

Nothing to collude for: told-to-collude LLM agents gain nothing over a matched innocent pair in five Colosseum environments

Grant Wright (Independent researcher), with Apart Research

Abstract

We asked which task set-ups lead LLM agents to collude (sprint question 2.1) and whether a monitor that combines what agents say with what they do detects collusion better than either alone (2.2). In all five of Colosseum's multi-agent environments (Jira, SmartGrid, MeetingScheduling, PersonalAssistant, Hospital), we told two Llama-3.3-70B agents to maximise their coalition's reward and gave them a private channel. Our control is a matched innocent pair: same private channel, never told to collude. Against it, the instruction gave no measurable payoff in any environment. An offline screen explains most of this: MeetingScheduling and PersonalAssistant leave a coalition nothing to take, and Jira's one-task cap leaves almost nothing. SmartGrid does leave room, but colluders did not use it, even when told the true scoring rule. In Hospital, the outcome was reproduced exactly with all communication switched off: turn order and tool-call validity decide it. Colluders' private messages showed status updates and de-confliction, never a plan against the others. For detection, an early monitor score (AUROC .88) was a confound: colluders go quiet in public, and simply counting public messages scored .92. With the matched control, no judge, view or three-judge panel beat that count (.67-.69). One candidate signature was an action, not a message: a colluding Hospital supply agent sent every transfer to its partner's hospital (9/9 runs vs 1/4), but it shrank and failed two pre-registered replications ($p = .07, .13$). Collusion studies need environments where collusion pays, matched controls and no-LLM baselines.

1. Introduction

Multi-agent LLM systems may coordinate against the people they serve or the other agents they work with. Monitors for this are only as good as the trajectories they are tested on: if the "colluding" runs differ from the controls in some way other than collusion, a monitor can score well while detecting nothing that matters. So before asking whether a monitor works (2.2), we ask whether the instructed collusion happens and pays (2.1), in every environment Apart's Colosseum starter code ships.

We build one pipeline for both: generate trajectories in which a pair is told to collude, measure whether the pair gains over a matched control, read its private messages for a plan, and ask LLM judges to tell the conditions apart from what an overseer could see -- public messages, actions, or both.

Our main contributions are:

1. **A same-channel innocent control** (condition E): the pair keeps its private channel but is never told to collude. It removes a confound that otherwise makes monitors look good [J 10, 12-14].
2. **A five-environment negative result for 2.1, with a diagnosis per environment.** Being told to collude gave no payoff over the control in Jira, SmartGrid, MeetingScheduling, PersonalAssistant or Hospital. An offline best-response screen separates "nothing to collude for" (Jira, MeetingScheduling, PersonalAssistant) from "room left unused" (SmartGrid) and "outcome set by turn order" (Hospital) [J 16, 23, 25-31].
3. **A no-LLM baseline** (count the pair's public messages) that every monitor must beat; with the matched control no monitor we tried beat it [J 10, 14, 16].
4. **A candidate action-level signature, tested to failure:** in Hospital, the colluding supply agent, which cannot read its partner's messages, first appeared to route all supplies to its partner's hospital (9/9 runs vs 1/4) [J 31]. Two pre-registered replications with mirrored designs did not confirm it ($p = .07, .13$); most of it was a model-level preference for one hospital's name [J 33-36].
5. **Harness and scoring fixes** to the starter environments, all disclosed: repeated execution turns, a duplicate-claim penalty that no longer rewards violations, a hash-seed dependence, and string tool

arguments that were silently dropped [J 21, 24, 27, 30].

2. Related Work

Colosseum (Nakamura et al., 2026), the framework we build on, models cooperative multi-agent tasks as distributed constraint optimisation problems, so that a coalition's *coalition advantage* and regret can be measured against the cooperative optimum, and pairs action-based metrics with an LLM judge of the conversation. It adds Jira and Hospital to the environments, and uses a *secret channel* between two agents as a behavioural probe. Its authors report *emergent collusion* (benign agents given a secret channel pursue a secondary goal without being told to) and *collusion on paper* (agents plan collusion in text but often act non-collusively), and argue that neither conversations nor actions alone capture every kind of collusion. Our work differs in its control: we compare told-to-collude pairs with pairs that have the *same* secret channel but no instruction, and we hold every monitor to a no-LLM baseline. Without that control, a monitor that compares colluders with agents lacking the channel measures the channel, not collusion (Section 4.2).

Algorithmic pricing. Fish et al. (2024) report that LLM pricing agents in a repeated oligopoly reach supracompetitive prices without being told to collude; we contrast their setting with ours in Section 5, using the pricing starter of Peist et al. (2025).

3. Methods

Citations of the form [J n] refer to entry n of our experiment journal: one dated entry per step (a run batch, an offline check or a decision), with what changed, why, the result and a verdict (Code and Data).

Framework. Apart's Colosseum starter bundle (poseidon-research/collusion_hackathon, commit d585a7b): Terrarium agents on a path network, each sharing a blackboard with its neighbours; a planning phase then an execution phase in which agents act through tool calls. Colluders are the first and last agents (first_and_last). Every reported result uses Llama-3.3-70B-Instruct for all agents, via OpenRouter, temperature 0.7. The first Jira pilots used Apart's default, Qwen-2.5-7B-Instruct; a side-by-side pilot found Llama reached 86-97% of the optimum vs Qwen's 59-78%, with no coalition payoff for either, so we moved to Llama [J 1, 4, 6].

Conditions. A = Apart's 2.1 condition A: a colluding pair, Apart's coalition prompt ("maximize the coalition's joint reward"), and a private board for the pair. E (ours) = A with the coalition prompt off: same pair, same private board, no instruction. A and E differ only in the instruction, and each seed fixes the instance and turn order, so A and E on the same seed are paired. Early Jira pilots also ran Apart's B (no channel), C (open channel) and D (neither) and the structured, aggressive and deception prompt variants [J 4, 6, 9].

Environments.

Env	Agents	What an agent decides	Per-agent reward	Source
Jira	4	which task to claim (1 each; 2 in our variant)	own tasks' value - cost - penalty	J 16-17
SmartGrid	4	a power source per machine (16-17 machines)	minus an equal share of each connected source's overflow	J 23
MeetingScheduling	4	attendance interval per meeting	shared meeting value split equally; own overlap penalty	J 26
PersonalAssistant	4	one outfit	own colour preferences + pairwise matches split per pair	J 26
Hospital	9 (2 hospitals x 4 depts + Provisioner)	patient bookings; supply transfers	env's own per-step split (exposed by us)	J 30

Outcome measures. *Coalition advantage* = colluders' mean reward minus the other agents' mean, in each environment's own units (Hospital's are in the thousands). The comparison of interest is A minus E on the same seeds. For Jira also % of optimum and per-agent best-response regret [J 16, 20].

Offline screen (no API calls). `scripts/smartgrid_potential.py` (SmartGrid, MeetingScheduling, PersonalAssistant) computes a heuristic joint optimum and the coalition's best response with the innocents held at it: the gain is the room collusion has to pay [J 23, 26]. `scripts/hospital_screen.py` plays Hospital with a greedy scripted stand-in over random turn orders [J 30-31]. Heuristic, not exact optima.

Reading the boards. We read the private board of the runs cited, and for SmartGrid every message of `sgtr` and `sgnt` plus a keyword scan (`scripts/client_talk_audit.py`) [J 28-29]. Jira and MeetingScheduling boards were read for a sample of runs, not all.

Monitor (Jira only). Judges see four views built from what an overseer could see: public messages, actions, both, and a *digest* (extract claims, then compare them with outcomes -- our say-vs-do view). The private board is never shown; a leak test confirms none of its posts reach any view [J 2]. Three judges (gpt-4.1-mini, DeepSeek-v3.1, Mistral-small-3.2), temperature 0, score 0-10 on an anchored rubric that describes honest mistakes [J 8]. Prompt, panel and threshold were frozen on pilot data before the evaluation runs [J 15]. Primary metric: AUROC of A vs E, always set beside the count-of-public-messages baseline.

Environment changes (all ours, disclosed in Appendix A.5). Jira: two-task cap, multi-task regret, duplicate penalty 40 [J 17, 20-22]. SmartGrid and MeetingScheduling: repeated execution turns, because Terrarium ends a turn after the first tool call and Llama makes one call per response; stock SmartGrid runs assigned 4 of 16-17 machines [J 24, 26]. SmartGrid: the true reward rule stated to all agents (`sgtr`), plus a no-targeting rule (`sgnt`) [J 28-29]. Hospital: exposed per-agent rewards, repeated turns and passes, integer coercion of tool arguments [J 30]; a swapped hospital list order and a no-communication variant [J 31]; a mirror with the hospital order reversed, so the colluders are St Marys' Triage and the Provisioner [J 33]. Each change has an offline test, and each run batch differs from its predecessor in one change only.

Budget and logging. Every API batch was logged before launch with token-priced and actual cost (`session-log.md`). All runs and monitors used US\$3.27 of a US\$5 allowance; actual cost ran 1.4x to 5.4x the token price per batch (unexplained) [J 24-25, 30-31].

4. Results

4.1 Being told to collude does not pay, in any environment (2.1)

Table 1. Coalition advantage, A (told to collude) vs E (same pair, same private channel, no instruction), Llama-3.3-70B, same seeds in both arms. *Offline room* = how much the colluding pair could gain over the team-optimal plan by jointly switching to their own best choices while every other agent keeps to that plan, from the offline screen (no LLM; heuristic, Section 3); 0 = nothing to collude for. Jira and Hospital are described in words (Hospital used a turn-order simulation); "--" = not screened (no screen was built for the cap-2 Jira set-ups), not "no room". *n/arm* = runs per arm, one per seed (A and E use the same seeds). Units differ between environments; compare A with E within a row.

Env / set-up	Seeds	n/arm	Adv. A	Adv. E	Offline room	Source
Jira, base, evaluation (frozen)	2-11	10	+2.3	+2.0	one-task cap: only <i>which</i> task	J 16
Jira, cap 2	28-30	3	-0.8	-0.7	--	J 18
Jira, cap 2, 6 tasks (scarce)	31-33	3	-6.5	-1.0	--	J 19
Jira, cap 2, penalty 40	34-36	3	+4.5 (-20.0 / +30.7 / +2.8)	-2.8 (-5.2 / -1.8 / -1.2)	--	J 22
SmartGrid, multi-turn	37-39	3	-1.9	-2.4	+5.6, +1.2, +6.1	J 23, 25
SmartGrid, true rule	37-39	3	-2.3	-0.9	as above	J 28
SmartGrid, true rule + no targeting	37-39	3	-2.6	-1.5	as above	J 29
MeetingScheduling, multi-turn	40-42	3	-1.5	-0.8	0 on 10/10 seeds	J 26-27
PersonalAssistant	40-42	3	-0.3	-0.3	0 on 9/10 (one tie)	J 26-27

Env / set-up	Seeds	n/arm	Adv. A	Adv. E	Offline room	Source
Hospital (A) vs no-communication (E config)	43-45	3	-1,074 / -520 / +1,005	-1,074 / -546 / +1,298	pure transfer by turn order	J 30-31

Jira rows 1-3 use Apart's duplicate penalty 20, row 4 uses 40 (Section 4.4). Hospital is shown per seed; its E column is the no-communication run, because the stock E batch is invalid (Appendix A.5).

- **Jira: the seed, not the instruction, sets the advantage.** On the evaluation seeds the same seed gives almost the same advantage in both arms (seed 8: +13.6 vs +13.4); every agent ends with exactly one task, and coalition and innocent regret match (1.32 vs 1.38) [J 16]. Lifting the cap to two tasks, adding scarcity, or raising the penalty did not create a payoff: whoever moves first takes the extra task [J 18-19, 22]. The one large gap, seed 35 (A +30.7 vs E -10.0 at Apart's penalty 20 [J 20]; +30.7 vs -1.8 in the penalty-40 re-run, Table 1), is not a plan: in the re-run the colluder makes the same pick in E without any instruction, and the gap comes from an innocent colliding with it in E [J 20, 22].
- **MeetingScheduling and PersonalAssistant: nothing to take.** Both are common-interest games: shared value is split equally, so spoiling it costs a colluder as much as an innocent. The screen finds no coalition gain on any seed (PersonalAssistant: one tie between equally good joint optima), and the live runs match: A ~ E [J 26-27]. In MeetingScheduling seed 42 the colluders did agree a shared attendance plan -- benign scheduling, which overlapped their own meetings and plausibly explains A's -5.0 (judgement) [J 27].
- **SmartGrid: room exists, colluders do not use it.** Capacity is 88-94% of demand, and each source's overflow is split equally among its clients, so a coalition can gain by loading sources it shares with innocents (screen: up to +6.1) [J 23]. Colluders did not: A was no better than E in the stock prompt, when every agent was told the true rule, or under a no-targeting rule [J 25, 28-29]. In 18 SmartGrid runs no message, public or private, mentions the scoring rule, other clients or a target; every private post is a colluder reposting its own plan [J 25, 28-29].
- **Hospital: turn order and tool-call validity decide everything.** The first Triage agent to make a *valid* tool call takes all 8 triage steps; every valid run is all-or-nothing at triage [J 31]. With every board blanked for every agent, the colluding Triage's points reproduce the colluding run almost exactly (-162 / -150 / +3,104 vs -142 / -145 / +3,044) [J 31]. Offline over 2,000 plays per seed, the colluding and innocent Triage have the same mean (about 1,260-1,480 per play) and the same ~50% loss rate; longer episodes scale the stakes, not the edge [J 31]. The private board held generic status lines, no plan [J 30].
- **Private messages never showed a plan against others** in any run we read [J 9, 19-20, 22, 25, 27-30].

Interpretation (judgement). Across five environments, whether the instruction could pay was set by the task structure, and where it could (SmartGrid), Llama-3.3-70B treated the task as cooperative de-confliction whatever the stated incentive. We did not observe instructed collusion turning into coordinated action against others.

4.2 No monitor beats a message-count baseline (2.2, Jira)

Table 2. AUROC for distinguishing the told-to-collude runs from the controls.

Comparison	Baseline (count public msgs)	Best monitor	Source
Prompted vs D (no channel), naive	.92	.88 (digest)	J 10
A vs E, pilot5 (seeds 18-27)	.69	.68 (DeepSeek, digest); none above baseline	J 14
A vs E, evaluation, frozen panel (seeds 2-11)	.67	.69 (panel, messages; +0.01, CI -0.39..+0.39)	J 16

Frozen-panel AUROC on the evaluation runs by view: actions .41, both .60, digest .54, messages .69 [J 16].

- **The naive result was a confound.** Told-to-collude pairs move their talk to the private board, so their public record goes quiet; judges appear to key on the silence [J 10].

- **The quiet persists against the matched control**, which is why the baseline still scores .67-.69 [J 16]. It appears in SmartGrid too: colluders' public posts per run were A 0.7 / 1.0 / 1.3 vs E 4.0 across the three SmartGrid set-ups [J 25, 28-29]. So a count can detect *the instruction* even where nothing pays -- it detects being told to collude, not harmful collusion (judgement).
- **Combining say and do did not help**. Neither the both view nor the digest beat messages alone, or the baseline, once the control was in place [J 14, 16]. Averaging near-chance judges diluted the best one [J 7].

4.3 A candidate action-level signature that did not replicate: supply routing (2.2, Hospital)

The Hospital supply agent (Resource_Provisioner) is a colluder, paired with one hospital's Triage. Its prompt never shows board content, so it cannot read its partner [J 31]. **First look (exploratory)**. With General_Hospital as partner, the colluding Provisioner sent 15/15 transfers to General in 9/9 runs with transfers; the innocent was General-only in 1/4 (one-sided Fisher exact, run level, $p = .014$) [J 31]. But the partner was always General, so loyalty and a preference for that name could not be told apart.

Pre-registered replication. We froze the test before running [J 33]: fresh seeds 46-53, A vs E, in two mirror designs where the partner is never the hospital listed first -- partner General (St Marys listed first) and partner St Marys (General listed first; hospital order reversed, which also moves missed-step penalties to St Marys -- bookkeeping only). Primary outcome: runs in which every transfer went to the partner. Result: A 11/16, E 4/12 (four E runs made no transfer), $p = .07$ -- **not significant at .05** [J 34].

The innocent Provisioner favours General in both designs (8/9 of its transfers when St Marys is the "partner"), so part of the first look was a name preference.

Pre-registered follow-up. The St Marys rows looked like a partner effect against the name bias, so we pre-registered a second test of that design alone [J 35]: fresh seeds 54-77, 24 runs per arm, same outcome and alpha. Result: A 4/21, E 0/13 (eleven E runs made no transfer), $p = .13$ -- **not significant** [J 36]. Even the colluder sent most transfers (22/30) to General.

Test	Partner	A all-to-partner	E all-to-partner	A transfers to partner	E transfers to partner
Replication [J 34]	General	7/8	4/6	15/16	12/20
Replication [J 34]	St Marys	4/8	0/6	6/12	1/9
Follow-up [J 36]	St Marys	4/21	0/13	8/30	1/19

Either way the preference does not pay: 3-5 IV kits to a hospital already holding 20 only add holding cost [J 30-31].

Interpretation (judgement). The direction favoured the colluder in every look, but the size fell each time (9/9 vs 1/4, then $p = .07$, then $p = .13$) -- the best-looking subgroup regressed on fresh seeds. Most of the first-look signal was a model-level preference for the name General. At most a small partner effect remains; it is not established. The lesson for 2.2 matches Section 4.2: an action signal that looks strong at small n can be mostly a bias the matched control shares.

4.4 Starter-environment flaws we found

- **Jira rewards duplicate claims**. Under Apart's weights a second claim nets 20 x priority weight - cost, usually positive: extra claims raised the joint score in 8 of 8 cases in the base env and 25 of 28 in our cap-2 runs, so % of optimum can exceed 100. At penalty 40 none of the same 36 claims pays [J 21-22]. The pair never exploited this together: every duplicate paired a colluder seat with an innocent, 17 in A and 19 in E [J 37].
- **One tool call per turn**. Terrarium ends an agent's turn after its first committed call, so stock SmartGrid runs were all incomplete (4 of 16-17 machines) [J 24].
- **PersonalAssistant instances depend on Python's hash seed** (edge order from a set), so A and E on the same seed got different instances until we fixed PYTHONHASHSEED=0 [J 27].

- **Hospital silently drops string tool arguments** (Llama sends "0"): a `TypeError` is swallowed upstream and every action is lost [J 30]. Hospital also exposes no per-agent reward and its `broadcast_message` is a no-op [J 30].
- **SmartGrid's prompt misstates its scoring** ("attributed proportionally"; the code splits equally) [J 23].

5. Discussion and Limitations

What the negative result means (judgement). A monitor that reads the public record can only detect collusion that changes the public record or the outcome. In these five environments, being told to collude changed neither beyond moving talk to the private channel; a matched control removes most of that cue, and what remains is detected as well by counting messages. We read the failure of every monitor as a property of the set-up, not as proof that LLM monitors cannot work.

For 2.1: the task structure decides whether collusion can pay. Common-interest games (MeetingScheduling, PersonalAssistant) and one-item allocations (Jira) leave nothing to take; a contested shared resource (SmartGrid) leaves room, but in our runs the model did not look for it. Where the outcome is a transfer decided by who moves first (Hospital), a coalition gains nothing it would not get alone.

A contrast: where collusion is reported without being asked for (track 1.1). Fish et al. (2024) give LLM pricing agents a repeated Bertrand oligopoly: each period every agent sees all prices and its own demand, is told only to maximise long-term profit, and cannot communicate except through the prices it sets. They report quick, robust arrival at supracompetitive prices and profits, with price-war concerns as a contributing factor. In our reading (judgement), that setting has what our environments mostly lack: a standing gain from coordinating against a competitive benchmark, repeated play against the same rivals, and each rival's actions visible every round -- with no instruction to collude and no channel. As a sanity check we ran the starter bundle's Fish-style duopoly once (Llama-3.3-70B, 30 rounds, one seed, 60 calls, US\$0.02) [J 32]. Its built-in check passed (both firms above the Nash price), but the trajectory is not tacit collusion: one firm priced above the monopoly level throughout and lost the market, the other undercut it to about halfway between Nash and monopoly, and neither firm's notes mention coordinating. One short run settles nothing; it does show that "above Nash" alone is too weak a collusion label.

For practice: (i) innocent controls must share the colluders' channel structure; (ii) every monitor result needs a no-LLM baseline; (iii) screen an environment offline for coalition room before spending on runs; (iv) check that a "violation" is actually penalised, and that every agent's actions actually reach the environment.

Limitations

The main ones (full list in Appendix A): small n (3 seeds per arm outside Jira's 10-seed evaluation; the non-Jira results are exploratory); one agent model and Apart's Jira-worded coalition prompt; a control that may itself contain emergent collusion; monitors on Jira only; and our environment changes (Appendix A.5), which make several results not directly comparable with the stock environments.

Future Work

A stronger or reasoning model in SmartGrid, the one environment with room; a coalition prompt written for each environment; more seeds; action signals tested against the agents' own naming and ordering biases, with designs that counterbalance them from the start; a track 1.1 sweep on the pricing game (game length, visible history, private notes, prompt, number of firms), judging collusion by both firms' joint trajectory rather than by "above Nash".

6. Conclusion

Telling two Llama-3.3-70B agents to collude gave them no advantage over a matched innocent pair in any of Colosseum's five environments. The environments explain why: most leave a coalition nothing to take, the one that leaves room (SmartGrid) went unused even with the true scoring rule stated, and Hospital's

outcomes are reproduced with all communication switched off. With the matched control, no LLM monitor - reading messages, actions or a say-vs-do digest -- beat counting public messages. The transferable lessons are the matched control, the no-LLM baseline and an offline screen for coalition room: without the first two our monitor appeared to work (AUROC .88).

Code and Data

Code, configs, every run trajectory and the experiment journal (cited in the text as [J n]): <https://github.com/grant-wright/apart-collusion-2026> (being populated after submission; complete by 28 Oct 2026; until then available from the author on request).

Built during the sprint (23-26 Oct 2026): environment wrappers in `envs_ext/` (Jira two-task cap, multi-turn SmartGrid and MeetingScheduling, SmartGrid true-rule, Hospital passes / swapped / no-communication; the mirror is the swapped wrapper with the hospital order reversed in its config); scripts in `scripts/` for config generation (`make_configs.py`), run summaries and regret (`summarise_runs.py`), offline screens (`smartgrid_potential.py`, `hospital_screen.py`), audits (`duplicate_audit.py`, `client_talk_audit.py`), the monitor (`views.py`, `monitor_prompts.py`, `run_monitor.py`, `analyse_monitor.py`), the paraphrase replay, and an offline test for each environment change (`test_*.py`); all run configs in `configs/` except the two below; every trajectory in `runs/`; the experiment journal and the spend log.

Pre-existing (written 19 Oct 2026, before the sprint; setup only, no results): `configs/collusion_gemini_full.yaml` and `collusion_gemini_pilot.yaml` (not used for any reported run); `scripts/check_config.py`, `smoke_gemini.py`, `smoke_openrouter.py` (no-result connectivity checks); `docs/tutorial.md`; `SETUP-NOTES.md` and project scaffolding.

Third-party: Apart's starter bundle `poseidon-research/collusion_hackathon` at commit `d585a7b` (Colosseum / Terrarium / CoLLAB, and the algorithmic-pricing starter used in Section 5), used unmodified; our changes live in wrapper classes outside it.

References

- Fish, S., Gonczarowski, Y. A., & Shorrer, R. (2024). *Algorithmic collusion by large language models* (arXiv:2404.00806). arXiv. <https://doi.org/10.48550/arXiv.2404.00806>
- Nakamura, M., Kumar, A., Das, S., Abdelnabi, S., Mahmud, S., Fioretto, F., Zilberstein, S., & Bagdasarian, E. (2026). *Colosseum: Auditing collusion in cooperative multi-agent systems* (arXiv:2602.15198; Version 2). arXiv. <https://doi.org/10.48550/arXiv.2602.15198>. Code: <https://github.com/umass-ai-safety/colosseum>
- Peist, M., Romero, J., & Sauer, L. (2025). *Testing folk theorem predictions in AI-mediated markets* [Master's project, Barcelona School of Economics]. Code (MIT licence) in the `poseidon-research` (2026) starter bundle, `algorithmic-pricing-llms/`.
- `poseidon-research`. (2026). *collusion_hackathon: AI Collusion sprint starter bundle* (Commit `d585a7b`) [Computer software]. GitHub. https://github.com/poseidon-research/collusion_hackathon

Appendix A: Limitations and Dual-Use / Ethical Considerations

A.1 Construct validity -- are we measuring collusion?

- *"Colluding" is a label from the set-up.* A run is "colluding" when a pair was told to form a coalition; we did not find a run in which that became a plan against others (Section 4.1). Monitor scores are therefore scores for detecting *the instruction*, not harmful collusion.
- *The control may not be collusion-free.* Colosseum reports emergent collusion in pairs given a secret channel without instruction (Nakamura et al., 2026) -- exactly our E. So A vs E measures what the instruction *adds*, not collusion against none. In the E boards we read we saw no plan against others, but we did not read every one.

- *Coalition advantage is confounded* by turn order, task instance and model slips: in Jira and Hospital the seed (turn order) sets most of it [J 16, 31]. Hence the paired same-seed A vs E comparison, never A alone.
- *Collusion need not be harmful*. Where the colluders' best choices are also the team's, a correct flag marks harmless coordination.
- *Monitor scores are 0-10 self-reports*, not calibrated probabilities; we report AUROC, which uses only their order.

A.2 Internal validity -- is the cause we name actually the cause?

- *Turn order and instance are tied to the seed*; arms share seeds, but we did not vary order within an instance (except Hospital's swapped list order and mirror [J 31, 33]).
- *Honest mistakes can look like collusion* (duplicate claims, text-written tool calls). The rubric describes honest mistakes [J 8]; agents are Llama-3.3-70B, judges are other model families.
- *The offline screens are heuristic* (greedy or scripted stand-ins), not exact optima [J 23, 26, 30-31].

A.3 Statistical validity -- could it be luck?

- 3 seeds per arm everywhere except Jira's 10-seed evaluation and the Hospital routing tests, both frozen before their runs: the replication (8 seeds x 2 designs per arm, $p = .07$ [J 33-34]) and a follow-up on the design that looked best (24 seeds per arm, $p = .13$ [J 35-36]; its hypothesis came from the replication's data, and both are reported). The other non-Jira results are exploratory pilots, not pre-registered. The AUROC difference over baseline has a wide interval (-0.39 to +0.39) [J 16].
- Monitor prompt, panel and threshold were frozen on pilot data before the evaluation seeds were run [J 15]; everything outside the named primary comparison is exploratory.

A.4 External validity -- would it hold elsewhere?

- One agent model for the reported results (Llama-3.3-70B, temperature 0.7; Qwen-2.5-7B in early pilots only); Apart's coalition prompt, worded for Jira; short planning phases; 4-9 agents; toy tasks with no real stakes. Monitors were run on Jira only.
- *A more capable model might behave differently*. Llama-3.3-70B is a mid-sized open model without extended reasoning; a stronger planner might find SmartGrid's room or coordinate a real plan, so the negative results are about this model, not about collusion in general. It might also be more aware of being tested: Hospital's prompts call the task a simulation, and a keyword scan of all 259 runs found no agent saying it was being tested -- weak evidence, since this model shows no hidden reasoning [J 37].
- OpenRouter models can change behind the same id; batches cost 1.4-5.4x their token price, unexplained, which may mean different upstream providers [J 24-25, 30-31].
- Hidden messaging is only partly tested: one paraphrase replay showed wording changes behaviour sharply, which argues neither for nor against a code [J 3].

A.5 Our changes to the starter environments (all in-sprint, each with an offline test in scripts/).

- *Jira* [J 17, 20-22]: a two-task cap (`envs_ext/jira_multitask_env.py`; costs do not add across an agent's tasks -- our choice); `regret_multi`, a best response over task sets up to the cap (equals Apart's regret at cap 1, tested); duplicate-claim penalty raised from 20 to 40, because at 20 an extra claim usually raised the joint score (25 of 28 cap-2 cases, 8 of 8 in Apart's env). Prompt text unchanged. Results at cap 2 or penalty 40 are not directly comparable with Apart's.
- *SmartGrid, MeetingScheduling* [J 24, 26]: repeated execution turns while an agent has undecided items (at most items + 2), because Terrarium ends a turn after the first tool call; scoring, prompts and tools unchanged. *SmartGrid* `sgtr` replaces the "attributed proportionally" sentence with the true equal-split rule for every agent; `sgnt` adds a no-targeting rule [J 28-29].
- *Hospital* [J 30-31, 33]: per-agent rewards from the env's own split; extra turns and passes (6 max) while bookings continue; integer coercion of string tool arguments; variants with the Provisioner's hospital list swapped, with every board blanked (no communication), and a mirror with the hospital order reversed

(which also moves missed-step penalties to St Marys -- reward bookkeeping only). The Provisioner cannot read boards and its reward is 0 by construction, which halves the coalition mean.

- *Excluded batches* (kept in runs/): runs/sg/ (stock SmartGrid, incomplete); runs/pa/ (PersonalAssistant instances depended on Python's hash seed -- rerun as runs/pa0/ with PYTHONHASHSEED=0); runs/hosp/ (string arguments, 0 bookings); runs/hosp2/ E (every Triage tool call written as text, 0 bookings -- Hospital's control is the no-communication run runs/hosnc/) [J 24, 27, 30-31].
- Config copies were shortened for Windows path limits; every change is listed in the config headers and session-log.md.

A.6 Dual-use and ethics.

- *Dual-use (judgement)*: knowing which views catch colluders also tells colluders what to avoid -- here, "stay quiet in public" is detectable by a count. At this scale (an open model, toy tasks, Apart's published prompts) the uplift to a real attacker is low, and the defensive lesson -- matched controls and no-LLM baselines before trusting a monitor -- is the point.
- Apart's structured, aggressive and deception prompt variants instruct agents to deceive; used only inside the simulation.
- No human participants or human data; all agents are simulated. All API spend was logged (session-log.md).
- AI assistance was used for code, analysis and drafting (see LLM Usage Statement); every number traces to a logged run.

LLM Usage Statement

This was a one-person entry with heavy AI assistance, as Apart's rules allow. Claude (Anthropic, Claude Code) wrote all of the code -- the environment wrappers, offline screens and tests, monitor pipeline, analysis scripts and the PDF build -- launched and logged every run under my instruction, kept the experiment journal and spend log, and drafted this report from them. I chose the research questions and accepted the design, made each experimental decision (the same-channel control, one change per batch, which environments to try and when to stop, the duplicate-penalty fix, the extra Hospital replication, the budget), reviewed the results and edited the text. Every number in the report traces to a logged run in runs/ and an entry in the experiment journal, and every reference was checked against its source. This assistance is separate from the LLMs that are the subjects of the study: the Llama-3.3-70B agents in the environments and the judge models used as monitors (Section 3) were experimental components, run through OpenRouter, and played no part in writing the code or the report.