A Deterministic Testbed for Self-Organizing Agent-Team Coordination

Ablatable Mechanisms and Measurable Noise-Robustness under a Matched Experiment Budget

Daniel Ari Friedman
Active Inference Institute
daniel@activeinference.institute
ORCID: 0000-0001-6232-9096

June 1, 2026

Contents

1 Abstract

Recent work on *AutoScientists* [Gao et al., 2026] coordinates self-organizing teams of language-model agents through a small set of shared mechanisms: a champion-and-experiment-log shared state, a registry of retired dead-end directions, effect-size ranking of candidate directions, noise-band confirmation of claimed improvements, and stagnation-driven reorganization of teams. This exemplar provides a deterministic, standalone reference implementation of those mechanisms and studies them honestly as a *testbed* rather than as a performance claim.

We make the comparison fair by holding the total number of objective evaluations fixed: coordinated teams *partition* a single sequential experiment budget rather than adding parallel compute. Under that matched budget, coordination cannot — and in our results does not — beat a single-thread baseline on the final champion metric; we report the actual numbers and claim no speedup. What the testbed *does* demonstrate are two distinct, independently measurable benefits. First, noise-robustness: because the objective is stochastic, a single observed gain can be a draw of evaluation noise, so we separate the *reported* champion metric from the *clean* noise-free ground truth and show that noise-band confirmation shrinks the gap between them by roughly an order of magnitude — with confirmation on, the final champion's reported metric sits 0.0012 above its clean value, against 0.0156 with confirmation removed, while every configuration reaches the same clean optimum. Second, search hygiene: the dead-end registry, consulted by the proposer, cuts redundant re-probes of retired directions from 36 to 0 and halts at 36 of the 60 experiments — the same clean answer, reached with less waste. A per-mechanism ablation isolates each component's contribution, and the language-model proposer is a clean plug-in seam: a deterministic rule-based agent drives the reproducible figures, and a live Hermes agent (served by Ollama) can be swapped in without touching the coordination loop.

2 Introduction

2.1 Motivation

Long-running scientific experimentation — tuning a model, searching a design space, optimizing a noisy objective over many trials — has become a target for multi-agent language-model systems. *AutoScientists* [Gao et al., 2026] frames this as a coordination problem: several agent teams share a running record of what has been tried, retire directions that repeatedly fail, prioritize directions with large observed effects, confirm claimed improvements against evaluation noise, and reorganize when progress stalls. These are appealing ideas, but they are easy to *describe* and hard to *attribute*: when a coordinated system performs well, which mechanism deserves the credit, and how much of an apparent gain is simply noise?

This exemplar exists to make those questions answerable on a small, fully reproducible artifact. It is one of a family of research-project templates in this repository, each pairing a tested computational core with a rendered manuscript. Here the core is a deterministic re-implementation of the AutoScientists coordination mechanisms, and the manuscript is an honest report of what they do.

2.2 An honest framing

It is tempting to advertise multi-agent coordination as "faster" or "better" search. We deliberately do not. The decisive design choice in this testbed is that **coordinated teams partition the same sequential experiment budget as the baseline**; they do not add parallel compute. Splitting a fixed budget across teams is a *constraint*, not extra horsepower. Under such a matched budget there is no mechanism by which dividing the work can beat doing it in one undivided thread on the final metric — and our results confirm that the clean-metric advantage of coordination over the baseline is exactly zero.

What remains, and what is genuinely worth demonstrating, are two benefits that the matched budget does *not* foreclose: **robustness to evaluation noise** and **search hygiene**. The objective is stochastic: every evaluation adds a seeded perturbation, so an observed "improvement" may be a lucky draw rather than a real gain. We therefore track two quantities throughout:

- the **reported metric** — the value the search believes its champion achieves, computed from noisy observations; and
- the **clean metric** — the noise-free ground-truth value at the champion's parameters, available to us only because the objective is synthetic.

A configuration that accepts noise-inflated champions will show a large reported-minus-clean gap. Noise-band confirmation is precisely the mechanism that closes that gap, and the testbed measures by how much. Separately, we track how the budget is *spent*: the dead-end registry, consulted by the proposer, lets the search avoid re-probing directions already known to fail and halt once they are exhausted. Neither benefit is a speedup — they change how good the reported answer is and how much of the budget is wasted, not how good the clean answer is.

2.3 Contributions

1. **A deterministic, ablatable reference** for the five AutoScientists coordination mechanisms, with every mechanism switchable via a single configuration object so its contribution can be isolated.
2. **An honest matched-budget comparison** of coordinated teams against a single-thread baseline that reports the actual numbers and makes no speedup claim.
3. **A reported-vs-clean noise-robustness measurement** that quantifies the value of noise-band confirmation as a roughly order-of-magnitude reduction in accepted noise.
4. **A search-hygiene measurement** that quantifies the dead-end registry as a reduction of redundant re-probes from 36 to 0 and an early halt at 36 of 60 experiments, with the clean optimum unchanged.
5. **A language-model plug-in seam** (`Proposer` protocol) that lets a live Hermes agent replace the deterministic proposer without modifying the coordination loop, exercised by an opt-in `requires_ollama` test.

The remainder of the paper specifies the mechanisms (sec. 3), reports the matched-budget comparison and the per-mechanism ablation with the numbers our scripts actually produce (sec. 4), states the scope and limits of those claims (sec. 8), and documents reproduction (sec. 7).

3 Methodology

The testbed is a single coordination loop over a fixed budget of experiments. Each mechanism is an independent module so it can be tested and ablated in isolation; the loop wires them together. All logic lives in `src/`; the analysis scripts only orchestrate, plot, and write.

3.1 The synthetic objective

The objective stands in for the expensive, stochastic evaluation a real run would optimize (a validation score, a correlation, a loss). It is a pure function of `(params, seed)`: identical inputs always yield identical outputs, which is what makes the whole exemplar reproducible.

For a parameter vector $x \in \mathbb{R}^d$ with optimum at the origin, the **clean** (noise-free) value is

$$f(x) = -\sum_{i=1}^{d} \left[x_i^2 + \rho\left(1 - \cos(2\pi x_i)\right)\right],$$

a smooth global peak at $x = 0$ (where $f = 0$) minus shallow cosine ripples of amplitude ρ that create deceptive local optima along each axis. Higher is better. A single noisy **observation** adds a seeded, zero-centred perturbation:

$$\tilde{f}(x, s) = f(x) + \varepsilon(x, s), \qquad |\varepsilon| \le \sigma_{\text{noise}},$$

where $\varepsilon(x, s)$ is derived deterministically from a hash of the rounded x and the seed s. Re-evaluating the same point under a *different* seed gives a different draw (modelling run-to-run variance); the same seed always reproduces the same value. We use $d = 4$, ripple $\rho = 0.15$, and noise scale $\sigma_{\text{noise}} = 0.02$.

3.2 Shared state

The deterministic core mirrors the AutoScientists shared state: an immutable **champion** record p (parameters, metric, originating experiment index) plus an append-only **experiment log** L of structured outcomes. Recording an outcome appends it to L and promotes the champion only when the outcome *improved* — i.e. it was confirmed and beat the incumbent. The champion metric is the value plotted against experiment count.

3.3 The five mechanisms

Noise-band confirmation. Because a single observed gain may be noise, a candidate is confirmed only when its mean metric over several seeds exceeds the incumbent by more than an empirical noise band. For seeds S and per-evaluation noise σ_{noise}, the band is $\sigma \cdot \sigma_{\text{noise}}/\sqrt{|S|}$ standard errors of the mean (default $\sigma = 2$), so it shrinks as more seeds are averaged. A candidate is confirmed iff its mean-over-seeds delta exceeds the band. This estimator is domain-agnostic; a synchronized generic copy lives at `infrastructure.scientific.confirmation` for reuse.

Dead-end registry. A registry D keyed by `(axis, direction)` tracks consecutive non-improving experiments. A direction is *retired* after it fails to improve the champion `threshold` times in a row (default 3); a confirmed improvement clears the streak. Agents consult D before proposing so exhausted directions are not re-explored. An axis is *fully retired* only when both its increase and decrease directions are retired.

Effect-size ranking. The analyst role prioritizes directions with large observed effects. We estimate each axis's effect size as the mean absolute metric delta observed for it in L, then order axes by descending effect — with the deliberate twist that **untried axes sort first**, so under-explored directions are probed before the search exploits known-large-effect axes. Ties break by axis index for determinism.

Stagnation-driven reorganization. A detector fires when the champion has not improved within a window of recent experiments (default 10). On firing, teams are re-partitioned around the currently most-promising live axes, dropping fully-retired ones.

Team partitioning. Live axes are dealt round-robin across `num_teams` teams (default 3) so each team works a complementary slice of the ranked directions. Crucially, the teams share one budget: experiment t is taken by team t mod num_teams.

3.4 The coordination loop

```
for each experiment in the budget:
    pick the next team and its live (non-retired) axes
    proposer proposes the next (axis, signed step) from shared state
    evaluate the candidate; if confirmation is on, average over seeds and test the band
    record the outcome; promote the champion if it improved
    update the dead-end registry
    if reorganization is on and the search is stagnant, re-partition teams
```

Every mechanism is gated by a boolean in `SearchConfig`. With all toggles off and a single team, the loop reduces exactly to the single-thread baseline — which is what makes the ablation a clean subtraction.

3.5 The proposer seam

The loop depends only on a `Proposer` protocol — `propose(state, axes, proposer_id) -> Proposal`. Two *real* implementations are provided (no mocks):

- **`DeterministicProposer`** — a rule-based policy that probes the next assigned axis in the direction that most recently improved it, defaulting toward the origin. It drives every rendered figure and test.
- **`HermesProposer`** — renders the shared state to a prompt, asks a Hermes model (served by Ollama) for the next `(axis, step, rationale)` as JSON, and parses the reply, rejecting any axis outside the assigned set. Its infrastructure-LLM import is lazy, so the deterministic core tests and renders with no Ollama dependency.

Swapping one for the other is the only change needed to turn the deterministic reference run into a live agentic one.

4 Results

All numbers below are produced by the analysis scripts in `scripts/` and written to `output/data/` as machine-readable JSON alongside the figures. They are deterministic: re-running the scripts reproduces them exactly. The budget is fixed at 60 experiments for every configuration.

4.1 Matched-budget comparison

fig. 1 plots the champion trajectory of the coordinated three-team configuration against the single-thread baseline over the shared 60-experiment budget. The two curves track each other and converge to the same value.

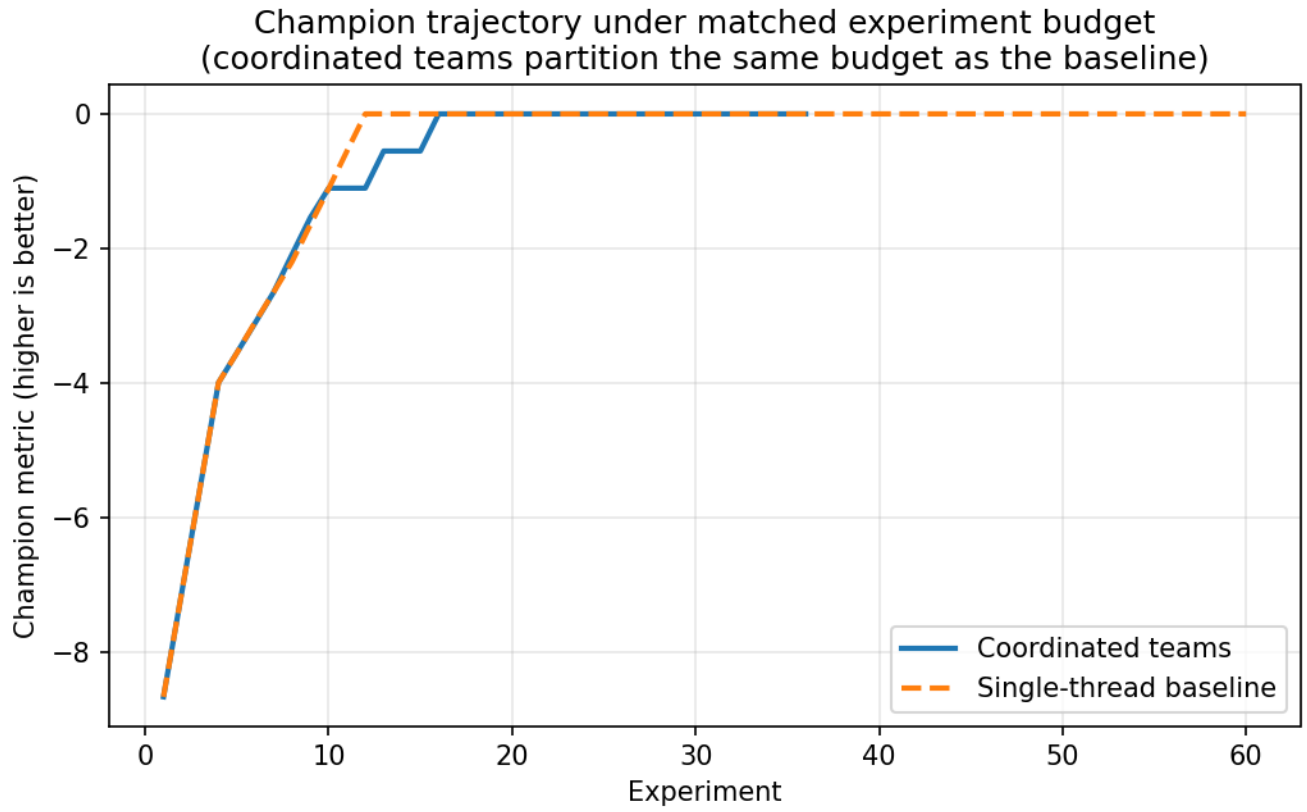

Figure 1: Champion metric (higher is better) versus experiment index for coordinated teams (solid) and the single-thread baseline (dashed) under a matched experiment budget. Coordinated teams partition the same sequential budget as the baseline rather than adding parallel compute, so this is a robustness/efficiency comparison, not a speedup. Produced by `run_search_comparison.py`.

The summary in `output/data/search_comparison.json` reports the decisive quantities:

Configuration	Reported metric	Clean metric	Experiments to optimum	Experiments used	Redundant re-probes
Coordinated teams	0.0012	0.0000	16	36	0
Single-thread baseline	0.0012	0.0000	12	60	36

Both configurations reach the **same clean ground-truth optimum** (0.0000, the global peak), and the clean-metric advantage of coordination over the baseline is exactly 0.0000. This is the honest headline: under a matched sequential budget, splitting the work into coordinated teams does **not** beat the undivided baseline on solution quality. If anything it is slightly *slower* to first reach the optimum — the coordinated run gets there at experiment 16 versus the baseline's 12, because partitioning four axes across three teams interleaves the descent. We make no speedup claim, because the testbed is constructed so that none would be honest.

What the coordinated configuration *does* gain is **search hygiene**: it retires exhausted directions and stops, using only 36 of the 60 experiments with zero redundant re-probes, whereas the baseline — which runs without the dead-end registry — spends the full budget and wastes 36 experiments re-testing directions already known to fail. The coordination machinery changes *how* the budget is spent, not how good the final answer is.

4.2 Per-mechanism ablation

The testbed separates two distinct, independently measurable benefits — noise robustness and search hygiene — from the mechanisms that do not move the needle on this objective. fig. 2 and fig. 3, drawn from `output/data/ablation.json`, switch off one mechanism at a time starting from the full coordinated configuration.

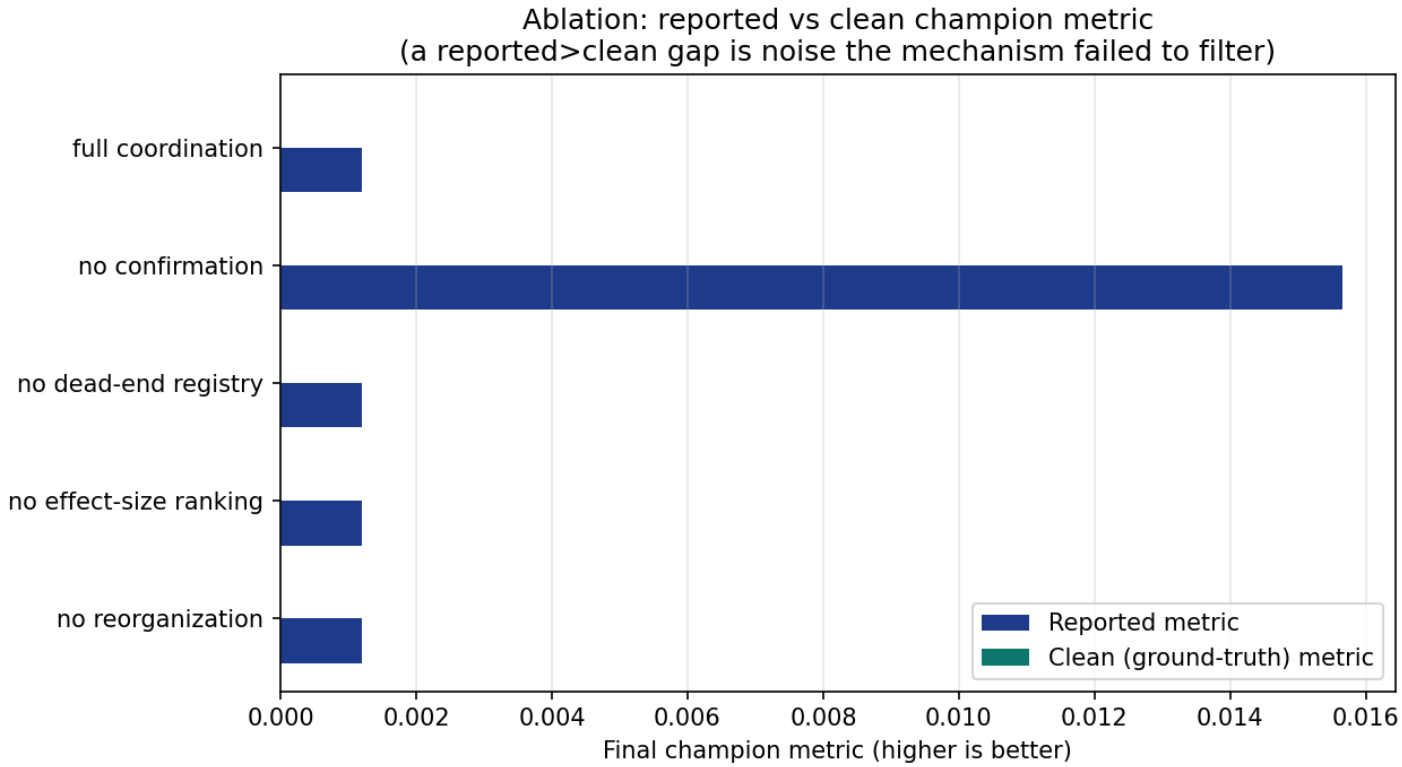

Figure 2: Reported versus clean (ground-truth) champion metric for the full configuration and each single-mechanism ablation. A reported-greater-than-clean gap is accepted noise. Removing noise-band confirmation inflates the gap roughly thirteenfold while the clean metric is unchanged. Produced by `run_ablation.py`.

Configuration	Reported metric	Clean metric	Noise inflation	Experiments used	Redundant re-probes
Full coordination	0.00121	0.0000	0.00121	36	0
No confirmation	0.01565	0.0000	0.01565	36	0
No dead-end registry	0.00121	0.0000	0.00121	60	36
No effect-size ranking	0.00121	0.0000	0.00121	36	0
No reorganization	0.00121	0.0000	0.00121	36	0

Confirmation is the load-bearing mechanism for honesty. Removing noise-band confirmation leaves the clean metric untouched (the search still lands on the optimum) but inflates the *reported* metric from 0.00121 to 0.01565 — a roughly 13× increase in accepted noise. Without confirmation the search promotes a champion whose reported value overstates its true value by an order of magnitude more; with confirmation the reported metric stays close to the truth. This is exactly the failure mode noise-band confirmation is designed to prevent, and the testbed measures its size.

The dead-end registry is the load-bearing mechanism for efficiency. fig. 3 shows that removing it is the only ablation that changes the experiment budget profile: the registry-consulting proposer otherwise never re-probes a retired direction

(redundant re-probes = 0) and halts at 36 experiments once every direction is exhausted, while the no-registry configuration burns all 60 experiments and wastes 36 of them re-testing known dead ends. Crucially, this hygiene gain leaves the clean metric **unchanged** at 0.0000 — the registry buys a leaner search, not a better answer.

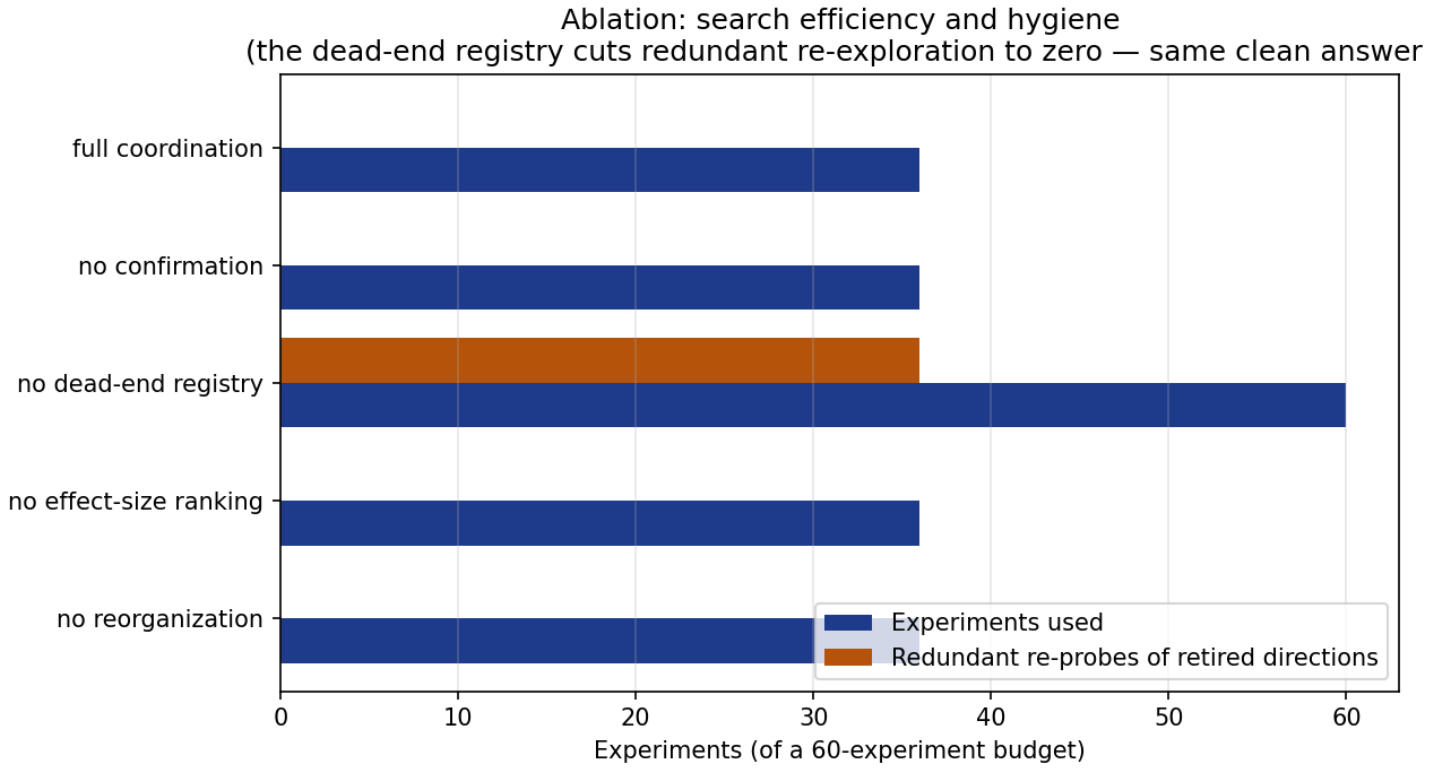

Figure 3: Experiments used and redundant re-probes of retired directions per configuration. Only the dead-end-registry ablation changes the profile: without the registry the search wastes 36 experiments re-exploring known-dead directions and never halts early. Produced by `run_ablation.py`.

Effect-size ranking and reorganization do not move any metric here. Removing either leaves the reported metric, clean metric, experiments used, and redundant re-probes all unchanged. On this small, separable objective with a deterministic proposer, those two mechanisms reshape the *order* in which directions are tried without changing the destination, the noise, or the efficiency within the budget. We report this plainly rather than dressing it up: both are correctly implemented and independently tested, but their benefit is about exploration bookkeeping on harder or more deceptive landscapes, not about measurable gains on this testbed.

4.3 What the numbers say

Four honest conclusions follow directly from the data:

1. Under a matched budget, coordinated teams neither beat nor lose to the single-thread baseline on clean solution quality (advantage = 0.0000), and partitioning is in fact marginally slower to first reach the optimum (16 vs 12 experiments).
2. Noise-band confirmation delivers a measurable, reproducible robustness benefit: a $\sim 13\times$ reduction in accepted noise ($0.01565 \rightarrow 0.00121$ reported-vs-clean gap).
3. The dead-end registry delivers a measurable efficiency benefit: redundant re-probes fall from 36 to 0 and the search halts at 36 rather than 60 experiments — with the clean answer unchanged.
4. Effect-size ranking and reorganization are correctly implemented and ablatable, but do not by themselves change any measured quantity on this objective — a result we report rather than obscure.

5 Conclusion

This exemplar re-implements the AutoScientists coordination mechanisms [Gao et al., 2026] as a deterministic, ablatable testbed and reports what they actually do under a fair, matched experiment budget. The central methodological commitment is honesty about the comparison: coordinated teams partition one sequential budget rather than adding parallel compute, so we neither expect nor observe a speedup, and we say so. The clean-metric advantage of coordination over a single-thread baseline is exactly zero.

Two results are worth keeping. The first is the noise-robustness measurement: by separating the reported champion metric from the clean ground-truth metric — possible only because the objective is synthetic — the testbed quantifies noise-band confirmation as a roughly thirteenfold reduction in accepted noise, with the clean optimum reached either way. The second is search hygiene: the dead-end registry, consulted by the proposer, drives redundant re-probes of retired directions from 36 to 0 and lets the search halt at 36 of the 60 experiments instead of burning the full budget — without changing the clean answer. The remaining structural mechanisms (effect-size ranking, stagnation reorganization) are correctly implemented and independently testable, but on this separable objective they reshape the search path without changing its destination, noise, or efficiency within the budget; we report that rather than overstate it.

Two properties make the artifact reusable. First, every mechanism is gated behind a single configuration object, so the ablation is a clean subtraction and the testbed extends naturally to harder objectives where the structural mechanisms have more to do. Second, the language-model proposer is a genuine plug-in seam: the deterministic proposer drives the reproducible figures, and a live Hermes agent can replace it without touching the coordination loop. The honest-testbed framing is the contribution — a small, fully reproducible instrument for attributing coordination effects to mechanisms and for distinguishing real gains from noise.

6 Experimental Setup

6.1 Objective and budget

All experiments optimize the synthetic objective of sec. 3 with $d = 4$ dimensions, ripple amplitude $\rho = 0.15$, and per-evaluation noise scale $\sigma_{\text{noise}} = 0.02$. The global optimum is the origin, where the clean objective equals 0. Every configuration is given the **same** budget of 60 sequential experiments; coordinated configurations partition that budget across teams.

6.2 Configurations

The configurations compared in sec. 4 correspond directly to `SearchConfig` objects:

- **Coordinated teams** — the full configuration: 3 teams, all mechanisms on (`use_confirmation`, `use_dead_ends`, `use_ranking`, `use_reorganization` all true).
- **Single-thread baseline** — `SearchConfig.single_thread_baseline()`: 1 team, confirmation on (so the baseline is itself noise-honest), all structural coordination off.
- **Ablations** — the full configuration with exactly one mechanism switched off, generated via `dataclasses.replace`.

Confirmation averages each candidate over seeds $(101, 202, 303)$ and tests against a $\sigma = 2$ noise band; the primary evaluation seed is 7; the stagnation window is 10 experiments; a direction is retired after 3 consecutive non-improving experiments. These values are the `SearchConfig` defaults and are echoed in `manuscript/config.yaml`.

6.3 Proposer

The rendered figures and the JSON summaries are produced with `DeterministicProposer`, a rule-based agent that reads the shared state and emits a concrete proposal. No mock objects are used anywhere. The live `HermesProposer` path is not part of the rendered pipeline; it is exercised separately (see sec. 7).

6.4 Outputs

Two thin orchestrator scripts produce all results:

- `scripts/run_search_comparison.py` → `../figures/search_comparison.png`, `output/data/search_comparison.json`.
- `scripts/run_ablation.py` → `../figures/ablation.png`, `output/data/ablation.json`.

Each script imports all computation from `src/`, runs the configurations, and writes a figure plus a machine-readable summary. The numbers quoted in sec. 4 are read directly from those JSON files.

7 Reproducibility

Every figure and number in this manuscript is regenerable from a clean checkout with fixed seeds and no network access.

7.1 Deterministic core

The objective is a pure function of `(params, seed)`, and the coordination loop is deterministic given the objective, proposer, and configuration. Re-running the analysis scripts reproduces the figures and the JSON summaries byte-for-byte.

```
# Regenerate the matched-budget comparison and the ablation
uv run python projects/templates/template_autoscientists/scripts/run_search_comparison.py
uv run python projects/templates/template_autoscientists/scripts/run_ablation.py
```

7.2 Tests and coverage

The project carries its own test suite under `tests/`, run as a standalone per-project gate. There are no mocks anywhere: the `DeterministicProposer`, the synthetic objective, and the registries are all real objects exercised with real numerical inputs.

```
# Project test suite with the per-project coverage gate
uv run pytest projects/templates/template_autoscientists/tests/ \
    --cov=projects/templates/template_autoscientists/src --cov-fail-under=90
```

The deterministic core is tested to full coverage. The live language-model path is excluded from the coverage gate (`# pragma: no cover`) because it requires an external service.

7.3 The live Hermes agent (opt-in)

`HermesProposer` calls a Hermes model served by Ollama. It is not part of the rendered pipeline and is exercised only by an opt-in test marked `requires_ollama`:

```
# One-time: start Ollama and pull a Hermes model
ollama serve
ollama pull hermes3

# Run the live round-trip test
uv run pytest projects/templates/template_autoscientists/tests/test_hermes_live.py \
    -m requires_ollama -v
```

Because the loop depends only on the `Proposer` protocol, swapping `DeterministicProposer` for `HermesProposer` is the single change needed to turn the reproducible reference run into a live agentic one — the coordination mechanisms, ablation toggles, and confirmation logic are unchanged.

7.4 Shared estimator

The noise-band confirmation estimator is generic. A synchronized copy lives at `infrastructure.scientific.confirmation` (`confirm_improvement`, `Confirmation`) for reuse by any other project that compares a stochastic metric to a baseline, and is covered by `tests/infra_tests/scientific/test_confirmation.py`. The project keeps its own standalone copy so the exemplar runs self-contained.

8 Scope and Related Work

8.1 What this exemplar claims

- The five AutoScientists coordination mechanisms [Gao et al., 2026] are re-implemented faithfully enough to be **individually ablatable**, and each is independently tested.
- Under a **matched sequential experiment budget**, coordinated teams reach the same clean optimum as a single-thread baseline (advantage = 0.0000).
- **Noise-band confirmation** produces a measurable, reproducible reduction in accepted evaluation noise (a $\sim 13\times$ smaller reported-vs-clean gap on this objective).
- The **dead-end registry** produces a measurable, reproducible search-hygiene gain: consulted by the proposer, it cuts redundant re-probes of retired directions from 36 to 0 and halts the search at 36 of 60 experiments, with the clean optimum unchanged.
- The **language-model proposer is a clean plug-in seam**: a live Hermes agent can replace the deterministic proposer without changing the coordination loop.

8.2 What this exemplar does *not* claim

- **No speedup.** Because coordinated teams partition one budget rather than adding parallel compute, this testbed is not evidence that coordination is faster or reaches better solutions. It is constructed so that no such claim would be honest, and the measured advantage is zero.
- **No generalization of the magnitudes.** The $\sim 13\times$ noise-reduction figure, the $36 \to 0$ redundancy reduction, and the null effect of effect-size ranking and reorganization on every measured quantity are properties of *this* synthetic objective, budget, and deterministic proposer. They illustrate the measurement, not a universal constant.
- **No agentic-quality claim.** The live Hermes path demonstrates that the seam works, not that an LLM proposer outperforms the rule-based one.

8.3 Relationship to AutoScientists

The original system [Gao et al., 2026] runs real language-model agent teams on real, expensive scientific tasks and reports end-to-end performance. This exemplar deliberately strips that to a deterministic core so the *mechanisms* can be attributed and the noise behavior measured in isolation. It is a complement — a microscope on the coordination primitives — not a reproduction of the full system or its empirical results.

8.4 Related context

The confirmation mechanism is an application of standard effect-size and standard-error reasoning [Cohen, 1988] to online acceptance decisions. The dead-end registry, effect-size ranking, and reorganization are coordination heuristics whose lineage runs through population- and restart-based search [Whitley, 2001]; the contribution here is not the heuristics but their honest, ablatable measurement. The broader setting — teams of language-model agents pursuing a long-running objective — sits within the rapidly growing literature on LLM-based autonomous agents [Wang et al., 2024]. Throughout, the emphasis on regenerable figures, fixed seeds, and a tested core follows the reproducible-research tradition [Peng, 2011].

9 References

Bibliography lives in manuscript/references.bib and is read by Pandoc during PDF render. The build pipeline invokes Pandoc with `--natbib`, so every `[@key]` citation in the manuscript is rewritten to the appropriate `\cite{}`/`\citep{}`/`\citet{}` LaTeX command and resolved against the bib file.

To validate that `references.bib` is syntactically clean and contains the required fields per entry type:

```
uv run python -m infrastructure.reference.citation.cli validate \
    projects/templates/template_autoscientists/manuscript/references.bib --strict
```

References

Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 2 edition, 1988. ISBN 978-0-8058-0283-2.

Shanghua Gao, Wenhao Fang, and Marinka Zitnik. AutoScientists: Self-organizing agent teams for long-running scientific experimentation. *arXiv preprint arXiv:2605.28655*, 2026. doi: 10.48550/arXiv.2605.28655. URL https://arxiv.org/abs/2605.28655.

Roger D Peng. Reproducible research in computational science. *Science*, 334(6060):1226–1227, 2011. doi: 10.1126/science.1213847.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), 2024. doi: 10.1007/s11704-024-40231-1. URL https://arxiv.org/abs/2308.11432.

Darrell Whitley. An overview of evolutionary algorithms: practical issues and common pitfalls. *Information and Software Technology*, 43(14):817–831, 2001. doi: 10.1016/S0950-5849(01)00188-4.