Feng et al., 2025), EmbedLLM (Zhuang et al., 2025), and Avengers (Zhang et al., 2026) learn to send a prompt to one model of a fixed pool of a few to about a hundred, from large labeled sets; RouterEval scales the pool to thousands of real LLMs (Huang et al., 2025) and LLMRouterBench finds that routers often fail to beat a best-single-model baseline (Li et al., 2026). DiSRouter removes the central router: models are arranged in a chain and each decides, from its own self-assessment, whether to answer the prompt or pass it on (Zheng et al., 2026). These routers assume a central trusted trainer and honest labels; we run their published methods on our probe budget and on their own benchmarks, and add a dishonest fraction.

Task allocation and reputation. Contract-net negotiation (Smith, 1980) and the market-based architectures cataloged by the multi-robot allocation taxonomy (Gerkey & Matarić, 2004) allocate tasks on self-reported bids, which an agent can inflate. Reputation systems aggregate peer reports (Jøsang et al., 2007); EigenTrust weights them by the reporters' own reputation (Kamvar et al., 2003), and any such scheme falls to forged identities without an identity authority (Douceur, 2002). In peer-to-peer networks, super-peer overlays elect one representative per group on self-reported capacity, a two-level hierarchy that we run as a baseline (Yang & Garcia-Molina, 2003; Montresor, 2004). Two older results shape MIDIAN's tree: hierarchical routing keeps each node's routing table logarithmic in the network size (Kleinrock & Kamoun, 1977), which motivates the tree's $\log n$ routing depth, and the economics of knowledge hierarchies favors a small, constant span of subordinates per level (Garicano, 2000), which motivates the cohort size.

Verification, bandits, and self-assessment. Crowdsourcing infers worker reliability from labeler agreement (Dawid & Skene, 1979) or scores it against gold items (Ho et al., 2013); peer prediction shows that a small amount of ground truth checked against reports buys incentives that agreement alone cannot provide (Gao et al., 2019), which is the design of our audit. Best-arm identification allocates a probe budget adaptively (Audibert et al., 2010; Karnin et al., 2013), and bandits and skill-rating systems learn online (Auer et al., 2002; Chapelle & Li, 2011; Li et al., 2010; Herbrich et al., 2006); all assume a trusted observer, and we run them as baselines. Language models overrate their own competence (Kadavath et al., 2022; Yin et al., 2023; Xiong et al., 2024) and prefer their own outputs when judging (Zheng et al., 2023; Panickssery et al., 2024), and attractive metadata alone can redirect tool selection (Mo et al., 2025; Wang et al., 2026).

3 MIDIAN: A TREE OF VERIFIED COMPETENCE

Terms. A *model* is a language model, and an *agent* is a model behind a persona. A *task* belongs to one of F *families* (a kind of task). An agent's *skill* on a family is its probability of solving a task of that family, and a *specialist* is an agent whose skill is high on a few families and low elsewhere. A *framework* is an orchestration library that runs a team of agents; its *architecture* is how it wires and supervises them, and the *paradigm* every framework in our evaluation shares is a supervisor model that reads agent descriptions and picks one. A *method* is any routing procedure we score; an *arm* is one bar of a figure, a method or a cross-fitted pool. A *router* is an algorithm that picks an agent from declarations, probe outcomes, or peer reports without a supervisor. A *declared* value is what an agent says about itself. A *scan* reads one value for every agent and takes the argmax. A *retriever* ranks agent descriptions against a task by text, a *dense retriever* by embedding similarity, and a *shortlist* is the k candidates a shortlist method hands a framework. A *population* is one set of n agents. A *backend* is the source of outcomes: live execution, a recorded benchmark, or simulation. A *regime* is the honest or lying condition, and the *oracle* is the method that reads true skill. A *cell* is one combination of backend, population, regime, and budget; a *unit* is one (cell, seed) run; a *grid* is a sweep of cells.

MIDIAN reads only probes and reports, never a declaration (an agent's self-description or self-rating). It has three parts: a tree of cohorts whose estimates come from peers, a routing rule that descends the tree, and two defenses that keep the estimates accurate when reporters are dishonest. Everything below runs live (Section 4.1): a probe is a model call on a fresh instance scored by its verifier, a report is a message from a peer that observed it, and an audit during the build re-runs a real instance, while online the routed task's own verified outcome serves as the audit, so nothing is re-run.

Cohorts and estimates. Agents are partitioned uniformly at random into leaf cohorts of $r{=}10$. Each member is probed b times per family, the probe budget $b \in \{1, 3, 5\}$ of Section 4.1. MIDIAN never receives a probe's outcome directly. The other $r{-}1$ members observe the probe and report what they saw; the estimate is built from those reports, and ground truth enters the tree only through the audits below. The estimate $\hat{S}_{af}$ is a trimmed mean of the reports about (a, f), dropping $\lfloor \delta(r-1) \rfloor$ reports from each side with $\delta{=}1/3$, so three of nine reporters can be wrong about any single probe without changing the estimate. Cohorts are the *leaves*: they are grouped r at a time into parents, the parents are grouped again, and so on up to a single *root*. Every node between is an *internal node*, and a node's members are the agents in its subtree. The depth is $\lceil \log_r n \rceil$: two levels at a hundred agents, five at a hundred thousand.

Summaries and routing. Each node stores, per family, the largest estimate in its subtree and the child that holds it; a parent's summary is the maximum over its children's. A task of family f enters at the root and descends by largest summary, reaching a leaf in $\lceil \log_r n \rceil$ steps and $r\lceil \log_r n \rceil$ comparisons, versus n for any flat scan. The tree is constructed once from the build probes and is then maintained by three operations. An *update*: when the routed agent's outcome comes back, its estimate moves by a running mean and the summaries along its path are recomputed ($r\lceil \log_r n \rceil$ comparisons), which can promote or demote that agent at every level of the path. An *exclusion*: a reporter caught by an audit (below) is removed from all later aggregation and its cohort is re-aggregated from stored per-peer means (a cost independent of n), followed by one path recomputation. A *re-verification*: a candidate forwarded above the leaves is re-probed by other cohorts (below). These bounds are worst-case, not amortized: T routed tasks and x exclusions cost $O((T + x) \log n)$ comparisons plus x constant-size re-aggregations, and a reporter is excluded at most once, so $x \leq n$. Populations are fixed here; in principle, a join is a leaf insertion at the cost of its own Fb probes and a leave is an exclusion.

Defenses against a cartel of liars. The adversary is a cartel: a fraction β of agents, the least skilled first. Cartel members inflate their declared skill (by 0.4 in every family, clipped to one); as reporters, they endorse one another and vote down the strongest fifth of the honest agents they observe, while executing at their true skill. Liars may instead be drawn at random, the milder case at small populations (Section 4.4). Throughout, "the cartel" means half the population lying, low-skill-first. MIDIAN never reads a declaration, so the inflated claim cannot reach it; the false reports can. A cartel corrupts the tree in two places. At the leaves, colluding reporters endorse one another and vote down the strongest fifth; with three colluders in a cohort of ten, the trim removes roughly one colluder's reports per side. Above the leaves, a corrupted cohort estimate is promoted unchanged, so one cohort's lie can become the root's pick for a family.

Audits. Five percent of level-0 probe instances are re-run by the cohort, and the true outcome is compared with every report about them. A reporter caught misreporting twice is excluded from all later aggregation. This is the spot-check of Gao et al. (2019). An honest reporter is struck only if a re-run flips the outcome. Online, audits continue at the same rate on routed outcomes, so nothing is re-run, and an exclusion re-aggregates the affected cohort from stored per-peer means. Audit re-runs are probes beyond the nFb build budget: they add 3–5% to it, at most 7% in any cell, while every baseline stays at or under nFb.

Verified promotion and a cached root. Before a parent compares its children, each forwarded candidate is re-probed at the parent by one randomly chosen member of each sibling subtree, so a cohort cannot vouch for its own member higher up. The verified mean is folded into the estimate, weighted by probe count. To pay for this, at $b \geq 3$ we reduce the leaf budget to $b_0{=}b{-}1$ probes per agent and family and spend the remainder on the candidates that reach the upper levels (at $b{=}1$ the single probe stays at the leaf): about nine verification probes per promoted candidate at $r{=}10$, so at $b{=}3$ verification outweighs the leaf probes for the candidates that determine the route. The root caches its per-family pick, so a route costs one comparison and two messages until an observation on the path changes the pick. At $b{=}1$ no probes remain for verification, and MIDIAN runs with audits

alone and a cached root; the tree with neither defense is *MIDIAN without defenses* in Section 4, the single-defense variants appear in Appendix B, and its cost is in Section 4.6.

4 EXPERIMENTS AND RESULTS

4.1 EXPERIMENT DESIGN

Three elements are fixed before any method runs: populations whose true skill is measured and then hidden, so the right agent for every task is known to the benchmark and to no method; a ledger that charges every channel a method reads; and an adversary.

Populations with known skill. A population has n agents and tasks fall into F families; true skill $S \in [0,1]^{n \times F}$, the probability that agent a solves a task of family f, is measured once and shown to no method. Success is the fraction of the Q routed tasks that the chosen agent solves, so every difference below is in units of task success unless stated as a fraction of the oracle. The oracle reads S and bounds success from above, up to sampling of the test stream. Five backends supply populations. Four replace live execution with recorded or simulated outcomes; three are the routing literature's own benchmarks, so the published routers are evaluated on their home ground: *replay* of RouterBench's eleven models to 10^6 agents (Hu et al., 2024), *RouterEval*'s pools of 10 to 5,000 leaderboard LLMs on MMLU (Huang et al., 2025; Hendrycks et al., 2021), twenty of *LLMRouterBench*'s open models (Li et al., 2026), and a *calibrated Bernoulli* simulation that resamples measured live skill rows to 10^7 agents (Appendix A). The fifth is live.

Live testing. A recorded or simulated backend can show only how a router reads a skill matrix; the live backend answers whether routing holds when the agents are real models, the frameworks real libraries, and every probe and task an actual execution. A live agent is an open model from a seven-rung ladder spanning 0.5B to 14B parameters (Qwen2.5 0.5B–14B, Gemma-2 2B and 9B (Qwen Team et al., 2024; Gemma Team et al., 2024)), so that skill varies widely. The model sits behind a persona naming its specialty families; on other families the worked example, family description, and Python tool are withheld, so the specialties are real. Tasks come from F=16 Reasoning Gym families (Stojanovski et al., 2025), each with a generator (so no instance is ever reused) and an exact verifier (so every outcome is checkable without a judge model; Appendix A lists them). Every probe is a model call by the probed agent, every routed task is executed by the chosen agent and scored by the verifier, and every framework runs through its own library with a real supervisor model. Behavior is fixed by the ⟨model, specialties, tool⟩ signature and memoized, so live populations of 10^2 to 10^5 agents are affordable (Appendix A).

Channels and the ledger. A method declares which of four channels it reads: *declared*, a self-rating matrix D that every agent publishes (a self-rating per family and, live, a 70-word self-description; on the other backends drawn from the live ratings' distribution given true skill; Appendix A); *probes*, at most b per agent and family, so a build costs at most nFb probes; *reports*, what a peer says it observed of another agent's probe; and a *bus*. Every channel use is charged in a ledger, every routed outcome is returned to the method, and no method sees S, the liar set, or β.

Liars. A fraction β of agents follow the liar model of Section 3, drawn at random or *low-skill-first*, the more important case: the agents with the most to gain from lying are the least able to do the work. The cartel is β=0.5, low-skill-first.

Baselines and protocol. The frameworks are the ten orchestration libraries of Section 2 (all but MetaGPT), which together cover the delegation primitives in current use; each runs through its own library at a pinned version, with a Qwen2.5-7B supervisor and the scaling adapter its documentation prescribes—a shortlist of k=10 descriptions from which the framework's own primitive selects (Appendix C). The routers are eighteen methods that route from the same four channels behind one `build`/`fetch`/`observe` interface, together covering every way of reading those channels: readers of declarations, trusted-observer probe scans and bandits, RouterBench's kNN and MLP routers on the same probe budget, and classical allocation and reputation schemes. Thirteen enter the figures, two directly (flat probe argmax and declared argmax) and eleven as members of two pooled bars, *best learned/declared router* and *best bandit* (the warm-start bandit's two priors count as one method); a pooled bar shows, for each seed, the pool member that performed best on the other seeds (Appendix B). Five do not: two exceed the budget every other method respects (verify-on-claim spends extra probes per task; route-to-k majority executes three agents) and three classical schemes

(gossip reputation, contract-net self-bid, referral network) fit neither pool; all five are scored on every backend in Table 1, and none beats MIDIAN on live, Bernoulli, or replay. Comparisons are paired: within a seed every method sees the same population, liars, task stream, and probe outcomes, and every whisker is ± 1 standard error over seeds. Unless stated otherwise, the population is the live specialist population at b=3, with ten seeds at $n \leq 10^3$ and three at $n \geq 10^4$.

4.2 Frameworks work on small teams and fail at scale

Frameworks route well when the pool is small and uniform, and poorly otherwise. On RouterEval's all-strong pool of ten real LLMs every framework we test routes within 0.03 task success of the oracle, and on the all-weak pool of ten the mean framework is 0.01 above it (the oracle is fixed on the training split and these two pools are not drawn; Appendix A). The shortfall opens only with heterogeneity and scale: the mean framework trails the oracle by 0.18 on a mixed pool of a hundred and by 0.22 on the five-thousand-model leaderboard (the mixed pools are Figure 10's groups). Live specialist populations show the same pattern (Figure 1; across n in Figure 7). In Figure 1 the best framework on its best text shortlist (rightmost bar) stays at 0.61–0.63 at every n, about 0.24 below the oracle; scale neither widens nor closes the gap. That figure is lenient-scored: a framework that names no valid agent falls back to the declared argmax of its shortlist. Scored strictly (such a task counted as failed), the best framework is 0.47–0.51 (Appendix C). With the default hashed TF-IDF shortlist (Figure 7), mean framework success falls from 0.48 at a hundred agents to 0.38 at a thousand and 0.29 at ten thousand, where every framework on that shortlist is below a uniform random pick (0.42). At a hundred thousand the shortlist is ten copies of one agent (same-signature agents write identical descriptions), and every framework returns 0.38.

That 0.38 floor is an artifact of one retriever. A production stack is the control: BM25 (Robertson & Zaragoza, 2009), MiniLM (Reimers & Gurevych, 2019), Qwen3-Embedding-8B dense retrieval with and without a task instruction, and reciprocal-rank fusion (Cormack et al., 2009) with a Qwen3-Reranker-4B cross-encoder (Zhang et al., 2025) (Appendix C). BM25 enters at every n through the fusion-plus-reranker shortlist. Every shortlist at n=10^5 is in Figure 2: better retrieval helps but does not close the gap. From hashed TF-IDF's 0.38 floor, MiniLM raises mean framework success to 0.46 and the best text pipeline, dense retrieval with a competence instruction, to 0.48, still 0.38 below the 0.86 oracle. The fusion-plus-reranker stack reaches only 0.43 (0.41 without an instruction; every variant in Figure 8), below its own dense component. The declared argmax over all 10^5 declarations scores 0.62 (the declared-argmax bar of Figure 1). Giving the frameworks the declared top ten as a shortlist (declared top-k in Figure 2) scores the same 0.62 on average: once the shortlist contains the right candidates, the supervisor recovers the argmax, and a good one adds a few hundredths. Relative to that 0.62, the text retrievers therefore lose 0.14–0.25 *before* the supervisor sees anything; even a perfect reading of the declared channel remains 0.24 below the 0.86 oracle; a larger supervisor does not help (Magentic-One's 14B orchestrator scores 0.013 below its 7B one, Appendix C).

4.3 MIDIAN's lead over every baseline grows with population size

The tree and the flat scan start level and separate as n grows. In Figure 1, under honest agents at b=3, MIDIAN rises from 0.78 at n=10^2 to 0.84 at 10^5 (0.81 at 10^3 and 10^4) while MIDIAN without defenses and the flat probe argmax decline from 0.78 to 0.75. Paired within seed, MIDIAN's lead over these two baselines grows from $+0.005$ and $+0.001$ at 10^2 to $+0.083$ and $+0.087$ at 10^5 (whiskers in Figure 5). With three probes per family hundreds of agents tie at three out of three; a flat argmax among them is arbitrary, and re-probing the candidates that reach the upper levels breaks the ties. When agents are honest, the best learned/declared router and the best bandit (Figure 1) sit at 0.74–0.77 at every live population, like the flat probe argmax, so at 10^5 MIDIAN leads every baseline by 0.08 or more.

The verification budget buys the lead: at b=1 no probes remain for verification, the warm-start bandit passes MIDIAN from 10^3 to 10^5 and the flat probe argmax at 10^5 (Figure 5), and at a thousand agents MIDIAN rises from 0.69 at b=1 to 0.81 at b=3 and 0.84 at b=5 (the overlaid bars of Figure 1), within 0.03 of the oracle.

The same ordering holds on every large population and reverses only on the two small or dense pools. On the three large populations of Figure 3 (each backend at its largest size, normalized by its oracle) MIDIAN reaches 0.95–0.97 of the oracle honest and 0.94–0.96 under the cartel on live, Bernoulli,

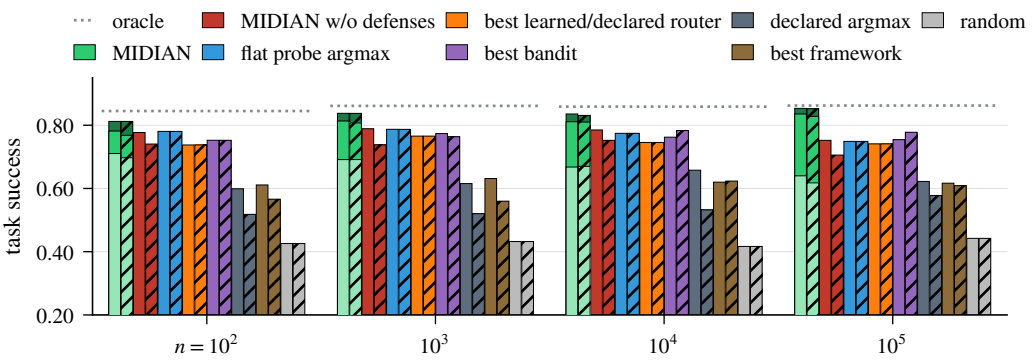

Figure 1: **MIDIAN is the most accurate arm from a thousand agents upward in both regimes (honest agents and the cartel), tied with the flat scan at a hundred, and its lead grows with n; under the cartel the declared argmax and MIDIAN without defenses lose.** Setup: mean task success of eight routing arms (MIDIAN, MIDIAN without defenses, flat probe argmax, declared argmax, best learned/declared router, best bandit, best framework, and random) on specialist populations of n=10^2–10^5 agents (F=16 families; 10 seeds at $n \leq 10^3$, 3 at $n \geq 10^4$) at b=3 probes per agent and family, solid under honest agents and hatched under the β=0.5 low-skill cartel; for MIDIAN the b=1 (light) and b=5 (dark) bars are overlaid behind the b=3 bar, so each visible segment is the gain from the next budget. Pooled arms (best learned/declared router, best bandit, best framework on its best text shortlist, lenient-scored; the same arm scored strictly is 0.13–0.21, and the best strictly scored framework is 0.47–0.51, Table 2) are cross-fitted per seed (Appendix B). Dotted line: oracle; axes start at 0.2. Other budgets, whiskers, and the five undrawn arms: Figure 5, Table 1.

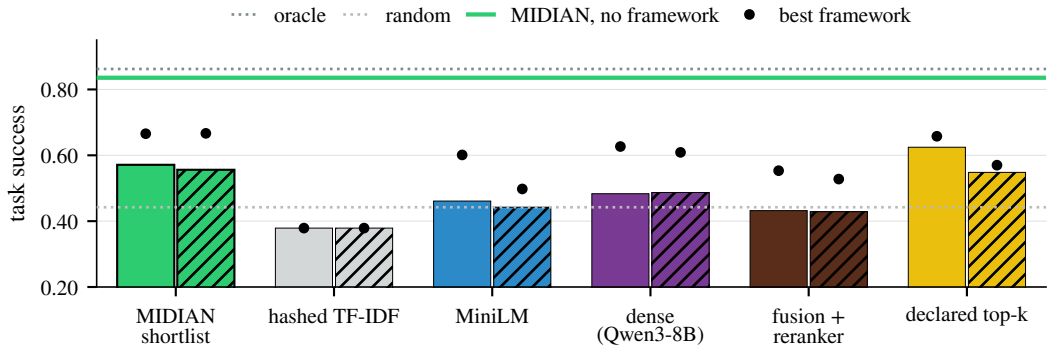

Figure 2: **Frameworks do better on MIDIAN's shortlist than on any text retriever, and under the cartel it beats every other shortlist—but no shortlist closes the gap to MIDIAN routing alone.** Each group of bars applies one shortlist method to the same ten deployed frameworks (AutoGen, CAMEL Workforce, CrewAI, Google ADK, LangGraph, LlamaIndex, Magentic-One, Microsoft Agent Framework, OpenAI Agents SDK, smolagents), each unmodified, on the live backend at n=10^5 (specialist population, b=3, 3 seeds), MIDIAN's shortlist at the left and the frameworks' own retrievers to the right. Solid bars: mean over the nine or ten frameworks with complete seeds under that regime, honest agents; hatched: the cartel; black dot: the best single framework (lenient scoring, Appendix C). When agents are honest, the declared top-k shortlist is best (0.62) and MIDIAN's is second (0.57, within 0.05). Under the cartel the ordering reverses: MIDIAN's shortlist holds at 0.56, 0.07 above the best text retriever (nine of the ten frameworks higher, exact paired test $p = 0.01$), while inflated claims pull declared top-k down to 0.55. Even the best framework on the best shortlist (0.67) stays 0.17 below MIDIAN routing alone (green line, 0.84). Dense and reranker at their best instruction (all variants: Figure 8). Upper dotted line: oracle; lower: random.

and replay, versus 0.87–0.91 for the flat probe argmax and 0.38–0.72 (honest) or 0.30–0.67 (cartel) for the declared argmax. Two baselines that learn from outcomes and read neither claims nor reports beat the tree on the RouterEval-family pools at b=3, in both regimes: on the 5,000-LLM leaderboard a contextual bandit reaches 0.89 of the oracle versus MIDIAN's 0.80, and on the twenty-model

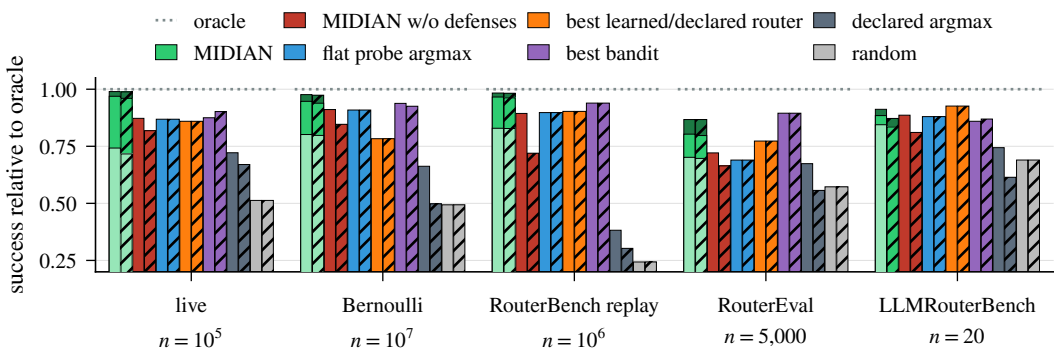

Figure 3: **On the large populations MIDIAN is the best arm in both regimes (honest and cartel); on the two small or dense pools of real LLMs (right) a method that learns from outcomes is more accurate (Section 4.3).** Setup: success divided by the backend's honest b=3 oracle (dotted line at 1) for live LLM agents (n=10^5, 3 seeds), calibrated Bernoulli (10^7, 100 seeds), RouterBench replay (10^6, 30 seeds $\times$ 3 population shapes), the RouterEval leaderboard (5,000 LLMs, 3 seeds), and LLMRouterBench (20 models, 5 seeds); encoding as in Figure 1, axes start at 0.2. LLMRouterBench has a fixed size; RouterEval's smaller pools appear only in the framework comparison of Figure 10. Declarations on the four recorded and simulated backends are calibrated to the live self-ratings (Appendix A). Whiskers: Figure 6.

LLMRouterBench pool the learned MLP router reaches 0.93 versus MIDIAN's 0.88 honest and 0.83 under the cartel (Section 5).

4.4 The advantage is largest under the cartel

Under the cartel MIDIAN is nearly unaffected—it moves by at most 0.014 on any live population, 0.01 of the oracle on Bernoulli, replay, and RouterEval, and 0.05 on LLMRouterBench (Figures 1 and 3)—while the declared argmax and MIDIAN without defenses lose accuracy. The declared argmax loses the description's contribution, falling by 0.04–0.13 live and by 0.08–0.16 of the oracle on the other backends. Every method that reads reports without auditing them is vulnerable to collusion: MIDIAN without defenses loses 0.03–0.05 live and 0.17 on replay, more than any other method in Figures 1–3 that reads no declaration. Random liars (not drawn) are the milder attack at small populations and a comparable one at large. At n=10^3 they cost the declared argmax 0.04, versus 0.10 for the cartel (gossip reputation: 0.11 versus 0.22; live cartel values not drawn). At 10^4 and 10^5 they cost it 0.15 and 0.06, versus 0.13 and 0.04 for the cartel, and MIDIAN without defenses loses 0.13 to random liars versus 0.05 to the cartel at 10^5. The defenses must be added audits first, then verification (Appendix B).

4.5 Frameworks improve when given MIDIAN's shortlist

The frameworks' selection primitives are adequate on a declared shortlist; their input is the limit. We therefore give each framework, unmodified, a different shortlist: the leaf cohort of the MIDIAN tree for the task's family, with the root's pick first (the MIDIAN shortlist; Figure 2 at n=10^5, Figure 7 across n). On the specialist population it beats each framework's default retriever in all 80 (framework, n, regime) cells (minimum margin $+0.047$) and each framework's best text retriever in 70 of them, including nine of ten frameworks at 10^5. The MIDIAN shortlist keeps its value with scale: in Figure 7 mean framework success on it stays at 0.57–0.59 from 10^2 to 10^5 when agents are honest, while the default and MiniLM shortlists decay (0.48 to 0.38 and 0.53 to 0.46; lifts over the default at 10^5: Figure 9). At every n it is also the best shortlist under the cartel (at 10^5, nine of the ten frameworks score higher on it than on the best text retriever, exact paired test $p = 0.01$). At a hundred thousand agents the declared top-k shortlist has a higher honest mean (0.62), but it has two flaws. It is a ceiling: the ten highest claims are not the ten most skilled agents, so no framework on it exceeds 0.66. And it is the channel the cartel attacks: inflated claims reduce it to 0.55. Mean framework success on the MIDIAN shortlist loses 0.015 under the same cartel, and the best framework on it reaches 0.67 in both regimes (the black dots of Figure 2), the highest single-framework score of any shortlist, 0.20 short of the oracle; the best text retriever leaves the best framework 0.24 short. Even so, no shortlist

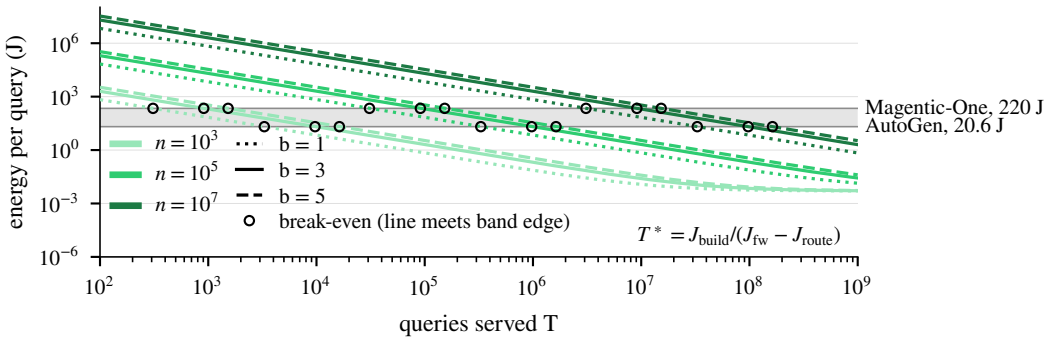

Figure 4: **MIDIAN routes far more cheaply than any framework once its build is repaid: after about ten queries per agent versus the cheapest framework, one versus the costliest; every later query costs under a hundredth of a joule.** Setup: each line is $J_{\text{build}}/T + J_{\text{route}}$ at n=10^3, 10^5, 10^7 (light to dark) and b=1, 3, 5 (dotted, solid, dashed), J_{build} the cell's measured build cost and J_{route} its per-query ledger energy (Appendix E); the band is the ten frameworks' per-query supervisor energy measured live at n=10^3 (20.6 J, AutoGen, to 220 J, Magentic-One). Each open circle marks a break-even point T^*=$J_{\text{build}}/(J_{\text{fw}}-J_{\text{route}})$ where a MIDIAN line crosses a band edge; beyond it MIDIAN is cheaper for every query. Energy is estimated from call counts.

closes the gap to MIDIAN itself: the best framework on any shortlist reaches 0.67 versus 0.84 for MIDIAN routing alone.

On RouterEval's pools of real LLMs the bottleneck is the framework rather than the shortlist: random already scores 0.53–0.54, and every shortlist lands within -0.04 to $+0.13$ of it. The MIDIAN shortlist still lifts the frameworks—it is the best shortlist under the cartel at a hundred models and within 0.07 of the best otherwise—but no shortlist brings a framework to MIDIAN's own level: the best framework on any shortlist reaches 0.67, while MIDIAN routing the whole pool reaches 0.74 at five thousand models versus an oracle of 0.92 (Figure 10).

4.6 Cost

MIDIAN's cost is a one-time build and a route that calls no model. The build is nFb probes plus audit re-runs (1.03–1.05×, 2.0×10^5 J at a thousand agents at b=3), with verification accounting for the $(b - b_0)/b$ share. Maintenance is charged per routed task: an update of $r\lceil\log_r n\rceil$ comparisons and $2 + d$ messages (5–9 mJ), and on five percent of tasks an audit of the routed outcome, about $0.05(r - 1)$ report messages and no probe (the task's own execution supplies the ground truth). An exclusion triggers a constant-size re-aggregation (at most n times over the tree's life) and calls no model. That is 25 to 80 messages and comparisons per routed task from 10^2 to 10^7 agents, versus n for every scan (Figure 11). With a probe at 4.0 J and a 7B supervisor call at 20.6 J, the route costs under a thousandth of one supervisor call; a framework spends one (AutoGen) to 10.7 (Magentic-One) per task at every n (Figure 4, Table 3). Break-even is linear in n and b: at b=3 the build is repaid after about ten tasks per agent versus the cheapest framework and one versus the costliest (Table 3).

5 Limitations and Conclusion

The tree's claim is accuracy at scale under a cartel: on the two small or dense pools of real LLMs a bandit or a learned router is more accurate (Section 4.3). No baseline here spends its probes adaptively; under honest reporters, elimination schedules (Audibert et al., 2010; Karnin et al., 2013) should improve on the tree. And every method here routes at the task-family level; routing below it would need probes and verification on individual prompts.

The agentic web is being built on self-descriptions, which stop working at a thousand agents, far short of the populations proposed for it (Raskar et al., 2025; Yang et al., 2025). A tree that probes, audits, and promotes on verified outcomes hands existing frameworks a better shortlist than they can retrieve. Routing in an open agent network should rest on verified outcomes, not on what agents claim; MIDIAN does so at a hundred thousand live agents, under collusion, for under a thousandth of a supervisor call's cost per task.

AI USE STATEMENT

Language models are the subject of this paper: the agents in the live benchmark, the supervisors inside the ten frameworks, and the dense retrievers and the reranker of Section 4.2 are language models, described in Section 4.1 and Appendix C. In producing the paper, we used generative AI tools to edit and smooth the language of text written by the authors and to draft boilerplate code (data loading, plotting, and experiment orchestration) under the authors' direction. We did not use generative AI tools to generate research ideas, to design the method or the experiments, or to analyze results. We have reviewed all AI-assisted work: every claim, number, and citation was checked by the authors against the result tables and primary sources, and AI-drafted code was reviewed and tested by the authors before use. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies routing among software agents and involves no human subjects and no personal data. Its adversary is a synthetic one: a fraction of agents that inflate their declared competence and collude in peer reports. As in adversarial machine learning for a single model, the defenses here are evaluated against the attacks we specify, not against every attack; other collusion strategies may fall outside them and require further tuning, which is why the liar model will be released as part of the benchmark. We release no attack tooling beyond that liar model. All task data come from public generators and benchmarks with their own licenses. This work adheres to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

Section 4.1 and Appendix A define the populations, channels, liar model, probe budget, seeds, and pairing; Section 3 and Appendix B give every hyperparameter of MIDIAN and of the eighteen baseline routers; Appendix C lists the ten frameworks at pinned versions, their selection primitives, the supervisor model and serving configuration, the fallback rule, and every retrieval variant with its exact instruction strings. The benchmark harness, method implementations, framework adapters, result tables, and the scripts that draw every figure from those tables will be released under an open license.

REFERENCES

A2A Project. Agent2Agent (A2A) protocol specification, version 1.0.0. https://a2a-protocol.org/latest/specification/, 2026. Linux Foundation project; defines the Agent Card self-description format. Accessed 2026-09-23.

Anthropic. Model Context Protocol specification (protocol revision 2026-07-28). https://modelcontextprotocol.io/specification/2026-07-28, 2026. Open specification; source at https://github.com/modelcontextprotocol/modelcontextprotocol. Accessed 2026-09-23.

Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *Proceedings of the 23rd Conference on Learning Theory (COLT)*, pp. 41–53, 2010.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256, 2002. doi: 10.1023/A:1013689704352.

Olivier Chapelle and Lihong Li. An empirical evaluation of Thompson sampling. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 24, 2011.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 758–759, 2009. doi: 10.1145/1571941.1572114.

CrewAI, Inc. CrewAI: Framework for orchestrating role-playing, autonomous AI agents. https://github.com/crewAIInc/crewAI, 2023. Accessed 2026-09-23.

A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):20–28, 1979. doi: 10.2307/2346806.

John R. Douceur. The Sybil attack. In *Peer-to-Peer Systems: First International Workshop on Peer-to-Peer Systems (IPTPS)*, Lecture Notes in Computer Science, pp. 251–260, 2002. doi: 10.1007/3-540-45748-8_24.

Tao Feng, Yanzhen Shen, and Jiaxuan You. GraphRouter: A graph-based router for LLM selections. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.03834.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Erkang Zhu, Friederike Niedtner, Grace Proebsting, et al. Magentic-One: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.

Xi Alice Gao, James R. Wright, and Kevin Leyton-Brown. Incentivizing evaluation with peer prediction and limited access to ground truth. *Artificial Intelligence*, 275:618–638, 2019. doi: 10.1016/j.artint.2019.03.004.

Luis Garicano. Hierarchies and the organization of knowledge in production. *Journal of Political Economy*, 108(5):874–904, 2000. doi: 10.1086/317671.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Brian P. Gerkey and Maja J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004. doi: 10.1177/0278364904045564.

Google. Agent Development Kit (ADK). https://github.com/google/adk-python, 2025. Accessed 2026-09-23.

Florian Grötschla, Luis Müller, Jan Tönshoff, Mikhail Galkin, and Bryan Perozzi. AgentsNet: Coordination and collaborative reasoning in multi-agent LLMs. In *ICML 2025 Workshop on Multi-Agent Systems in the Era of Foundation Models*, 2025. arXiv:2507.08616.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021. arXiv:2009.03300.

Ralf Herbrich, Tom Minka, and Thore Graepel. TrueSkill: A Bayesian skill rating system. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 19, pp. 569–576, 2006.

Chien-Ju Ho, Shahin Jabbari, and Jennifer Wortman Vaughan. Adaptive task assignment for crowdsourced classification. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, volume 28 of *Proceedings of Machine Learning Research*, pp. 534–542, 2013.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, et al. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2308.00352.

Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Ziyu Ye, Bowei Xia, Tao Sun, Zhaoxuan Jin, et al. OWL: Optimized workforce learning for general multi-agent assistance in real-world task automation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2505.23885.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. RouterBench: A benchmark for multi-LLM routing system. In *ICML 2024 Workshop on Agentic Markets*, 2024. arXiv:2403.12031.

Zhongzhan Huang, Guoming Ling, Yupei Lin, Yandong Chen, Shanshan Zhong, Hefeng Wu, and Liang Lin. RouterEval: A comprehensive benchmark for routing LLMs to explore model-level scaling up in LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 3860–3887, 2025. doi: 10.18653/v1/2025.findings-emnlp.208. arXiv:2503.10657.

Hugging Face. smolagents: A barebones library for agents that think in code. https://github.com/huggingface/smolagents, 2024. Accessed 2026-09-23.

Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007. doi: 10.1016/j.dss.2005.05.019.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th International Conference on World Wide Web (WWW)*, pp. 640–651, 2003. doi: 10.1145/775152.775242.

Zohar Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1238–1246, 2013.

Zixuan Ke, Yifei Ming, Austin Xu, Ryan Chin, Xuan-Phi Nguyen, Prathyusha Jwalapuram, Jiayu Wang, Semih Yavuz, Caiming Xiong, and Shafiq Joty. MAS-Orchestra: Understanding and improving multi-agent reasoning through holistic orchestration and controlled benchmarks. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2601.14652.

Leonard Kleinrock and Farouk Kamoun. Hierarchical routing for large networks: Performance evaluation and optimization. *Computer Networks*, 1(3):155–174, 1977. doi: 10.1016/0376-5075(77)90002-2.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165. arXiv:2309.06180.

LangChain, Inc. LangGraph. https://github.com/langchain-ai/langgraph, 2023. Accessed 2026-09-23.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023. arXiv:2303.17760.

Hao Li, Yiqun Zhang, Zhaoyan Guo, Chenxu Wang, Shengji Tang, Qiaosheng Zhang, Yang Chen, Biqing Qi, et al. LLMRouterBench: A massive benchmark and unified framework for LLM routing. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 37733–37754, 2026. doi: 10.18653/v1/2026.findings-acl.1881. arXiv:2601.07206.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pp. 661–670, 2010. doi: 10.1145/1772690.1772758.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic LLM-powered agent network for task-oriented agent collaboration. In *Conference on Language Modeling (COLM)*, 2024. arXiv:2310.02170.

LlamaIndex. LlamaIndex. https://github.com/run-llama/llama_index, 2022. Accessed 2026-09-23.

Yu. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020. doi: 10.1109/TPAMI.2018.2889473. arXiv:1603.09320.

Microsoft. Microsoft Agent Framework. https://github.com/microsoft/agent-framework, 2025. Accessed 2026-09-23.

Kanghua Mo, Li Hu, Yucheng Long, and Zhihao Li. Attractive metadata attack: Inducing LLM agents to invoke malicious tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2508.02110.

Alberto Montresor. A robust protocol for building superpeer overlay topologies. In *Proceedings of the 4th International Conference on Peer-to-Peer Computing (P2P)*, pp. 202–209, 2004. doi: 10.1109/PTP.2004.1334948.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs with preference data. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2406.18665.

OpenAI. OpenAI Agents SDK. https://github.com/openai/openai-agents-python, 2025. Accessed 2026-09-23.

Abdelghny Orogat, Ana Rostam, and Essam Mansour. Architectural design, not only model intelligence, governs multi-agent LLM performance. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2027. Introduces MAFBench. arXiv:2602.03128; v1 title: "Understanding Multi-Agent LLM Frameworks: A Unified Benchmark and Experimental Analysis".

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. arXiv:2404.13076.

Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, et al. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*, 2024.

Ramesh Raskar, Pradyumna Chari, John Zinky, Mahesh Lambe, Jared James Grogan, Sichao Wang, Rajesh Ranjan, Rekha Singhal, et al. Beyond DNS: Unlocking the internet of AI agents via the NANDA index and verified AgentFacts. *arXiv preprint arXiv:2507.14263*, 2025.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019. doi: 10.18653/v1/D19-1410.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019.

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

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning Gym: Reasoning environments for reinforcement learning with verifiable rewards. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2505.24760.

William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3–4):285–294, 1933. doi: 10.1093/biomet/25.3-4.285.

Zihan Wang, Rui Zhang, Yu Liu, Wenshu Fan, Wenbo Jiang, Qingchuan Zhao, Hongwei Li, and Guowen Xu. MPMA: Preference manipulation attack against model context protocol. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(42):35838–35846, 2026. doi: 10.1609/aaai.v40i42.40898. arXiv:2505.11154.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In *First Conference on Language Modeling (COLM)*, 2024. arXiv:2308.08155.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can LLMs express their uncertainty? An empirical evaluation of confidence elicitation in LLMs. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2306.13063.

Beverly Yang and Hector Garcia-Molina. Designing a super-peer network. In *Proceedings of the 19th International Conference on Data Engineering (ICDE)*, pp. 49–60, 2003. doi: 10.1109/ICDE.2003.1260781.

Yingxuan Yang, Mulei Ma, Yuxuan Huang, Huacan Chai, Chenyu Gong, Haoran Geng, Yuanjian Zhou, Ying Wen, et al. Agentic Web: Weaving the next web with AI agents. *arXiv preprint arXiv:2507.21206*, 2025.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. Do large language models know what they don't know? In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8653–8665, 2023. doi: 10.18653/v1/2023.findings-acl.551. arXiv:2305.18153.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, et al. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

Yiqun Zhang, Hao Li, Chenxu Wang, Linyao Chen, Qiaosheng Zhang, Peng Ye, Shi Feng, Xinrun Wang, et al. The Avengers: A routing recipe for collective intelligence in language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(41):34870–34878, 2026. doi: 10.1609/aaai.v40i41.40790. arXiv:2505.19797 (preprint title: "The Avengers: A Simple Recipe for Uniting Smaller Language Models to Challenge Proprietary Giants").

Hang Zheng, Hongshen Xu, Yongkai Lin, Shuai Fan, Lu Chen, and Kai Yu. DiSRouter: Distributed self-routing for LLM selections. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.19208.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 36, 2023. arXiv:2306.05685.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. EmbedLLM: Learning compact representations of large language models. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.02223.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language agents as optimizable graphs. In *International Conference on Machine Learning (ICML)*, 2024. GPTSwarm; arXiv:2402.16823.

A BENCHMARK DETAILS

Agents on each backend. An agent is an integer identifier in a population of n; a task belongs to exactly one of F families. On the *live* backend an agent is a profile: a model from the ladder Qwen2.5 0.5B, 1.5B, 3B, 7B, and 14B or Gemma-2 2B and 9B, served by vLLM at temperature zero (Kwon et al., 2023); a set of three specialty families drawn at random; and a Python tool for models of 3B and above. This is the *specialist* shape, used throughout unless stated otherwise; two further live shapes are *heavy-tail* (ten percent of agents run a random large model specialized in every family and ninety percent a random small model with no specialty) and *bimodal* (twenty percent run Qwen2.5-7B specialized in every family and eighty percent Qwen2.5-0.5B with no specialty). Behavior on a family is determined by the per-request signature $\langle$model, specialized in that family or not, tool, token limit$\rangle$, the projection of the agent's $\langle$model, specialties, tool$\rangle$ signature of Section 4.1 onto one family, so there are at most fourteen distinct behaviors per family whatever n is (seven models, specialized in the family or not; tool and token limit are fixed by the model). Every request is memoized on this signature: the first agent with a given signature executes an instance and every later same-signature agent receives the stored output, so a population of 10^5 agents costs no more model calls than fourteen behaviors per family require, which is what makes 10^5 affordable; same-signature agents are therefore exact clones. On the *calibrated Bernoulli* backend an agent is a row of the skill matrix resampled with replacement from the measured live specialist matrix (n=10^3, 887 distinct rows, mean skill 0.419), and an outcome is a coin flip with that skill. On *replay* an agent is one of RouterBench's eleven models with a per-category mask; a masked category executes as that category's weakest model, execution is a lookup of the recorded outcome, and each category's prompts are split 70/30 between probes and tasks. On *RouterEval* and *LLMRouterBench* an agent is one real LLM from a fixed pool of recorded per-prompt scores; the pool, its skill, and the low-skill-first liar set are the same in every seed; each test prompt is routed at most once and the agent order is shuffled per seed. RouterEval's pools are its own hard-setting pools of $m \in \{10, 10^2, 10^3\}$ leaderboard models in the compositions *strong-to-weak* (mixed ability; the default in Figures 3 and 10), *all-strong*, and *all-weak*, plus the full 5,000-model leaderboard.

Skill and its measurement. S_{af} is the Bernoulli probability, the measured live accuracy, or the recorded accuracy of agent a on family f. On the live backend it is measured per signature on a fixed set of 200 instances per family (60 for the two largest models), disjoint from task and probe instances; a population is admitted when between-agent variance exceeds the binomial floor by a factor of 1.5 per family. Only the oracle, the low-skill-first liar selection, and offline diagnostics read S.

Task families and scoring. Live tasks come from sixteen Reasoning Gym generators: basic arithmetic, chain sum, letter counting, syllogism, family relationships, gcd, lcm, prime factorization, number sorting, simple equations, count bits, number format, time intervals, word sequence reversal, binary alternation, and calendar arithmetic (reasoning-gym 0.1.19, arithmetic families capped at three terms of two digits). The agent's answer is scored by the generator's verifier and counts as solved at a score of at least 0.99. Replay uses the 64 largest RouterBench categories with scores binarized at 0.5. RouterEval uses the sixteen MMLU subjects with the most training prompts; probes read training prompts and tasks read test prompts, so the oracle is the best model on the training split and can occasionally be exceeded on a finite test stream. LLMRouterBench routes over twenty of its open 7–9B models on fifteen of its datasets with a 70/30 split. Family demand is uniform. Each unit routes Q=1,000 tasks, or 300 at live $n \geq 10^4$, on RouterEval, and on LLMRouterBench.

Probes and budget. The j-th probe of (a, f) always maps to the same instance and indices are reset before each method. The build budget is nFb probes with $b \in \{1, 3, 5\}$ (Section 4.1); only MIDIAN's audit reruns exceed it (Appendix E).

Declarations. Honest live declarations are self-ratings: each signature's model is asked what fraction of problems of the family it solves correctly, given the family description and an example; an unparseable answer becomes 0.5. On the specialist population at n=10^3 it takes eleven distinct values, has mean 0.686 versus a true mean of 0.419, and correlates with S at 0.36. The 70-word paragraph of Section 4.1 also carries a clause listing the agent's true specialty families; the paragraph is generated once per population and is unaffected by the liar model, so text-retrieval shortlists are identical across liar regimes and the lie reaches a text-routing framework only through the fallback

rule (Appendix C). On the other backends a declaration is drawn independently per agent and family from the live self-ratings' empirical distribution over their eleven values given the decile of true skill, fitted on the live specialist populations; the draws reproduce the live mean (0.69) and correlation with S (0.35–0.36). The fit uses the live specialist populations only and is applied regardless of population shape, so on replay's heavy-tail and bimodal pools the claims are less informative than live self-ratings would be.

Liars. There are $\mathrm{round}(\beta n)$ liars, chosen uniformly or as the agents with the lowest mean true skill. The declared row becomes $\mathrm{clip}(D + 0.4, 0, 1)$; the "strongest fifth" of Section 3 is ranked by the liar's own observed means, and every other honest agent is reported truthfully. Collusion in reports is enabled in every grid; "cartel" denotes β=0.5, low-skill-first selection. On the live specialist population at n=10^3 the liar half has mean skill 0.27 versus 0.57 for the honest half, and a specialist liar's mean declaration after inflation is about 0.88.

Seeds, pairing, and intervals. A seed redraws the population, declaration noise, random liar set, task stream, probe instances, and every method's random stream from a stable hash. Within a (cell, seed) unit the runner builds one world and one stream, runs the oracle, and resets the world before each method, so all methods face the same agents, liars, declarations, tasks, and probe outcomes. The unit of replication is the seed; rows are averaged per seed, the mean of seed means is reported, and every whisker is ± 1 standard error of that mean over seeds, about a 68% interval; with three seeds it rests on two degrees of freedom. In Figure 3 every bar and interval of a backend is divided by that backend's honest b=3 oracle mean.

B Methods and hyperparameters

MIDIAN and its ablations. MIDIAN without defenses: r=10, δ=1/3, online updates on; the leaf partition and the grouping of cohorts into parents are uniformly random at build; per-(member, family) reports are pooled and trimmed by $\max(0, \min(\lfloor \delta(r-1) \rfloor, ((r-1)b-1)/2))$ single reports per side, three per side at every b; a singleton cohort uses its own probe mean; ties in a node's maximum go to the lowest child slot; the online running mean starts with count $w_0 = (r-1)b - 2 \cdot \mathrm{trim}$ so that the build estimate is weighted as that many observations. MIDIAN w/o audits adds verification at promotion and a cached root pick, and trims per peer rather than per report. MIDIAN w/o verification adds the audits of Section 3 (re-run probability 0.05 on the same instance; a report that disagrees with the true outcome is a strike). MIDIAN itself combines both (Section 3), with the verification remainder $nF(b-b_0)$ spread over the C candidates forwarded above the leaves at $e = \lfloor (b-b_0)n/C \rfloor$ probes each, trimmed per reporter; online, a routed outcome updates the path and is audited with probability 0.05 through the cohort's report channel. None of r, δ, the audit rate, the two-strike rule, or b_0 was tuned on reported seeds.

Why audits come first. Verification before audits re-confirms a corrupted candidate with the same corrupted reporters and loses accuracy, whereas audits first and verification second gain at every population from 10^3 to 10^7. On the live specialist population at n=10^3 under the cartel, adding audits to MIDIAN without defenses gains $+0.053$, and adding verification afterwards a further $+0.016$; adding verification first loses 0.025 and adding audits afterwards recovers $+0.094$. Two exceptions: at n=10^2 MIDIAN w/o verification (0.777) edges MIDIAN (0.768), and at n=10^5 under the cartel, adding verification after audits gains $+0.079$, so at scale it buys accuracy as well as a cheaper route.

Baseline routers. *Oracle*: the agent with the highest true skill in the task's family. *Flat probe argmax (online)*: every agent probed b times per family by a trusted observer, argmax of the probe mean with ties to the lowest id, running-mean update on the routed agent; after b probes estimates lie on $\{0, 1/b, \ldots, 1\}$, so at n=10^5 and b=1 about 30,000 agents tie at the top. *Declared argmax*: argmax of D in the family. *Random*. *UCB1* per family with c=$\sqrt{2}$ (Auer et al., 2002); *Thompson* per family with Beta(1,1) priors plus the probes (Thompson, 1933; Chapelle & Li, 2011); *warm-start bandit*: Thompson with prior $\alpha_0 = n_0 D$, $\beta_0 = n_0(1-D)$, n_0=5, so declarations are worth five pseudo-observations versus three probes, and a variant with n_0=0.5, tuned on held-out seeds in one honest live cell; *LinUCB* (Li et al., 2010) per family, a ridge model over the context (1, mean on f, $\sqrt{\text{count}}$, mean over families) with α=1, whose exploration bonus is computed from the agent's own intercept and observation count on the family alone, so that agents tied at the top estimate are not

separated by how atypical their cross-family mean is; it never uses identity, specialty, or declarations; *TrueSkill* per family from about $nb/2$ random pairs drawn with replacement, self-pairs dropped, each member probed on its own instance, about nb probes in all (Herbrich et al., 2006). *kNN* and *kNN-online*: RouterBench's kNN router on our budget, in which the predicted success of a is the mean outcome of a's b nearest probes by cosine similarity on MiniLM embeddings (the backend's embeddings on RouterEval); *MLP*: one regressor on [agent one-hot $\oplus$ prompt embedding] fit on the nFb probes, quadratic in n and therefore run only at $n \leq 10^3$ live and on pools below 5,000; *flat-NSW*: flat probe means in a hierarchical navigable small-world (HNSW) inner-product index (Malkov & Yashunin, 2020); *cluster-head*: k-means on D with the highest-declaring member as head, a two-level super-peer hierarchy over declarations (Yang & Garcia-Molina, 2003); *DiSRouter cascade*: agents are ordered by ascending mean declaration, and the first whose declared skill in the family exceeds 0.7 takes the task (Zheng et al., 2026). The last two read only declarations and spend no probes. No adaptive elimination schedule is among the baselines; every probe-based baseline spends its budget non-adaptively. Five further methods are not drawn: *verify-on-claim* (route by declared claim, then re-probe the candidate before committing, about 2.4 probes per routed task outside the build budget), *gossip reputation* (greedy routing on gossiped peer reputations), *contract-net self-bid* (Smith, 1980), *referral network*, and *route-to-k majority*, which executes three agents per task and is therefore outside the one-execution budget. Their success on every backend is in Table 1. On ten million Bernoulli agents and a million replayed RouterBench agents MIDIAN beats all five in both regimes, the closest being verify-on-claim, 0.05–0.14 behind. Gossip reputation is strong when agents are honest and drops sharply under the cartel; on the RouterEval leaderboard its honest score exceeds MIDIAN's, and verify-on-claim, nearly flat across regimes, edges MIDIAN by 0.01 on LLMRouterBench under the cartel.

Table 1: Success of the five undrawn arms and MIDIAN at b=3, honest / cartel (live n=10^3 honest only), with the oracle of each backend. Bernoulli: 100 seeds; replay: 30 seeds $\times$ 3 shapes; RouterEval: 3 seeds; LLMRouterBench: 5 seeds; live: n=10^3, seeds 1–5, honest.

	live 10^3	Bernoulli 10^7	replay 10^6	RouterEval 5,000	LLMRouterBench
oracle	0.861	0.846	0.792	0.920	0.750
MIDIAN	0.813	0.801 / 0.794	0.765 / 0.762	0.739 / 0.733	0.663 / 0.626
verify-on-claim	0.722	0.754 / 0.722	0.668 / 0.625	0.652 / 0.636	0.647 / 0.635
gossip reputation	0.666	0.671 / 0.309	0.366 / 0.244	0.784 / 0.543	0.663 / 0.425
contract-net self-bid	0.602	0.575 / 0.448	0.293 / 0.214	0.597 / 0.517	0.590 / 0.503
route-to-k majority	0.595	0.612 / 0.439	0.266 / 0.175	0.614 / 0.574	0.592 / 0.487
referral network	0.556	0.555 / 0.393	0.417 / 0.224	0.592 / 0.521	0.589 / 0.511

"Best learned/declared router" and "best bandit" in Figures 1–3 are cross-fitted: for each seed, the pool member with the highest mean on the other seeds is chosen and its score on that seed is recorded, so no bar is a maximum of noisy means. The learned/declared pool is kNN, kNN-online, MLP, flat-NSW, cluster-head, and the DiSRouter cascade; the bandit pool is UCB1, Thompson, warm-start at both n_0, LinUCB (the fixed-bonus variant only), and TrueSkill. Not every member runs in every cell: TrueSkill stops below n=10^5, kNN and MLP need prompt text and so skip Bernoulli and replay, and MLP runs only below n=5,000; a pool takes the best of the members that ran. At b=3 the learned pick is kNN-online on the live backend except at 10^4, where it is the flat-NSW index; the flat-NSW index on Bernoulli, replay, and RouterEval; and the MLP router on LLMRouterBench. The bandit pick is LinUCB at live 10^2, on replay, RouterEval, and LLMRouterBench, and the warm-start bandit (n_0=5) on live 10^4–10^5 under the cartel and on Bernoulli.

C Frameworks and shortlists

The framework pipeline. A framework method never routes over the whole population. At build time it indexes the agents' self-descriptions. Per task it retrieves a shortlist of k=10 (ordered best-first, ties to the lowest id; outside the MIDIAN-shortlist mode the shortlist depends only on the family), sends the task text and the shortlist's name–description pairs to the framework library running in its own virtual environment as a subprocess with a fresh team per request and no memory, lets the library's own delegation primitive name one agent with Qwen2.5-7B-Instruct as supervisor at temperature zero, stops before the named agent would execute, and maps the name back to an id. The

framework receives no probes, reports, outcomes, or declared numbers (on the live backend), and no true skill or liar labels. A valid pick is a known name in the shortlist; anything else (the supervisor answering the task itself, returning no choice, naming a non-candidate, or calling a tool named after an agent instead of the routing tool) routes to the declared argmax within the shortlist and is scored leniently as that agent's outcome; a strict metric scores every non-pick as a failure. Both are in Table 2 for every framework in one cell: the two differ only where non-picks are common, the two lenient leaders are the two frameworks that most often fail to name an agent, and the best framework under strict scoring is the Microsoft Agent Framework or AutoGen at 0.47–0.51. Infrastructure errors are retried once, and a unit in which more than $\max(3, 2\%)$ of requests fail writes no row, so a broken environment produces no framework row. Supervisor picks are as reproducible as greedy decoding under varying batch composition on a vLLM fleet, which is not bit-exact.

Table 2: Lenient and strict success and non-pick rate per framework on the live specialist population at n=10^3 with the MiniLM shortlist, honest, b=3 (10 seeds). Lenient routes a non-pick to the declared argmax of the shortlist; strict counts it as a failure.

framework	lenient	strict	non-pick
AutoGen	0.471	0.471	0.000
CAMEL Workforce	0.490	0.489	0.001
CrewAI	0.481	0.480	0.001
Google ADK	0.555	0.311	0.338
LangGraph	0.490	0.453	0.056
LlamaIndex	0.494	0.474	0.027
Microsoft Agent Framework	0.525	0.511	0.015
Magentic-One	0.632	0.129	0.687
OpenAI Agents SDK	0.485	0.458	0.042
smolagents	0.466	0.411	0.056

The ten frameworks. LangGraph (langgraph 1.2.11, langgraph-supervisor 0.0.31): supervisor graph; the pick is the first transfer tool call. CrewAI (1.15.18): hierarchical crew with a manager. AutoGen (autogen-agentchat 0.7.5): `SelectorGroupChat`, the first speaker-selection event; its own selector falls back to the first participant after three failed attempts, which counts as a pick. Magentic-One (from the AutoGen distribution): one-turn ledger; the pick is read from "Next Speaker"; a 14B-orchestrator variant is run at $n \leq 10^3$. Microsoft Agent Framework (agent-framework-core 1.16.0): router orchestrator, one round. OpenAI Agents SDK (openai-agents 0.22.0): triage agent with one handoff per candidate. Google ADK (google-adk 2.8.0): router agent with sub-agents. LlamaIndex (llama-index-core 0.14.24): one single-selector call. smolagents (1.26.0): manager with managed agents as tools, one step. CAMEL Workforce (camel-ai 0.2.90): one worker per candidate; the assignment callback records the pick. The 14B-orchestrator result of Section 4.2: on live populations at n=10^3 across three population shapes and four liar fractions, the 14B orchestrator cuts Magentic-One's non-pick rate from 60% to 43% and raises strict success from 0.17 to 0.29, yet lowers lenient success by 0.013, tying the 7B arm on the bimodal and heavy-tail shapes and losing on every specialist cell.

Shortlist variants. *Hashed TF-IDF (default)*: tokens hashed into 4,096 buckets, IDF fitted on descriptions and family queries, cosine against the family description, no de-duplication; on the live backend, same-signature agents have identical descriptions, so the stable sort fills the top ten with clones: about 4.4 distinct signatures per shortlist at 10^3, 2.6 at 10^4, and exactly one at 10^5. Every other variant de-duplicates descriptions first. *BM25* (k_1=1.5, length-normalization parameter 0.75) (Robertson & Zaragoza, 2009). *MiniLM*: cosine between all-MiniLM-L6-v2 embeddings (Reimers & Gurevych, 2019). *Qwen3-8B dense*: cosine between Qwen3-Embedding-8B embeddings (Zhang et al., 2025) with the model's stock query prefix ("Given a web search query, retrieve relevant passages that answer the query"); the *I-competent* and *I-demonstrated* variants replace it with "Given a task family, retrieve agents that are competent at solving tasks of that family" and "…agents whose demonstrated skill at that family is highest". *Fusion + reranker*: reciprocal-rank fusion of BM25 and dense (Cormack et al., 2009), top fifty, reranked by Qwen3-Reranker-4B as a cross-encoder on (family description, agent description), top ten kept; the instruction variants change only which fifty reach the reranker. *Declared top-k*: the ten highest declared competences for the family, no text retrieval, the idealization of agent-card discovery in which an orchestrator shortlists by published claims. *MIDIAN*

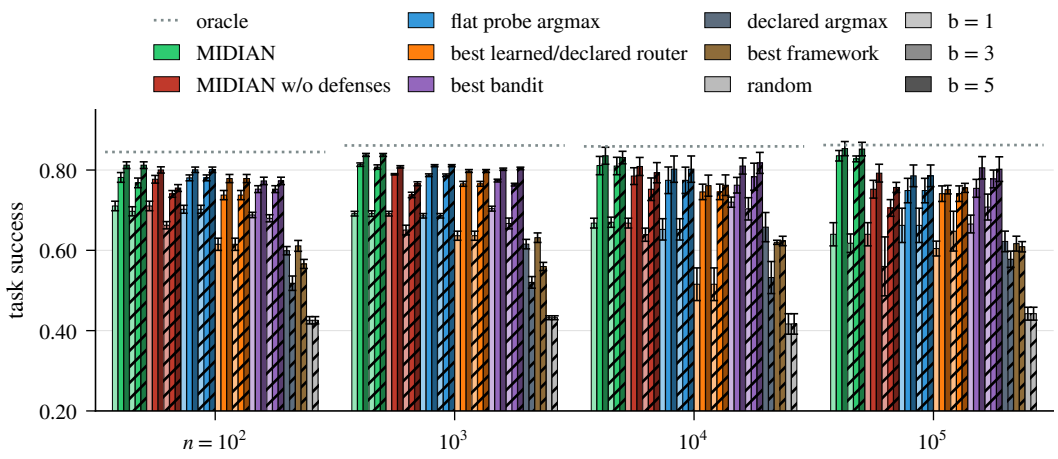

Figure 5: **MIDIAN's lead on live populations appears at b=3 and widens at b=5; at b=1, with no verification budget, it is an ordinary probe scan, and the warm-start bandit passes it from 10^3 to 10^5 and the flat probe argmax at 10^5.** Setup as in Figure 1, each arm at each budget drawn as its own bar (light, mid, dark = b=1, 3, 5) with ± 1 s.e. whiskers over seeds.

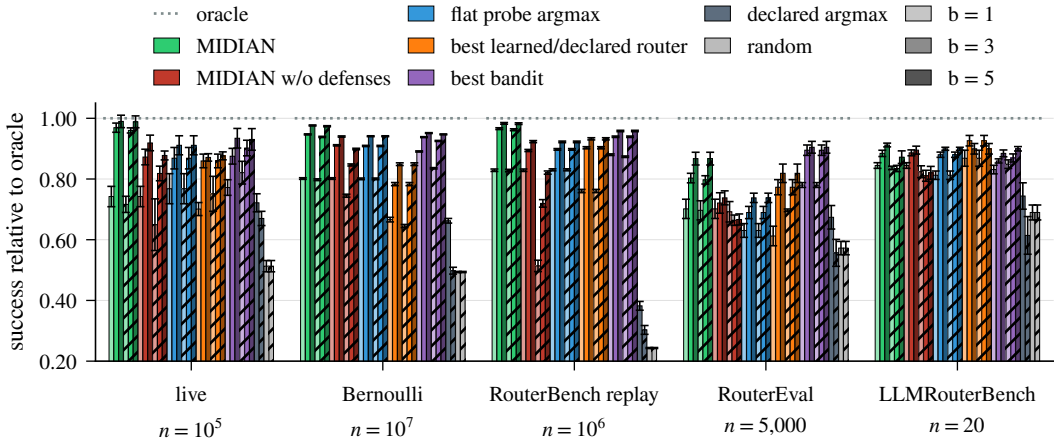

Figure 6: **MIDIAN's lead on the large populations needs a verification budget: at b=3 and b=5 it holds on live, Bernoulli, and replay, and at b=1 the warm-start bandit passes it on Bernoulli and replay.** Setup as in Figure 3, each budget drawn as its own bar with ± 1 s.e. whiskers over seeds; every bar and whisker of a backend is divided by that backend's honest b=3 oracle mean.

shortlist: a full MIDIAN instance built on the same budget; the shortlist is the root's cached pick for the family followed by the other members of that pick's leaf cohort; the framework's chosen outcome feeds back to the tree, which keeps auditing online. At n=10^5 with half the population lying at random, 74% of the declared top-k consists of liars (41% under the cartel, whose weakest agents mostly miss the top ten even inflated by 0.4); the MIDIAN shortlist holds liars at their base rate (49–53%), which is immaterial because its order is set by verified outcomes.

Framework sets per bar. The bars of Figures 2 and 7–10 are means over the frameworks present in that regime of a shortlist. A framework enters a bar only with complete seeds under that regime, and the honest and cartel sets can differ by one framework; on the live backend every bar has nine or ten of the ten frameworks, and on RouterEval, where Magentic-One is excluded by design, six to nine. These differences can move a mean by a few hundredths; none changes an ordering discussed in the text.

D ADDITIONAL FIGURES

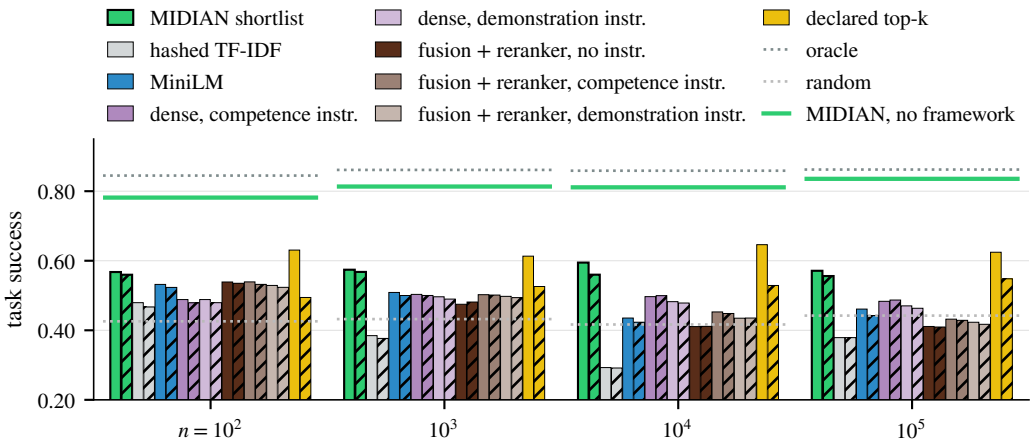

Figure 7: **MIDIAN's shortlist keeps its value from a hundred agents to a hundred thousand while the default and MiniLM shortlists decay.** Mean framework success on it stays at 0.57–0.59 when agents are honest, and it is the best shortlist under the cartel at every n. Setup: frameworks by shortlist across n on the live backend (specialist, b=3; at $n \leq 10^3$ 10 seeds, or 3 for declared top-k and the instruction variants of dense and reranker; 3 seeds at $n \geq 10^4$); bars: unweighted mean over frameworks of each framework's seed-mean success, nine or ten frameworks per bar, solid honest, hatched cartel; dotted: oracle; green: MIDIAN routing the whole population. The default hashed TF-IDF shortlist decays with n and is a clone floor at 10^5.

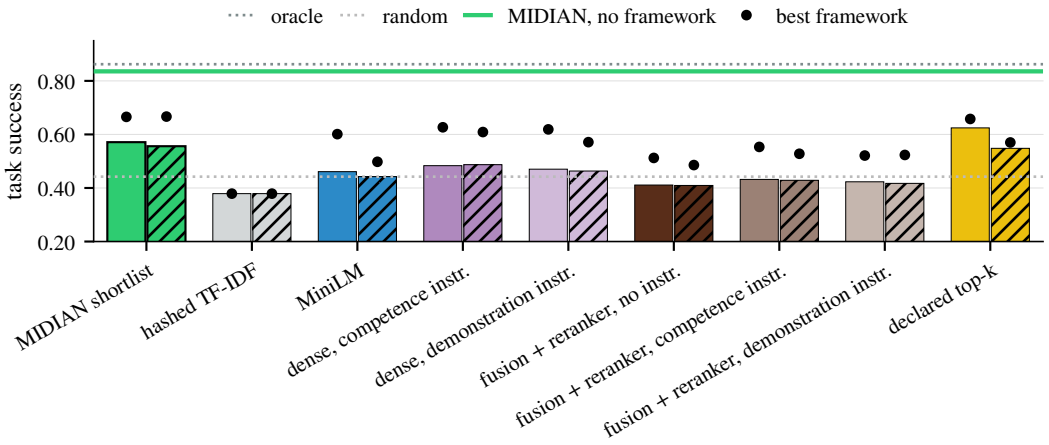

Figure 8: **Retrieval instructions move a shortlist by hundredths and change no ordering.** Figure 2 with the fusion-plus-reranker stack shown with no instruction, the competence instruction, and the demonstration instruction, and dense retrieval with the two instructions; setup as in Figure 2.

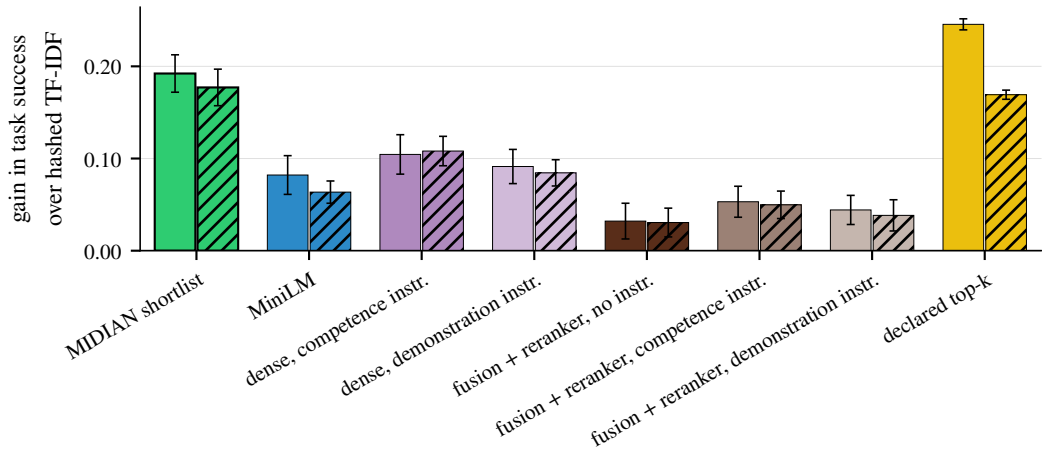

Figure 9: **Relative to the default shortlist, MIDIAN lifts frameworks by 0.19 and the declared top-k list by 0.25; text retrievers add at most 0.11.** Setup: live, specialist, n=10^5, b=3, 3 seeds; each bar is shortlist minus hashed TF-IDF averaged over the nine or ten frameworks present under both shortlists; solid honest, hatched cartel; whiskers are ± 1 s.e. across frameworks, a between-framework spread rather than seed uncertainty. TF-IDF here is a constant (ten clones of one agent), so the lift includes the gain from de-duplication; the declared top-k whisker is several times narrower than the others because, given the top ten by claim, every framework makes the same choice.

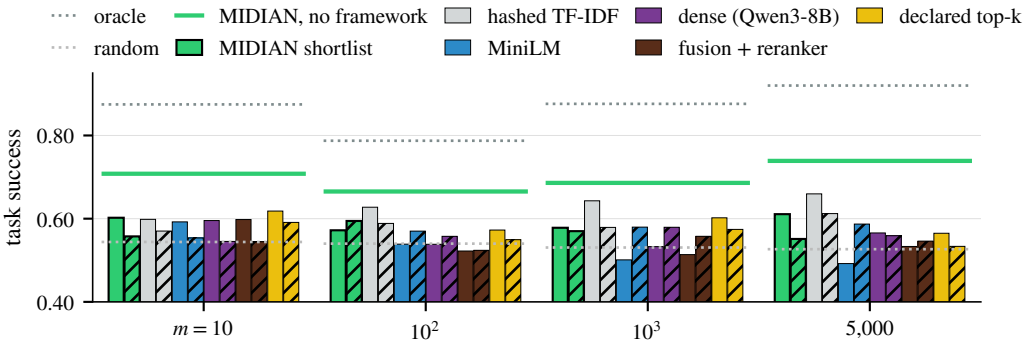

Figure 10: **On pools of real LLMs no shortlist rescues a framework: the best framework on any shortlist (0.67) stays below MIDIAN routing alone (0.74).** The MIDIAN shortlist is the best under the cartel at a hundred models and within 0.07 of the best elsewhere. Setup: Figure 2 repeated with the same frameworks and shortlists on RouterEval's strong-to-weak pools of n=10, 10^2, and 10^3 LLMs (5 seeds) and the 5,000-LLM leaderboard (3 seeds), b=3, where a description is the agent's claim vector rendered as text and the supervisor sees only the task family; bars: mean over frameworks (at most nine; Magentic-One excluded by design), solid honest, hatched cartel; dense and reranker at their best instruction per pool; upper dotted line: oracle; lower: random; green: MIDIAN on the whole pool; axes start at 0.4. At n=10 every shortlist returns the whole pool, so that group is the frameworks with no shortlist at all. Hashed TF-IDF over the rendered claims is the best honest shortlist from a hundred models upward, and the MIDIAN shortlist's cartel penalty is at most 0.06.

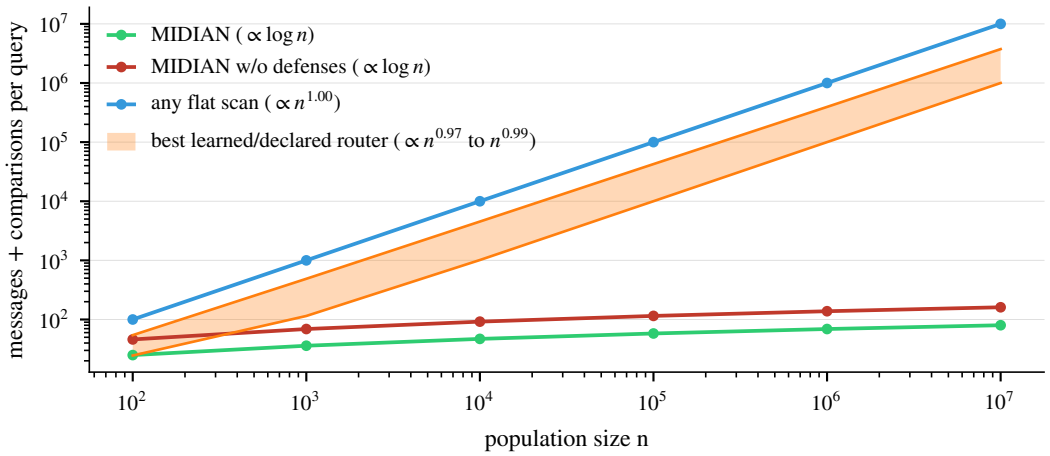

Figure 11: **Routing work grows with the depth of the tree, as $\log n$, while every scan grows as n.** MIDIAN does 25 to 80 messages and comparisons per query from $n{=}10^2$ to 10^7, five orders of magnitude below a flat scan at ten million agents. Setup: messages plus comparisons the ledger charges per routed query on calibrated Bernoulli populations at $b{=}3$ (exact counts, medians over 100 seeds per cell; the build is not included); flat probe argmax, declared argmax, and every bandit coincide at exactly n (one line); the learned-router band runs from the cluster-head router (about $n/10$) to the DiSRouter cascade (about $0.37n$). The flat-NSW index, a pool member with a constant 50 comparisons, is not drawn; random does no routing work.

E Cost accounting

What the ledger counts. Per routed task the ledger charges messages and comparisons; Figure 11 is their sum; it omits forwarding steps (d for MIDIAN without defenses, none for MIDIAN), audit reports (about $0.05(r-1)$ per task for MIDIAN), and the build. With depth $d = \lceil \log_r n \rceil$ and $r{=}10$: MIDIAN sends $2 + d$ messages and makes $1 + 10d$ comparisons; MIDIAN without defenses $3d$ and $20d$; flat probe argmax, declared argmax, and every bandit (UCB1, Thompson, warm-start, LinUCB, TrueSkill) send nothing and compare n; the cluster-head router sends 4 and compares about $n/10 + 10$; the DiSRouter cascade sends messages only, about $0.37n$ at 10^7, and makes no comparisons; the flat-NSW index compares $\mathrm{ef}{=}50$ at every n and its $\lceil \log_2 n \rceil$ forwarding steps are not counted; kNN and MLP compare n where they run. A framework sends $k + 2{=}12$ messages and compares $k{=}10$ per task, whatever n is, plus its supervisor calls. The counts are identical across seeds for every method except the cluster-head router and the cascade, whose medians are drawn.

Energy model. GPU-seconds for one model call are $p\,(A \cdot \text{prompt tokens} + B \cdot \text{generated tokens})$ with p the parameter count in billions and $B{=}5A$ (a decode token taken as five prefill tokens). A is calibrated so that a 7B supervisor call of 1,900 prompt and 65 generated tokens costs $1/34$ GPU-s, the throughput of a saturated single-GPU 7B replica; at 700 W that call is 20.6 J. A probe is one call by the probed agent's own model at that model's measured mean token counts (the 7B and 14B models use the 3B means); the mix-weighted mean over the specialist population's seven models is 0.0058 GPU-s, 4.04 J (heavy-tail 1.23 J, bimodal 1.42 J). A message is 10^{-3} J, an RPC of about 100 μs on a 10 W core; a comparison 10^{-8} J. The routed task's own execution is excluded; it is the same for every arm. A framework's per-task energy is its mean live latency relative to AutoGen's single call (1.90 s) times that call's energy, so "call-equivalents" are a latency ratio; latency is measured at $n{=}10^3$ over ten seeds, three population shapes, and four liar fractions on a shared vLLM fleet, and only the band's two edges enter Figure 4. The uncertain parts are the 5:1 decode ratio, the single throughput calibration, latency standing in for GPU work, and 700 W as the draw; a 400 W draw scales every model-call term by $4/7$ and leaves the break-even almost unchanged, since build and band scale together, and charging a message at 10^{-2} J moves it by under 0.3%.

Table 3: Cost at $n{=}10^3$, $b{=}3$. Work is messages plus comparisons per routed task; J_{build} the one-time build (probes at 4.04 J plus build messages); J_{route} the energy of one routing decision; call-equivalents a framework's mean latency relative to AutoGen's single 7B call (the 14B orchestrator's energy is additionally scaled by $14/7$); break-even $J^{\text{M}}_{\text{build}}/(J_{\text{route}} - J^{\text{M}}_{\text{route}})$, the number of tasks after which a framework has spent MIDIAN's whole build.

	work / task	J_{build} (J)	J_{route} (J)	call-equiv.	break-even (tasks)
routers on verified outcomes					
MIDIAN w/o defenses	69	$1.9{\times}10^5$	$9{\times}10^{-3}$	0	—
MIDIAN	36	$2.0{\times}10^5$	$5{\times}10^{-3}$	0	—
flat probe argmax; bandits; kNN / MLP	1,000	$1.9{\times}10^5$	10^{-5}	0	—
frameworks: no build, one or more supervisor calls per task					
AutoGen	22	0	20.6	1.00	9,690
Microsoft Agent Framework	22	0	25.1	1.22	7,970
Google ADK	22	0	25.3	1.23	7,910
smolagents	22	0	31.2	1.51	6,410
LangGraph	22	0	40.9	1.99	4,880
OpenAI Agents SDK	22	0	43.3	2.10	4,610
CrewAI	22	0	65.2	3.17	3,060
LlamaIndex	22	0	97.4	4.73	2,050
CAMEL Workforce	22	0	104.8	5.09	1,900
Magentic-One (7B orchestrator)	22	0	219.8	10.67	908
Magentic-One (14B orchestrator)	22	0	367.9	8.93	543

Build and break-even by scale and budget. MIDIAN's build is 1.03–1.05 times nFb probes: 16,805, 49,428, and 83,008 at $n{=}10^3$ and $b{=}1, 3, 5$ (live ledgers), $1.68{\times}10^6$, $4.96{\times}10^6$, and $8.32{\times}10^6$ at 10^5 (live), and $1.68{\times}10^8$, $4.96{\times}10^8$, and $8.32{\times}10^8$ at 10^7 (Bernoulli ledgers, priced at the live per-probe energy; the live and Bernoulli counts agree to 0.03% where both exist); the largest overspend in any cell is 1.07 (LLMRouterBench, $b{=}1$), and on RouterEval's pools of a thousand and fewer MIDIAN spends 0.92 of the budget. Its per-query energy is 5.0, 7.0, and 9.0 mJ at the three populations and does not depend on b. Break-even versus the cheapest framework is 3,300, 9,690, and 16,280 queries at $n{=}10^3$ and $b{=}1, 3, 5$, and versus the costliest of the ten, Magentic-One with its 7B orchestrator, 309, 908, and 1,530; each scales linearly with n, so per agent it is 3.3, 9.7, and 16.3 queries versus AutoGen and 0.31, 0.91, and 1.53 versus Magentic-One at every population size. MIDIAN without defenses builds with exactly nFb probes and adds 432 J of build reports at $n{=}10^3$.

What the comparison leaves out. The framework band is a live measurement at $n{=}10^3$ applied at every n: the ledger charges a framework k comparisons and $k + 2$ messages per task whatever n is, so retrieval over n descriptions is not charged; the registry build of n messages (1 J at 10^3, 10^4 J at 10^7) is not drawn; and the supervisor prompt is taken as $k{=}10$ descriptions at every n. Each omission favors the frameworks, so the band is a lower bound on their cost at large n under this model. Flat probe argmax, which builds with exactly nFb probes and routes on n comparisons, would lie within a few percent of MIDIAN in Figure 4: its build is 3–5% smaller and its per-query floor is 10^{-5} J at 10^3 and 0.1 J at 10^7. The energy advantage in that figure is over the frameworks; the advantage over flat methods is Figure 11's routing work.