
Detecting Collusion with Pairwise Net Influence: A Baseline Comparison in Toy Multi-Agent Games¹

Soumya Nayak
PhD student at Wayland Baptist University

With
Apart Research

Abstract

Detecting which AI agents are secretly coordinating is a safety problem for multi-agent systems. Bonjour et al. (2022) proposed detecting collusion with mutual information between agents' actions, including a measure called pairwise net influence, which subtracts the strongest outside influence on each agent. I implemented this measure for the fully observable simultaneous case and tested whether it identifies the colluding pair better than plain correlation (Cramér's V), a baseline the paper does not include. I simulated four-agent Rock-Paper-Scissors with one hand-crafted colluding pair whose strength is set by a parameter p . My setup differs from the paper in several ways: four players instead of three, a different collusion rule, no poker or learned agents, and ranking pairs by the smaller of the two directional scores instead of applying a detection threshold. I tested four scenarios (no decoy, hub decoy, chain decoy, separate decoy pair) with 200 trials per setting and 95% Wilson confidence intervals. Both methods ranked the planted pair first almost every time once collusion was strong enough (for example $p \geq 0.6$ against a separate decoy of strength 0.5, with 500 rounds), and both performed poorly when collusion was weak. Across 7,200 trials the two methods chose different pairs in only 49 (0.7%), mostly in the hub and chain scenarios, where both were wrong in 34 of the 40 disagreements. I found no evidence that net influence outperforms plain correlation in this simple setting.

¹ Research conducted at the [AI Collusion Research Sprint](#), October 2026

1. Introduction

As interacting AI agents deploy, a critical failure mode is covert collusion—such as the sub-competitive price-setting observed in LLM agents (Fish et al., 2024). For auditors limited to action logs, Bonjour et al. (2022) propose an information-theoretic detector. Their individual influence $\gamma(i;j)$ computes the mutual information between the joint actions of agents i and j , while their net influence $\Gamma(i;j)$ subtracts the strongest individual influence any third agent exerts on j . Pairs are flagged if both directional net influences exceed a threshold. This framework assumes exactly two colluders and lacks validation against plain correlation benchmarks or LLM agents.

This work evaluates whether net-influence subtraction adds value over standard action correlation in a fully observable, simultaneous-action setting. I implement pairwise net influence in a four-agent Rock-Paper-Scissors game with one planted colluding pair, ranking pairs by their minimum directional score.

My contributions are:

- **Implementation:** Pairwise net influence for fully observable, simultaneous games, ranking pairs by their minimum directional score.
- **Benchmark Evaluation:** A rigorous comparison against a Cramér’s V baseline across four structural scenarios (no decoy, hub decoy, chain decoy, separate decoy pair), utilizing 200 trials per setting with 95% Wilson confidence intervals.
- **Error Analysis:** A targeted analysis of trial disagreements, revealing no evidence that net-influence subtraction outperforms plain correlation in this toy setting.

2. Related Work

While collusion detection is established in auctions (Porter and Zona, 1993), stock trading (Palshikar and Apte, 2008), and traditional games (Laasonen and Smed, 2015), detecting emergent collusion in modern AI agents (Fish et al., 2024; Nakamura et al., 2026) remains limited to internal probes (Rose et al., 2026) or mutual information-based action analysis (Bonjour et al., 2022). This work builds directly on Bonjour et al. (2022) but focuses strictly on fully observable, simultaneous-action settings. I expand their framework by introducing a Cramér’s V baseline to test if net-influence subtraction outperforms simple correlation, and evaluate structural robustness using decoy scenarios (hub, chain, separate pair) while ranking pairs by their minimum directional score rather than a threshold. Ultimately, this addresses the unverified gap of whether net influence improves upon basic correlation measures; in this toy setting, it did not..

3. Methods

Overview. I implemented the fully observable simultaneous case of the pairwise net influence detector from Bonjour et al. (2022) and compared it with Cramér’s V on simulated four-agent Rock-Paper-Scissors (RPS) games. Everything ran in Python (NumPy, Matplotlib) on a free Google Collab CPU with no API calls.

Game and simulators. Each game has 4 agents and n rounds. Moves are coded 0, 1, 2 (rock, paper, scissors), so $(m + 1) \bmod 3$ is the move that beats m . The simulator records moves only. It does not compute payoffs. By default, every agent plays uniformly at random. The colluding pair is always agents 2 and 3. Agent 3 plays the move that beats agent 2’s move with probability p (collusion strength) and plays randomly otherwise. I used four scenarios:

Scenario	Extra structure (strength q)
Basic	None
Hub decoy	Agents 1 and 2 copy agent 0 with probability q
Chain decoy	Agent 1 copies 0, and agent 2 copies 1, each with probability q
Separate decoy	Agent 1 plays the move that beats agent 0 with probability q , so the decoy pair (0, 1) shares no agent with (2, 3)

In the hub and chain scenarios, the decoy links share an agent with the real pair. This is where subtracting outside influence could change the ranking.

Detector. For two move sequences x and y , I estimated the joint distribution from move counts and computed mutual information in bits, which gives $\gamma(i;j)$. With no hidden state, this is the paper’s Equation 6. I then computed $\Gamma(i;j) = \gamma(i;j) - \max_{k \neq i, j} \gamma(k;j)$ (the paper’s Equation 4) for every ordered pair. The paper declares collusion when $\Gamma(i;j)$ and $\Gamma(j;i)$ both exceed a threshold α . I instead scored each pair by $\min(\Gamma(i;j), \Gamma(j;i))$, which passes the threshold only when both directions do, and ranked all six pairs.

Baseline. Cramér’s V is computed from the 3×3 table of move counts as $\sqrt{\chi^2 / (n \times 2)}$. I chose it because RPS moves are categories, so Spearman correlation does not apply. I rejected a same-move agreement rate because agent 3 never plays the same move as agent 2 when colluding, so that rate would miss the collusion.

Evaluation. A trial succeeds if the method’s top-ranked pair is (2, 3). Chance is $1/6$. Each setting used 200 trials with seeds 0 to 199. I report hit rates with 95% Wilson score intervals. I ran three experiments:

1. p swept from 0.0 to 0.8 in steps of 0.1 in all four scenarios, with $q = 0.5$ and 500 rounds.
2. Rounds swept over 100, 250, 500, 1000 and 2000 in the separate-decoy scenario, with $p = 0.6$ and $q = 0.5$.
3. A disagreement analysis of the trials where the two methods chose different pairs. For hub and chain at $q = 0.5$, I classified each disagreement as net influence right only, Cramér’s V right only, or both wrong. I repeated this for hub at $q = 0.3$ and $q = 0.7$.

What did not work. My first hub and chain settings (for example $p = 0.3$, $q = 0.5$ to 0.7) made the decoy links stronger than the real pair. Both methods then ranked a decoy pair first, so the comparison told me nothing. I moved to a p sweep that includes values where the real pair is the strongest link.

Reproducibility. All randomness comes from `np.random.default_rng(seed)`. The code and notebook are in the GitHub repository (add the link). Ties in ranking are resolved by Python’s `max`, which returns the first pair in pair order

4. Results

A trial succeeds if the top-ranked pair is the planted pair (2, 3). Chance is $1/6 \approx 0.17$. Each setting uses 200 trials, and both methods are scored on the same games.

4.1 Accuracy vs collusion strength

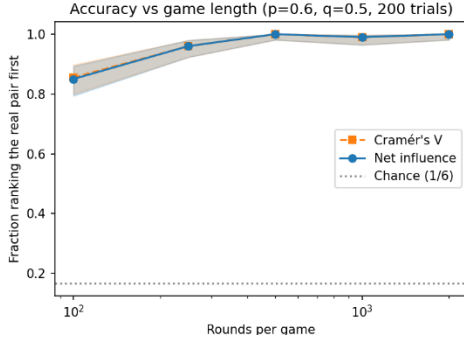


Figure 1: Fraction of trials in which each method ranked the real pair first, versus collusion strength p , in four scenarios (500 rounds, $q = 0.5$, 200 trials per point). Shaded bands: 95% Wilson CI. Dotted line: chance (1/6). With no decoy, both methods were at chance at $p = 0$ (0.14) and reached 1.00 by $p = 0.2$. With a decoy of strength $q = 0.5$, accuracy stayed near zero until the real pair was about as strong as the decoy, then rose sharply: in the separate-decoy scenario it was 0.01 at $p = 0.4$, 0.54 at $p = 0.5$ and 1.00 at $p = 0.6$. The two methods' curves overlap in all four panels. Both methods mostly pick up the strongest link, so both fail when a decoy is stronger than the real pair. This follows from the scenario design.

4.2 Accuracy vs game length

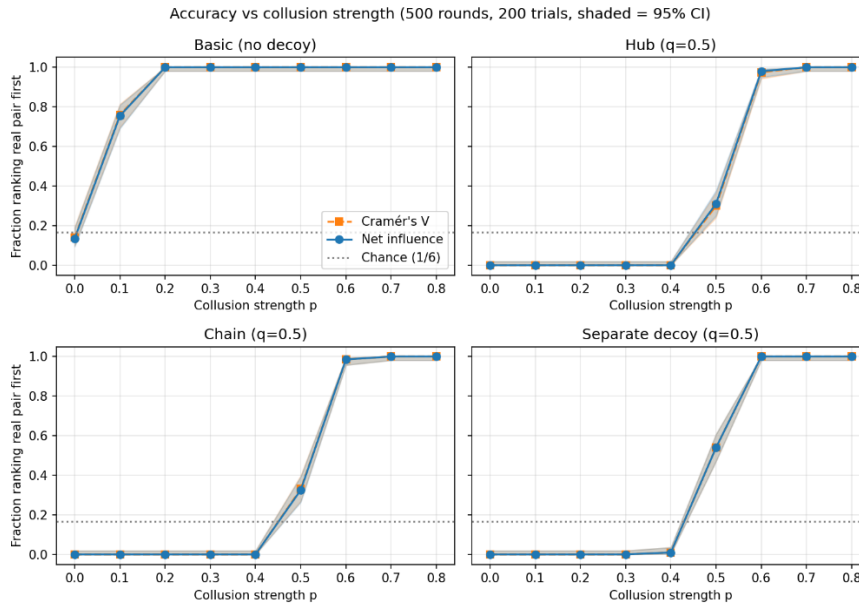


Figure 2: Accuracy versus rounds per game (separate decoy, $p = 0.6$, $q = 0.5$, 200 trials per point, shaded: 95% Wilson CI). Accuracy rose from 0.85 at 100 rounds to 0.96 at 250 and about 1.00 from 500 rounds on. The two methods had identical hit rates at every length and disagreed in only 3 of 1,000 trials in total, all at 100 rounds.

4.3 Where the methods disagree

Across four scenarios and nine values of p (7,200 trials), the methods chose different pairs in 49 trials (0.7%): 25 in hub, 15 in chain, 5 in basic and 4 in separate (1,800 trials each). A disagreement does not mean one method won. Table 1 classifies them.

<i>Setting</i>	<i>NI right only</i>	<i>CV right only</i>	<i>Both wrong</i>
<i>Hub, $q = 0.5$</i>	<i>3</i>	<i>0</i>	<i>22</i>
<i>Chain, $q = 0.5$</i>	<i>1</i>	<i>2</i>	<i>12</i>
<i>Hub, $q = 0.3$</i>	<i>0</i>	<i>0</i>	<i>4</i>
<i>Hub, $q = 0.7$</i>	<i>0</i>	<i>3</i>	<i>16</i>
<i>Total</i>	<i>4</i>	<i>5</i>	<i>54</i>

Table 1: Who was right when the methods disagreed (NI = net influence, CV = Cramér’s V). Each row pools $p = 0.0$ to 0.8 . The hub $q = 0.3$ and 0.7 rows are extra runs, so the total (63) exceeds the 49 above.

Most disagreements (54 of 63) are cases where both were wrong. Disagreements concentrate in hub and chain, where decoy links share an agent with the real pair, the only case where subtracting outside influence can change a ranking.

4.4 Robustness

With 200 trials per point, intervals at high-accuracy points are narrow (for example $[0.98, 1.00]$). The disagreement counts are small: a sign test on 4 vs 5 gives $p = 1.0$. Trials share seeds across p , so this is only a rough guide. I therefore cannot claim any difference between the methods. Equal performance held across four scenarios, five game lengths and three hub decoy strengths. I did not test other games, other numbers of agents, noisy or learned agents, or more than 200 trials per point.

5. Discussion and Limitations

In this toy setting, pairwise net influence and Cramér’s V found the planted pair equally often. Both mainly pick up the strongest dependence between agents, so both fail when an innocent decoy link is stronger than the real collusion. For AI safety, this suggests that detectors based on action dependence need a way to separate collusion from other strong dependence, and the net-influence subtraction did not provide that here. Both methods also needed enough rounds to work reliably.

Limitations

- **Toy setting.** I used scripted agents, one game (Rock-Paper-Scissors), four players and exactly one colluding pair. LLM agents or other games could change the result.
- **Assumed collusion and decoys.** Collusion was a simple same-round rule, and the decoys were my own constructions. Real collusion or innocent dependence that looks different could change the comparison.
- **Statistics.** 200 trials per point with seeds shared across p , and only 9 disagreements where exactly one method was right. A small real difference could exist.
- **Not a replication.** I did not reproduce the paper’s results, so I can only say my implementation follows its equations.

Future Work

Add a no-collusion condition with the threshold rule, test on LLM agents in a market or auction game and use more trials with independent seeds.

6. Conclusion

I implemented the pairwise net influence detector of Bonjour et al. (2022) for the fully observable simultaneous case and compared it with Cramér’s V on simulated four-agent Rock-Paper-Scissors with one planted colluding pair. Across four scenarios (no decoy, hub, chain and separate decoy), both methods ranked the real pair first almost every time once collusion was strong enough and enough rounds were available, and both failed when a decoy link was stronger than the real pair. The two methods chose different pairs in only 49 of 7,200 trials (0.7%), and in most of the disagreements I examined (54 of 63) both were wrong. I found no evidence that net influence outperforms plain correlation in this setting.

For AI safety, this suggests that detectors based on dependence between agents’ actions need an additional way to separate collusion from other strong dependence, such as shared incentives, before they can serve as reliable evidence. These results are limited to scripted agents, one game and one form of collusion, and I did not measure false alarms or test LLM agents. The next step is to add a no-collusion condition with the paper’s threshold rule and to test the method on LLM agents in a market or auction game.

Code and Data

- **Code repository:** <https://github.com/soumya-nayak-commits/collusion-detection>
- **Data/Datasets:** No external dataset was used. All data is simulated. Running python `run_experiments.py` regenerates them with fixed seeds (0 to 199).

References

1. Bonjour, T., Aggarwal, V., and Bhargava, B. (2022). Information theoretic approach to detect collusion in multi-agent games. *Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence (UAI)*, PMLR 180:223-232.
<https://proceedings.mlr.press/v180/bonjour22a.html>
2. Wilson, E. B. (1927). Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209-212.
<https://doi.org/10.1080/01621459.1927.10502953>
3. Cramér, H. (1946). *Mathematical Methods of Statistics*. Princeton University Press

LLM Usage Statement

I steered the project’s direction by treating Claude as an interactive technical advisor—asking targeted questions to guide development and draft code, while personally executing, validating, and interpreting every result.