When does multi-agent decomposition beat single-agent execution? Seven trials on the Turing SwarmBench benchmark suite reveal a context-and-authority bottleneck.

Eric Gitangu Independent Researcher, Nairobi, Kenya `developer.ericgitangu@gmail.com` | `linkedin.com/in/ericgitangu`

Prepared 2026-05-01.

Abstract

Multi-agent LLM systems are widely claimed to outperform single-agent LLMs on complex tasks, but the empirical conditions under which the advantage holds remain underspecified. This paper documents a seven-attempt benchmark-engineering study against the Turing SwarmBench Code/SWE assessment suite, executed against Kimi K2.5 served via Fireworks AI through the Harbor open-source evaluation harness. Six attempts measured the multi-vs-single reward gap empirically across progressively larger Python codebases (Click 5K LOC; Werkzeug 16K LOC; Apache Airflow 303K LOC) and two verifier types (executable AST-token assertions; LLM-as-judge over structured CVE extractions). A seventh attempt redesigned the task around a single-authority constraint and submitted a design-stage gap argument under the assessment's Section-8 oracle-only validation regime. Three pre-registered hypotheses (codebase size scales the wedge; bug count scales the wedge; per-item structured research scales the wedge) were tested. Two were falsified outright; the third produced a small but visible gap (0.05) before the unified-authority pivot. The headline finding: on Kimi K2.5 with kimi-cli's `Read` / `Grep` / `Edit` tool surface, multi-agent decomposition is rate-limited by *which bottleneck dominates the task*. It cures **context overflow on independent items**; it does *not* cure capability ceilings on individual subtasks, authority-disambiguation noise, or retrieval precision. Three of three published SwarmBench wedges (AGENTBENCHLANDSCAPE 0.53, MEDICALRESEARCH 0.46, VENDORCROSSREF 0.22) are LLM-judge tasks over many independent items — none from executable code-SWE. The implication for benchmark engineers and production multi-agent system designers is that the swarm advantage is structurally narrower than commonly portrayed and is engineered, not stumbled upon.

Keywords: multi-agent LLM systems, benchmark engineering, evaluation harness, context overflow, fan-out-synthesize, Kimi K2.5, Harbor framework, SwarmBench, falsification, security advisory extraction.

1. Introduction

The claim that "more agents do more work" has become a load-bearing assumption in production LLM system design. Frameworks such as AutoGen [1], MetaGPT [2], and CrewAI promise that decomposing a task across multiple specialised LLM instances yields scores no single instance can match. Multi-agent debate work [3] reports measurable gains on factuality and reasoning. Yet the *conditions* under which decomposition produces measurable benefit are rarely stated as falsifiable empirical claims — and benchmark suites that purport to measure the multi-agent advantage often select for tasks where the advantage is structurally guaranteed (very long inputs, explicit independence between items) rather than tested against tasks where it is genuinely uncertain.

The Turing SwarmBench Code/SWE assessment provides such a forcing function. The published quality gate [4] requires a strictly-greater-than-0.38 reward gap between a multi-agent run and a single-agent run on the same task, evaluated on Kimi K2.5 [5] inside the Harbor evaluation harness [6]. The benchmark's calibrated example tasks demonstrate that gaps in this range *do* emerge for some task families: AGENTBENCHLANDSCAPE reports a 0.53 gap on a research-synthesis LLM-judge task, MEDICALRESEARCH reports 0.46 on a medical-literature task, and VENDORCROSSREF reports 0.22 on a vendor-cross-reference data-analysis task. The existence proof is real. What is unclear is whether the wedge transfers to other task families — particularly to executable code-SWE, the assessment's nominal target domain.

This paper documents a seven-trial empirical investigation of the boundary. The research question is precise:

> *Under what task properties does multi-agent decomposition empirically produce a > 0.38 multi-vs-single reward advantage on Kimi K2.5, and where does the wedge collapse?*

The contribution is three-fold:

1. **Empirical falsification**, in the Popperian sense, of three pre-registered hypotheses relating codebase size, bug count, and per-item research load to the wedge magnitude — under the conditions tested, with one model, one harness, and k=n=1 trial counts.
2. **A taxonomy of bottleneck types** (context-overflow, capability-ceiling, authority-disambiguation, retrieval-precision) and a mapping of which the multi-agent decomposition pattern actually relieves.
3. **A design-stage demonstration** (attempt 7) of the unified-authority pivot — collapsing multi-source disagreement noise to expose the underlying wedge mechanism — submitted under the assessment's oracle-only Section-8 validation rule.

Single-trial counts at k=n=1 mean we cannot make strong statistical claims about effect-size precision. We lean instead on *falsification logic*: when the predicted gap is impossible (single = 1.000 caps the multi mean), one trial suffices to rule out the hypothesis under those conditions.

---

2. Related Work

Multi-agent LLM frameworks. AutoGen [1] formalised conversational multi-agent orchestration with code-executable conversation patterns. MetaGPT [2] introduced role-specialisation drawn from human software-engineering workflows (PM, Architect, Engineer, QA). The multi-agent debate paradigm of Du et al. [3] showed that having multiple instances debate for several rounds improves factual reasoning over a single CoT pass. None of this prior work conditions its claims on which *bottleneck* the task imposes — the question this study foregrounds.

Multi-agent benchmarks. AgentBench [7] established a multi-environment benchmark for LLM agents across eight domains. SWE-bench [8] and its human-verified subset SWE-bench Verified [9] focus on single-agent code repair against real GitHub issues. The Turing SwarmBench suite (used here, distinct from the similarly-named YuLan-SwarmIntell academic project [10]) explicitly conditions evaluation on the *gap* between coordinated swarms and solo agents, surfacing the question of when decomposition adds value at all.

Naming-collision note. Two distinct projects share the name "SwarmBench" in 2025–2026:

- *Turing's SwarmBench* — a Code/SWE evaluation suite built atop the Harbor open-source eval framework [6]; this is the benchmark this paper engages with.
- *YuLan-SwarmIntell SwarmBench* [10] — an academic project from the Gaoling School of Artificial Intelligence at Renmin University, evaluating LLM agents in spatially-constrained, decentralised swarm-coordination tasks (Pursuit, Foraging, Flocking, Transport).

The two projects do not share authorship, code, or task design. This paper engages only with the former.

Code-SWE evaluation harnesses. Harbor [6] provides the apparatus: oracle/single/multi/k/n trial structure, dockerised verifier execution, JSON-schema-validated task packaging, and per-trial reward aggregation. The SwarmBench-specific patches add `swarm-kimi-single` and `swarm-kimi-multi` orchestrator classes that wrap kimi-cli with fan-out-synthesize coordination drawn from `decomposition.yaml`.

---

3. Hypotheses

The hypotheses below were formulated from the design-time hypothesis space, in decreasing order of prior probability assigned at the start of the engineering work:

- **H1 — Codebase-size hypothesis.** *More LOC produces more context pressure; multi-agent advantage scales with codebase size.* Prior: high. This is the most popular assumption in production-system folklore.
- **H2 — Bug-count hypothesis.** *More independent bugs produce more decomposition surface; multi-agent advantage scales with bug count when bugs are mutually independent.* Prior: high-medium. This is the structural-independence argument.
- **H3 — Per-item structured-research hypothesis.** *More independent web-fetches with structured per-item extraction produce more parallelism benefit; multi-agent advantage scales with the number of independent retrieve-and-extract operations.* Prior: medium. Extrapolation from the published AGENTBENCHLANDSCAPE 0.53 wedge.

Each hypothesis was tested against at least one purpose-designed task, using Kimi K2.5 + Harbor. The seventh attempt is a design-stage refinement, not a fresh hypothesis.

---

4. Methods

4.1 Apparatus

- **Model:** Kimi K2.5 via Moonshot AI [5], served by Fireworks AI's hosted inference. 256K nominal context window. The model's tool-use surface is exposed to the agent through *kimi-cli*, which advertises `Read`, `Grep`, `Edit`, `Bash`, and a web-fetch tool.
- **Harness:** Harbor evaluation framework [6], invoked via `uv run harbor run -p <task-dir> -a {oracle,single-kimi-agent,multi-kimi-agent} -k 1 -n 1`. Oracle, single, and multi modes share verifier and dockerfile but differ in agent class.
- **Coordination patches:** SwarmBench-specific `swarm-kimi-multi` orchestrator reads `decomposition.yaml` and dispatches one Kimi sub-process per leaf sub-task with focused per-sub-task prompts. Sub-process composition is verified by inspecting `agent/kimi-cli.txt` after each run; for example, attempt 2 confirmed three `Agent` tool-call invocations matching the three sub-tasks (lines 1279, 1833, 2511 of the trajectory log).
- **Isolation:** Each trial executes inside a Docker container described by `environment/Dockerfile`. Runs are pinned to `linux/arm64` for reproducibility on the author's hardware.

4.2 Verifier types

Two verifier modalities were used:

- **Executable AST verifier** (`verifier_type = executable`). For attempts 1–5. Implemented as Python that `ast.parse()`s patched source files, locates target functions/classes, and asserts that specific tokens appear in the function body. Pattern adopted from the SwarmBench NetBox example task. This bypasses the dependency-installation problem for large codebases (Apache Airflow has 200+ runtime dependencies; pip install was infeasible inside the 600 s build timeout, so AST verification reads source files directly).
- **LLM-judge verifier** (`verifier_type = llm-judge`). For attempts 6 and 7. Implementation is the AGENTBENCHLANDSCAPE judge cloned verbatim: a Kimi-K2.5-as-judge call that compares the agent's structured JSON output against `tests/oracle.json` field-by-field and returns a `passed/total` partial-credit reward. Critically, the judge contains an **equality short-circuit** at line 38: if `agent_output == oracle` (deep-equal), it returns reward = 1.0 without constructing the OpenAI client. This is what makes attempt 7's oracle path API-key-independent.

4.3 Coordination pattern

`fan-out-synthesize` was used in every attempt. Fan-out sub-tasks operate in parallel; one synthesizer sub-task depends on all fan-outs. The orchestrator dispatches sub-Kimi processes for each sub-task in `decomposition.yaml`; the synthesizer reads the fan-outs' shared filesystem outputs (`/logs/agent/<sub-task-id>.json`) and produces the final consolidated artifact. The single-agent run ignores `decomposition.yaml` entirely (the orchestrator does not surface it) and is given the raw `instruction.md` only.

4.4 Datasets

Attempt	Source	Version	LOC	Bugs/CVEs	Domain	Source URL
1	pallets/click	8.3.3	~5K	8	code-SWE	github.com/pallets/click [11]
2	pallets/click (rescoped)	8.3.3	~5K	3	code-SWE	github.com/pallets/click
3	pallets/click (widened verifier)	8.3.3	~5K	3	code-SWE	github.com/pallets/click
4	pallets/werkzeug	3.1.7	~16K	7	code-SWE	github.com/pallets/werkzeug [12]
5	apache/airflow	3.2.1	303K	16	code-SWE	github.com/apache/airflow [13]
6	12 Python-ecosystem CVEs	2024 disclosures	n/a	12 CVEs	code-SWE (advisory extraction)	NVD ⊕ GHSA
7	16 Python-ecosystem CVEs (GHSA-only)	2024 disclosures	n/a	16 CVEs	code-SWE (advisory extraction)	github.com/advisories [14]

For attempts 1–5, the upstream parent SHA pre-dating the bug-fix milestone is the broken state; the milestone's merge SHAs supply the canonical fix. For attempts 6–7, the GHSA panel values constitute ground truth.

4.5 Procedure

Trial structure for each attempt:

1. Build the Docker image and run `oracle` once at k=1, n=1 to confirm the verifier returns reward = 1.000 on the canonical fix. This is the QUALITY_GATE_PRD S-01 row gate; without it, no further runs are meaningful.
2. Run `single-kimi-agent` once at k=1, n=1.
3. Run `multi-kimi-agent` once at k=1, n=1, *only if* single's reward < 1.0 (otherwise the gap is mathematically capped at zero).

Reward = fraction of verifier checks passed (executable mode) or fraction of (entry, field) pairs the judge marked correct (llm-judge mode). Gap = `multi_mean - single_mean`. The QUALITY_GATE_PRD threshold for "wedge present" is gap > 0.38.

---

5. Results

5.1 Per-attempt summary

#	Source	Verifier	Single	Multi	Gap	Verdict
1	Click 8.3.3, 8 bugs	executable AST	**0.81**	n/a	≤ 0.19	infeasible — single solved 5/8 in one context
2	Click 3-bug rescope	executable AST	0.57	**0.57**	**0.00**	bug 3 unfixable for Kimi regardless of agent count
3	Click 3-bug widened verifier	executable AST	0.57	n/a	≤ 0.43	sub-check inflation cannot synthesise a wedge
4	Werkzeug 3.1.7, 7 bugs	executable AST	**1.00**	n/a	≤ 0.0	single solved every bug
5	Airflow 3.2.1, 16 bugs, 303K LOC	executable AST	**1.00**	n/a	≤ 0.0	single solved every bug at 303K LOC
6	12-CVE advisory extraction (NVD ⊕ GHSA)	llm-judge	0.72	0.77	**0.05**	wedge mechanism visible; authority noise dominated
7	16-CVE advisory extraction (GHSA-only)	llm-judge	*predicted* 0.55	*predicted* 0.95	*predicted* 0.36	design-stage; not measured (per assessment Section 8)

Bold cells mark the load-bearing observations for falsification.

5.2 Attempt 1 — Click 8-bug (testing H1, H2)

- *Hypothesis:* a real OSS multi-bug fix milestone in a 5K-LOC codebase will produce a measurable wedge.
- *Single mean:* 0.8125 (13/16 verifier checks). Single solved 5 of 8 bugs in one 256K context.
- *Capped multi gap:* $1.0 - 0.8125 = 0.1875 < 0.38$.
- *Conclusion:* H1/H2 falsified for small codebases. Click's 5K LOC + 8 independent bugs fit comfortably within Kimi's working memory; no measurable context pressure.

5.3 Attempt 2 — Click rescoped to 3 bugs (testing residual H2)

- *Hypothesis:* narrowing the scope to bugs the single-agent missed will let multi-agent's focused per-bug prompts pull ahead.
- *Single mean:* 0.5714. *Multi mean:* 0.5714. *Gap:* 0.000.

- The multi orchestrator did spawn three focused sub-agents (verified by three `Agent` tool-call invocations in `kimi-cli.txt`), confirming the orchestration patch operates correctly. Both agents missed the same bug — Click's `CliRunner.isolation` contextmanager wiring (the fix lives in a function-signature default, `stdin=sys.__stdin__`).
- *Conclusion:* the bottleneck for this bug is Kimi's per-bug capacity, not orchestration. Multi-agent does not help when individual sub-tasks exceed the model's per-context capacity.

5.4 Attempt 3 — Click verifier widened (verifier engineering)

- *Hypothesis:* sub-dividing the missed bug's check into 6 differentiating sub-checks lets partial credit favour multi-agent.
- *Single mean:* 0.5714 (bug 3 missed wholesale; all 6 sub-checks fail). Multi not run.
- *Conclusion:* sub-check inflation cannot synthesise a wedge if both agents whiff on the underlying bug. Verifier engineering operates *on top of* model capacity, not around it.

5.5 Attempt 4 — Werkzeug 7-bug (testing H1 at medium scale)

- *Hypothesis:* a 16K-LOC library with 7 bugs across 5 subsystems pressures single's context.
- *Single mean:* **1.000** (14/14). Multi was not executed; the gap was already capped at zero.
- *Conclusion:* medium-scale code-SWE remains within Kimi's working memory. H1 falsified at 16K LOC.

5.6 Attempt 5 — Apache Airflow 16-bug, 303K LOC (testing H1 at full scale)

This was the strongest possible test of H1: 303K LOC across 16 disjoint subsystems (logging, DAG bundle/zip, ORM async-engine, multi-team auth, asset-graph RBAC, Alembic migrations, scheduler heartbeat, UI, DAG serialization, connection model, SDK config, SQLA session lifecycle, JWT cookies, `@task` decorator validation, OTel forked workers, structlog plugin loader). Verifier uses 16 AST-token checks à la NetBox.

- *Single mean:* **1.000** (16/16, ~52 minutes wall clock). Multi was not executed; gap capped at zero.
- *Conclusion:* H1 strongly falsified. **At 303K LOC across 16 unrelated subsystems, Kimi K2.5 still solves every bug single-agent.** The mechanism is straightforward: kimi-cli's `Read` / `Grep` / `Edit` tool surface lets the agent page in only the file or function it needs, working module-by-module without ever holding the whole codebase in attention. The "context overflow" failure mode that produces wedges in research-synthesis tasks does not exist for code-SWE under this tool surface.

5.7 Attempt 6 — 12-CVE advisory extraction, llm-judge, two-source (testing H3)

- *Hypothesis:* per-CVE structured extraction across 12 web-fetched advisories overflows context the way AGENTBENCHLANDSCAPE does.
- *Single mean:* 0.72. *Multi mean:* 0.77. *Gap:* **0.05**. Below the 0.38 threshold, but the wedge mechanism is now *visibly present* for the first time across the trial sequence.
- *What the judge log revealed:* of the single-agent's 34 wrong fields out of 120 (12 CVEs × 10 fields), approximately 80% were authority-disagreement cases (NVD vs GHSA picked different CVSS panels, different CWE class lists, different version-range syntax conventions). This is not a failure of decomposition; it is a failure to disambiguate between two equally-authoritative sources.
- *Conclusion:* H3 is partially supported (the wedge mechanism reproduces, in the same direction predicted), but the bottleneck for *this particular task design* is **authority disambiguation**, not context overflow — and authority disambiguation is exactly the kind of failure mode multi-agent decomposition does *not* fix. Each sub-agent independently faces the same NVD-vs-GHSA choice; running 12 sub-agents in parallel produces 12 independent coin flips, not 12 correct answers.

5.8 Attempt 7 — 16-CVE advisory extraction, GHSA-only, design-stage submission

The empirical falsifications above motivated a structural redesign rather than a sixth empirical attempt at the same wedge. The redesign introduces one constraint: **GitHub Security Advisories is the single source of truth.** No NVD, no OSV, no vendor blog posts. Severity uses GHSA's `Critical / High / Moderate / Low` vocabulary verbatim (note: `Moderate`, not `Medium`). Each of the 10 per-CVE fields is tied to a specific named GHSA UI panel, with deterministic per-field rules (CVSS panel for `cvss_base_score`, References-tab regex for `poc_public`, Description first-sentence for `root_cause_quote`, etc.).

The submission was made under the assessment's Section-8 oracle-only validation rule: the LLM Reviewer scores the design-stage gap argument, not measured single/multi runs. The oracle path is verified at reward = 1.000 deterministically — `solution/solve.sh` copies `solution/oracle.json` to `/logs/agent/output.json`, and the AGENTBENCHLANDSCAPE judge's line-38 equality short-circuit returns reward = 1.0 without constructing an OpenAI client.

The gap argument predicts:

Quantity	Lower	Centre	Upper
Single-agent reward (predicted)	0.50	0.55	0.60
Multi-agent reward (predicted)	0.92	0.95	1.00
Gap (predicted)	**0.32**	**0.36**	**0.45**

These numbers are predicted, not measured. They are presented here as design-stage estimates derived from a step-function model of rule-application fidelity over the 16 CVEs (see Section 6.3 and Figure 3) plus the empirical 0.20 of authority-disagreement noise that the unified-authority constraint removes (per attempt 6's judge log). The design-stage gap argument lives in `gap_strategy.md` of the submitted task package; it is graded on its own merits per the assessment rubric, not by reviewers re-running trials.

---

6. Discussion

6.1 Why codebase size does not scale the wedge for code-SWE

The strongest empirical finding of this study is that **codebase size does not produce the wedge under modern tool surfaces.** Attempts 4 (16K LOC) and 5 (303K LOC) both observed single-agent reward = 1.000. The mechanism is mechanical, not surprising in retrospect: kimi-cli exposes `Read`, `Grep`, and `Edit` tools that let the agent operate on file fragments rather than the whole codebase. A single agent armed with these tools can — and on Kimi K2.5 routinely does — solve 16 bugs across 303K LOC by paging in one file at a time, because the relevant context window for any individual fix is the function or two functions surrounding the bug, not the whole repository. The 256K context window is never saturated.

This contradicts a naive "more LOC = more pressure" mental model that pervades production-system design folklore. The pressure is not against context-window size; it is against *what fits inside one context block of attended tokens during reasoning*. Modern tool surfaces decouple the two.

The implication: any benchmark that aspires to measure a multi-agent wedge on executable code-SWE must either restrict the agent's tool surface (a stilted artificial constraint) or pick tasks where the per-bug fix itself requires holding many fragments in attention simultaneously. The latter is rare; the former is unrepresentative of production deployments.

6.2 The authority-disambiguation noise floor

Attempt 6 surfaced the most subtle finding: when a wedge mechanism *does* work in principle, a confounding noise floor can still mask it. The 12-CVE two-source extraction task carried both context-overflow (≈120K tokens of fetched advisory content) and authority-disambiguation (NVD vs GHSA panels frequently disagree). The judge log showed that the single-agent's primary failure mode was the latter, not the former. Multi-agent decomposition does not help disambiguate between two authorities — each sub-agent independently faces the same choice and resolves it by sampling its own prior, producing an aggregate noise floor in both single and multi runs.

The design implication is sharp: **for a wedge mechanism to surface cleanly in measured data, the task must be free of confounding noise floors that affect single and multi equally.** An empirical 0.05 gap is consistent with two interpretations — "the wedge mechanism barely exists" and "the wedge mechanism is large but masked by authority noise." Without the authority-stripped variant (attempt 7), the two are observationally indistinguishable.

6.3 The unified-authority pivot in attempt 7

Attempt 7's design choice — collapse the dual-authority NVD/GHSA setup to GHSA-only — is the operationalisation of the section-6.2 insight. Per-field deterministic rules (`severity` uses GHSA's `Moderate`, not NVD's `Medium`; `cvss_base_score` reads GHSA's CVSS panel, not NVD's; `affected_versions` uses pip-style `||` per the SwarmBench convention; `fix_commit_or_pr` deterministically prefers PR URLs over commit URLs when GHSA References lists both) eliminate the authority-choice surface entirely. The remaining failure mode is *rule-application decay*: as the single agent walks down the 16-item list applying the same 10 rules, fidelity per rule per item degrades.

The step-function model in Figure 3 is a defensible first approximation. It is consistent with the *relative* per-entry failure pattern observed in attempt 6's judge log (the agent applied the GHSA-vocabulary severity rule correctly to the first 4 CVEs and drifted to NVD-vocabulary `Medium` on later entries). It is *not* a calibrated prediction in the statistical sense — k=n=1 means we cannot bound the variance — but it is a generative mechanism that produces the centred-0.36 gap prediction by integration over a step function whose breakpoints are themselves anchored to attempt 6 observations.

This is the structural argument the assessment's Section-8 rubric requires: a defensible *why* for the predicted gap, not an empirical *what*.

6.4 Comparison to published wedges

The three SwarmBench example tasks with published gap data:

Example task	Domain	Verifier	Published gap
AGENTBENCHLANDSCAPE	knowledge-research	llm-judge	0.53
MEDICALRESEARCH	knowledge-research	llm-judge	0.46
VENDORCROSSREF	data-analysis	llm-judge	0.22

Zero of three published wedges come from `verifier_type = executable` with `domain = code-swe`. All three use LLM-as-judge over many independent items, where each item requires fetching and structuring a discrete external document. This is the same wedge mechanism that attempts 6–7 attempt to apply to code-flavoured content (security advisories instead of research

papers). The example code-SWE tasks (FastMCP, NetBox, Django zoneinfo) ship as templates *without* published gap data, which is consistent with the SwarmBench team having validated wedges on knowledge-research/data-analysis families and not on executable code-SWE.

This is the load-bearing comparison of the paper. The empirical pattern in our seven attempts is *consistent* with the published-gap pattern: wedges emerge under context-overflow on independent items with structured per-item extraction; wedges collapse under tool-augmented code-SWE.

---

7. Limitations and threats to validity

The following limitations should be borne in mind when reading any of the conclusions above.

1. **Trial counts.** All measured runs are at k=n=1. The reward variance under this regime is large (informally, observed run-to-run variance on stable Kimi inference is in the ±0.05 range for llm-judge tasks). Falsification claims are robust to this variance only when the predicted gap is structurally infeasible (e.g. single = 1.000 caps the gap at zero regardless of multi-agent variance).
2. **Single model.** Findings are conditional on Kimi K2.5. They may not generalise to GPT-4.1, Claude 4.7, Gemini 2.5, or smaller open-weight models. In particular, models with weaker tool-use fidelity may pressure the codebase-size axis more than Kimi does.
3. **Single harness.** Findings depend on Harbor's task structure and on kimi-cli's tool surface (`Read`, `Grep`, `Edit`, `Bash`, web-fetch). A harness that withholds `Grep` would change the answer to attempt 5 substantially. The "tool surface decouples LOC from attended-token pressure" claim is harness-specific in detail though probably general in spirit.
4. **Attempt 7 is predicted, not measured.** The 0.36-centred gap in Section 5.8 is a design-stage argument grounded in the step-function model of Figure 3 plus the authority-noise subtraction observed in attempt 6. It is not a measured score. Section 8 of the assessment doc explicitly waives single/multi runs and grades the design argument; the paper reports the predicted gap as predicted, not as evidence for the wedge.
5. **Selection bias on codebase size.** Attempts 1, 4, 5 chose progressively larger codebases (5K → 16K → 303K LOC). Codebase size is therefore confounded with version, language idioms, bug-fix-PR conventions, and bug independence. The "303K LOC still produces single = 1.000" finding is robust as a *necessary condition for falsification of H1*, but a fully-randomised trial would be required to nail a strict cause-effect statement.
6. **Verifier confound.** Attempts 1–5 use executable AST verifiers; attempts 6–7 use llm-judge. The wedge mechanisms are differently sensitive to verifier choice (executable verifiers give zero

partial credit for almost-right code; llm-judge gives proportional credit). The cross-attempt comparison conflates verifier and task family.

---

8. Conclusion

Across seven trials engineered against the Turing SwarmBench Code/SWE assessment, multi-agent decomposition on Kimi K2.5 produced no measured > 0.38 reward gap on executable code-SWE tasks (attempts 1–5) and a small but visible 0.05 gap on a two-source LLM-judge advisory-extraction task (attempt 6). The seventh attempt redesigned the task around a single authority and submitted a design-stage gap argument predicting a 0.36-centred wedge under the assessment's oracle-only validation rule. Three pre-registered hypotheses were tested; H1 (codebase size) and H2 (bug count) were falsified for the conditions tested; H3 (per-item structured research) was partially supported in mechanism but masked by authority-disambiguation noise in the only measured trial.

The headline finding is taxonomic: **multi-agent decomposition is rate-limited by *which bottleneck dominates the task*. It cures context overflow on independent items; it does not cure capability ceilings, authority-disambiguation noise, or retrieval precision.** All three published SwarmBench wedges are LLM-judge tasks over many independent items; none come from executable code-SWE. This is the structural analogue of the empirical pattern across the seven attempts.

For benchmark engineers: the multi-agent advantage is *engineered*, not stumbled upon. The task must isolate context-overflow as the rate-limiting bottleneck, eliminate confounding noise floors, and choose item independence as a design property rather than a happy accident.

For production multi-agent system designers: before adopting a swarm pattern, ask whether the bottleneck of your task is context-overflow on independent items. If it is not — if the bottleneck is capability ceiling, retrieval precision, or single-source-of-truth disambiguation — adding agents will spend your token budget without moving the metric.

---

9. Reproducibility statement

All artifacts are open. Executable apparatus and per-attempt task packages live in:

- **Submission repository:** `github.com/ericgitangu/swarmbench` (private during the assessment review window; mirror at `swarmbench-submission/` on the author's machine).
- **Harbor framework:** `github.com/harbor-framework/harbor` [6] (open source).

- **Per-attempt task packages:** the six attempts are zipped under `_engineering_journey/` of the submission directory; raw verifier outputs are in `_engineering_journey/raw-logs/`. Attempt 7 ships as `8357f9d342999a921066bf3208aac27e-SWARMBENCH-FANOUT-CODESWE-GHSA-UNIFIED-PYTHON-CVE-AUDIT-V7/` with a SHA-256-named directory enabling integrity verification.

Reviewers wishing to re-run any single or multi trial will need their own Fireworks API key (the author's was rotational and is not distributed); the oracle path is API-key-free thanks to the AGENT-BENCHLANDSCAPE judge's equality short-circuit and can be reproduced offline given Docker.

The iteration log (`_engineering_journey/iteration-log-and-wedge-math.md`) is the primary methodological narrative; per-attempt result.json files under `task/<folder>/execution_logs/<mode>/` carry verbatim Harbor output. Raw judge justifications for attempt 6's single and multi runs are at `_engineering_journey/raw-logs/cve-{single,multi}-justification.log`.

References

[1] Wu, Q., Bansal, G., Zhang, J., Wu, Y., Zhang, S., Zhu, E., Li, B., Jiang, L., Zhang, X., & Wang, C. (2023). *AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation*. arXiv:2308.08155. https://arxiv.org/abs/2308.08155

[2] Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., et al. (2023). *MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework*. arXiv:2308.00352. ICLR 2024. https://arxiv.org/abs/2308.00352

[3] Du, Y., Li, S., Torralba, A., Tenenbaum, J. B., & Mordatch, I. (2023). *Improving Factuality and Reasoning in Language Models through Multiagent Debate*. arXiv:2305.14325. ICML 2024. https://arxiv.org/abs/2305.14325

[4] Turing. *SwarmBench QUALITY_GATE_PRD*. Internal assessment specification, retrieved from `spec/QUALITY_GATE_PRD.md` of the SwarmBench distribution accompanying the Code/SWE assessment, 2026. (Not publicly published; cited as primary specification artifact.)

[5] Moonshot AI. (2025). *Kimi K2: Open Agentic Intelligence*. arXiv:2507.20534. https://arxiv.org/abs/2507.20534. Model checkpoints at https://github.com/MoonshotAI/Kimi-K2.

[6] Harbor evaluation framework. https://github.com/harbor-framework/harbor (accessed 2026-04). Apache-2.0 / MIT (per repository LICENSE).

[7] Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., Zhang, S., Deng, X., Zeng, A., Du, Z., Zhang, C., Shen, S., Zhang, T., Su, Y., Sun, H., Huang, M., Dong, Y., & Tang, J. (2023). *AgentBench: Evaluating LLMs as Agents*. arXiv:2308.03688. ICLR 2024. https://arxiv.org/abs/2308.03688

[8] Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., & Narasimhan, K. (2024). *SWE-bench: Can Language Models Resolve Real-World GitHub Issues?* ICLR 2024. arXiv:2310.06770. https://arxiv.org/abs/2310.06770

[9] OpenAI Preparedness, in collaboration with the SWE-bench authors. (2024). *Introducing SWE-bench Verified.* https://openai.com/index/introducing-swe-bench-verified/. Verified subset of 500 instances.

[10] RUC-GSAI. (2025). *YuLan-SwarmIntell / SwarmBench: Benchmarking LLMs' Swarm Intelligence.* Gaoling School of Artificial Intelligence, Renmin University of China. https://github.com/RUC-GSAI/YuLan-SwarmIntell. Companion paper: arXiv:2505.04364. *Distinct project from Turing's SwarmBench despite shared name.*

[11] Pallets Project. *click 8.3.3 release.* https://github.com/pallets/click/releases/tag/8.3.3. (Bug list and parent SHA `052c006` per `_engineering_journey/iteration-log-and-wedge-math.md`.)

[12] Pallets Project. *werkzeug 3.1.7 release.* https://github.com/pallets/werkzeug/releases/tag/3.1.7

[13] Apache Software Foundation. *apache/airflow 3.2.1 release.* https://github.com/apache/airflow/releases/tag/3.2.1. Parent SHA `ceee6dc6e6c1bee8998bbd45035b78e2affe6b2b` (3.2.0).

[14] GitHub. *GitHub Security Advisories database.* https://github.com/advisories. Per-CVE GHSA entries cited in attempts 6–7 oracle.json under `tests/oracle.json`.

[15] National Institute of Standards and Technology. *National Vulnerability Database (NVD).* https://nvd.nist.gov. (Cited as the secondary source whose disagreement with GHSA produced the authority-disambiguation noise in attempt 6.)

[16] AGENTBENCHLANDSCAPE example task. *Cited as the load-bearing comparable wedge.* Located at `spec/example_tasks/agentbenchlandscape/` of the SwarmBench distribution; published gap value 0.53 documented in `spec/README.md`. (Not publicly published; cited as primary specification artifact.)