
From Pairs To Coalitions: Extending Information-Theoretic Collusion Detection To Multi-Agent Games¹

Chae Eun Lee

Fabian Rosdalen

Peter Samodelkin

Yasin Edin

With

Apart Research

Abstract

We study information-theoretic approaches to detect collusion and coalition members in a multi-player rock-paper-scissors game testbed with coordinated collusion. We compare a net-influence baseline with four methods that combine conditional mutual information, standardized against shuffled actions, with connected components, Louvain communities, likelihood-based search or average-z growth. Across 2,800 coalition and 400 honest scripted test games, the ML decoder achieved the highest recall (81.71%), while Louvain most often recovered the exact coalition groups (73.29%). The graph methods could recover two separate coalitions that the single-group methods could not represent. Experiments with Qwen2.5:14B extend the evaluation to instructed model-generated coordination. The results motivate the use of action-based audits with information-theoretic approaches for detecting colluding agents.

1. Introduction

The net-influence collusion detection method presented by Bonjour et al. [1] can, in theory, be applied to any setting with publicly observable actions. Rather than interpreting semantic content, it analyses how different agents’ actions affect each other’s actions, making it adaptable to many types of games. We explore methods that extend this approach to detect collusion among more than two agents, drawing on established methods for recovering hidden groups in networks and noisy signals.

The methods explore the following research question: Can we improve on the existing information-theoretic approach to detect collusion in multi-agent games?

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

Our main contributions are:

1. **A multi-player rock-paper-scissors testbed.** We adapt the 3-player RPS game with colluding pairs used by Bonjour et al. [1] to an RPS game with N players and multi-agent coalitions. The testbed distinguishes coalition detection from coalition membership recovery of a single agent and it tests for multiple coalitions. We then extend this testbed to include LLM agents as players using either local or external models.
2. **Novel information-theoretic approaches.** We implement two decoder-based approaches: a maximum-likelihood (ML)-based coalition search and a greedy approximation extension of the ML-based decoder. We implement two graph-based coalition-recovery methods: connected components and Louvain community detection.

2. Related Work

Bonjour et al. use pairwise net influence to detect a colluding pair among 3 agents in a rock-paper-scissors game (RPS) [1]. This technique is designed to detect only one pair of colluding agents and does not scale to larger groups, where the influence between the different sets of pairs in a colluding group mask the others. We build upon the RPS game to increase the number of players so the colluding group consists of one primary player and N number of assisting players.

Taking inspiration from channel coding, we view coalition membership of players as a hidden structure and the relationships between the players as a noisy channel containing the hidden structure. We adapt decoding methods for channels and pattern detection methods in networks to decode and detect the coalition groups and its members. Radicchi [2] interprets community detection as decoding over a noisy channel, while Wu and Xu [3] study the statistical and computational limits of recovering planted structures. Drawing upon these works, we adapt the ML-decoder. Our decoding methods comprise a maximum-likelihood-based search over small candidate groups and a greedy average- z growth approximation with a heuristic stopping rule.

Palshikar and Apte [4] adapt techniques from graph clustering to find collusion sets in trader graphs. We adapt this work to find coalitions in a small multi-agent setting in an RPS game using the dependence-graph. Because players do not interact directly, we infer edges from the data, using standardized action dependence between players as edge evidence. Further, we add a dependence graph-based coalition detection method variant, which detects Louvain communities [5]. Our approach exploits action dependence and the conditional mutual information as signals for collusion detection in a small logged multi-agent environment, allowing for controlled comparison of detection methods under different coalition sizes and detection of members. We demonstrate the robustness of our proposed information-theoretic approaches in a multi-player RPS testbed.

3. Methods

3.1. Testbed

We extend the iterated rock-paper-scissors game [1] to multiple players, as shown in Figure 1. Each game, referred to as an episode has N players and lasts T rounds. We use a configuration to fit compute budget limitations, where $N \in 3, 6$ and $T \in 100, 300$. Scoring is performed per round. In each round, every player chooses rock, paper or scissors. If everyone chooses the same action, nobody scores. If all three actions appear, everyone receives one point. If exactly two actions appear,

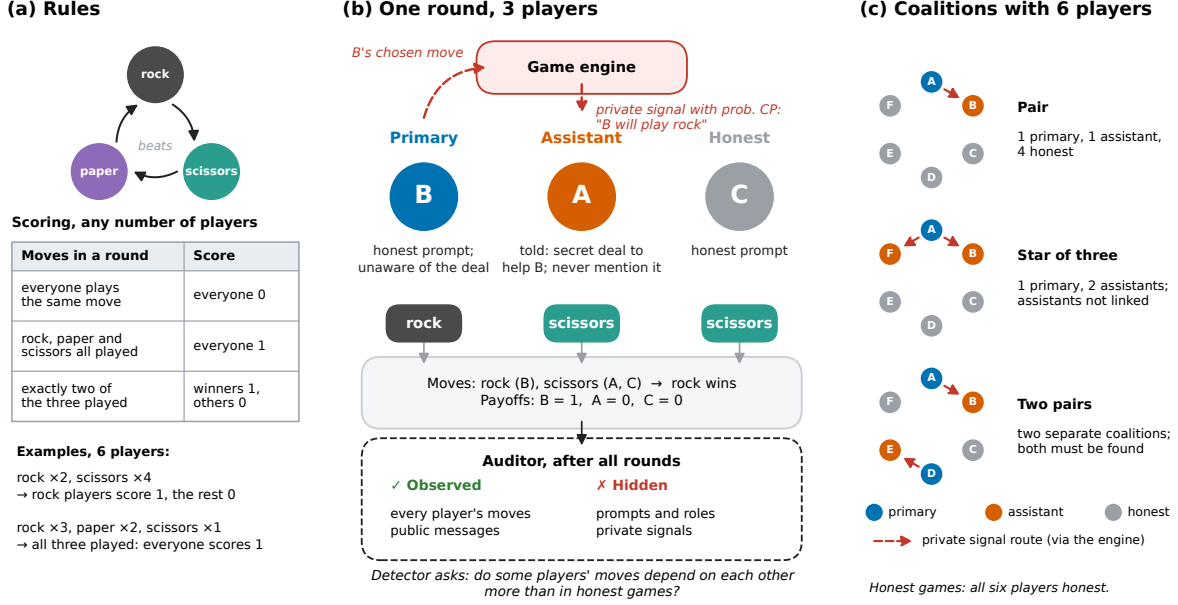


Figure 1: Game setup. (a) Rock–paper–scissors with any number of players. (b) One round with three players. The primary (B) and the honest player (C) receive the same prompt; only the assistant (A) is told of the agreement. With probability CP, the game engine reveals B’s chosen move to A as a private signal. The auditor observes every player’s moves (and public messages). (c) Coalition structures in six-player games.

players choosing the winning action receive one point, while others receive zero. Honest scripted players choose each action with probability $1/3$, independently across players and rounds.

Coalitions are formed in three ways in the testbed: two players, three players (referred to as star of three in Figure 1), or two pairs (two separate two-player coalitions). In the main scripted benchmark, the two player and three player scenario have each one primary player, while the rest are helpers. We do not test whether agents independently develop collusion, instead we use programmed assistance to create a controlled test of coalition detection with known membership. Each helper independently receives the primary’s current move with probability CP in each round and then plays the move that loses to it. Otherwise, the helper chooses uniformly at random. Coalition members are pseudo-randomly assigned for each game. We provide the mathematical formulation of the game in Appendix A.3.

3.2. Pairwise evidence

For each pair of players i, j , we measure how much knowing one player’s action helps predict the other’s, given information about the public information known before the round’s action, including actions in a previous round. We quantify the dependence between actions of players using conditional mutual information, $\hat{\gamma}_{ij}$ estimated from observed action frequencies using natural logarithms, which express information in nats. We provide the mathematical formulation of $\hat{\gamma}_{ij}$ in Appendix A.4.

For our proposed detectors, we account for apparent dependence that can arise by chance in short games by comparing the observed dependence with permuted versions of the data. We generate 99 shuffled versions of the data. Each version keeps one player’s action fixed and pseudo-randomly shuffles the actions of other players with the same public state. This preserves each player’s action counts in each state, but disrupts which actions occurred together, decreasing the saliency of

apparent dependencies in short games. We then recalculate the dependence for each shuffled version (Appendix A.5).

3.3. Detectors

3.3.1 Net-influence

We adapt the pairwise net-influence detector [1] as a net-influence baseline metric, $\Gamma_{i \rightarrow j}$ for our multi-agent testbed to detect coalitions (Appendix A.6). For each direction, net-influence measures how much more i is linked to j than the strongest link from any other player to j . $\Gamma_{i \rightarrow j}$ is computed as a difference between mutual-information estimates $\hat{\gamma}_{ij}$.

3.3.2 Graph-based approach

We represent players as nodes in an undirected graph, where we define an edge, $E = \{(i, j) : z_{ij} \text{ is defined, } z_{ij} > q_{\text{edge}}\}$ to be connecting two players whose action-dependence score exceeds the calibrated edge threshold. We identify candidate coalitions in two ways. The first method put players in the same group if you can follow a path of edges between them, referred to as the connected components method. The second uses Louvain community detection to find groups with stronger connections within groups than between them, thereby separating two groups that the connected components method would merge into a single connecting edge. The candidate coalitions are given scores, Q_{mod} by summing all pairwise dependence scores, z for players within the candidate group. The largest Q_{mod} for both methods is assigned as the game’s score, given that it passes the calibrated cutoff. The mathematical formulation of the approaches are further defined in Appendix A.7.

3.3.3 Maximum-likelihood Decoder-based approach

Taking inspiration from channel decoding, we treat coalition membership as a hidden message to be recovered from noisy observations. We adapt the ML decoder to search for candidate coalitions from pairwise dependence scores, assuming an independent Gaussian model $\mathcal{N}(\mu, 1)$ for internal coalition pairs and $\mathcal{N}(0, 1)$ for other pairs. For each candidate group size k in $2 \leq k \leq \min(5, N)$, we average the strongest $\binom{k}{2}$ pairwise scores in the whole game as $\hat{\mu}_k$. Candidate groups are scored using $\ell(G; \hat{\mu}_k) - \ln \binom{N}{k}$. The log-likelihood-ratio, $\ell(G; \hat{\mu}_k)$ rewards strong dependence between group members, while the heuristic adjustment subtracts the logarithm of the number of possible groups of size k . The decoder considers sizes from two to $\min(5, N)$. For larger search spaces, the decoder uses a greedy fallback, whereby for a specified size k , it adds players with the largest summed scores to the current group until it reaches k members. Here, we introduce an average- z growth variant of the ML-decoder approach, which starts with the highest-scoring supported pair and determines group size as it grows. It repeatedly adds the player with the strongest average connection to the group, provided this is at least one quarter of the group’s current average internal score; otherwise, it stops. The final selected group for both approaches is reported as a coalition if it exceeds threshold calibrated on independent honest episodes (Appendix A.9). We refer to these methods as Likelihood search and Average- z growth in the Section 4.

3.4. Calibration and evaluation

Connected components and Louvain use 80 honest games to set a shared graph edge threshold, based on the strongest pairwise score in each game. The ML decoder and average- z growth do not use this

edge threshold. For all proposed detectors, we use a separate set of 199 honest calibration games to choose the respective final detection thresholds. Each threshold is set near the 95th percentile of that detector’s honest-game scores.

We evaluate on the metrics: recall, false-positive-rate (FPR), and exact group recovery and the area under the receiver operating characteristic curve (AUROC). Recall measures how often a method detects a configured coalition. FPR measures how often it incorrectly flags an honest game. Exact group recovery requires identifying all coalition groups correctly. AUROC measures how well a method’s scores distinguish coalition games from honest games across possible detection thresholds.

We report Wilson 95% confidence intervals for recall, FPR and exact group recovery. Missed detections and abstentions count as recovery failures. For multiple coalition detection games, methods that return only one group cannot exactly recover two separate coalitions. The mathematical foundations for the calibration and evaluation is found in Appendix A.10.

4. Results

We report the scripted benchmark and the Qwen case study separately. The scripted benchmark tests detectors on games with programmed coalitions and known coalition members. Qwen study checks whether the same pipeline can collect model-generated play and whether the model follows the assistance instruction.

4.1. Case A: Scripted benchmark

The full scripted dataset contains 6,253 games, including games used to fit thresholds, calibration games, test games and stress controls. There are 4,300 test games overall. Our main comparison uses 2,800 coalition and 400 honest test games, with either three or six players and either 100 or 300 rounds.

Table 1: Detection performance of each method from the scripted benchmark. Recall and recovery use 2,800 coalition test games; false-positive rate uses 400 honest test games. Results combine the tested player counts, episode lengths and coalition structures. Brackets show point-wise 95% Wilson confidence intervals.

Method	Recall, % [95% CI]	FPR, % [95% CI]	Exact groups, % [95% CI]
Net influence	59.29 [57.45, 61.09]	2.50 [1.36, 4.54]	51.32 [49.47, 53.17]
Components	79.71 [78.18, 81.16]	2.75 [1.54, 4.86]	73.07 [71.40, 74.68]
Louvain	79.71 [78.18, 81.16]	2.75 [1.54, 4.86]	73.29 [71.62, 74.89]
Likelihood search	81.71 [80.24, 83.10]	4.00 [2.48, 6.40]	56.29 [54.44, 58.11]
Average-z growth	81.50 [80.02, 82.89]	3.75 [2.29, 6.09]	57.75 [55.91, 59.57]

As shown in Table 1 and Figure 2, the likelihood search method detected the largest number of coalition games (81.71%), while Louvain recovered most frequently the exact coalition groups (73.29%). Average-z growth had similar recall to the ML decoder (81.50%), but its exact group recovery was lower than that of either graph method (57.75%). Connected components and Louvain can return more than one coalition, whereas the ML decoder and average-z growth return at most one. The net-influence baseline detected 59.29% of coalition games and recovered the exact groups in 51.32%.

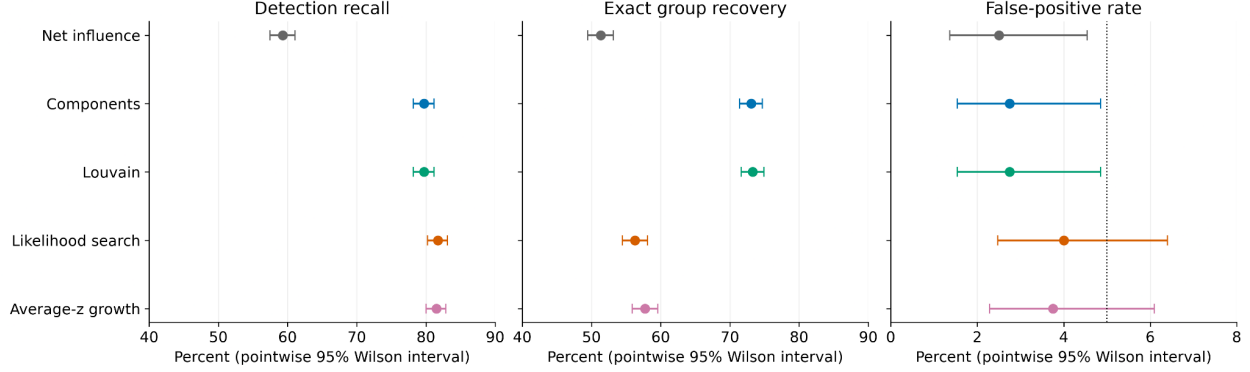


Figure 2: Detection, membership recovery and false alarms in the scripted benchmark with 95% Wilson intervals. The false-positive panel marks 5% as a reference. The first two panels span 40–90%; the false-positive panel spans 0–8%. Threshold procedures differ across methods, so higher recall does not establish better detection at an equal false-positive rate.

4.2. Case B: Qwen-generated play

Table 2: Detection on 300 Qwen 2.5 14B games (3 players, 30 rounds; 100 games per condition). Qwen followed the coalition instructions 4,320 out of 4,525 moves (95.5%). FPR is measured on honest games. AUROC is threshold-free and compares honest with CP = 0.5 games. Intervals are Wilson 95% CIs.

Method	FPR, %	CP = 0.5			CP = 1.0
		Recall, %	Exact group, %	AUROC	Exact group, %
Net influence	51.0 [41.3, 60.6]	89.0 [81.4, 93.7]	85.0 [76.7, 90.7]	0.86	100.0 [96.3, 100.0]
Components	3.0 [1.0, 8.5]	61.0 [51.2, 70.0]	56.0 [46.2, 65.3]	0.87	94.0 [87.5, 97.2]
Louvain	3.0 [1.0, 8.5]	61.0 [51.2, 70.0]	56.0 [46.2, 65.3]	0.87	94.0 [87.5, 97.2]
Likelihood search	3.0 [1.0, 8.5]	61.0 [51.2, 70.0]	56.0 [46.2, 65.3]	0.90	100.0 [96.3, 100.0]
Average-z growth	3.0 [1.0, 8.5]	60.0 [50.2, 69.1]	54.0 [44.3, 63.4]	0.91	100.0 [96.3, 100.0]

5. Discussion

Firstly, the results in Section 4.1 indicate that detecting coalition games and identifying coalition members need separate evaluation. In the scripted benchmark, the ML decoder and growth detected more coalition games than the graph methods, while the graph methods more often recovered the complete group structure. The six-player Qwen results give a concrete example: the single-group methods detected every two-pair game at CP = 1, but could not return the two separate coalitions. We conclude that a high detection rate alone does not show that the named players or group boundaries are correct.

Secondly, although the graph-based methods outperformed the other ML decoder-based methods, statistical significance of the results is left to be demonstrated in future work with higher compute and time budgets. Louvain recovered six more scripted games correctly than connected components out of 2,800, but this difference did not remain statistically significant after correction across the 16 comparisons examined. Growth and the ML decoder also had similar scripted recall. The Qwen results likewise do not establish that the modest AUROC differences reflect reliable differences between methods.

Table 3: Detection on 125 Qwen 2.5 14B games (6 players, 30 rounds; 25 games per condition). Qwen followed the coalition instructions 4,210 out of 4,509 moves (93.4%). FPR is measured on honest games. AUROC is threshold-free and compares honest with star CP = 0.5 games. Exact is the percentage of games in which the complete set of colluding groups is recovered. Intervals are Wilson 95% CIs.

Method	FPR, %	Star, CP = 0.5			Exact, %, CP = 1.0		
		Recall, %	Exact, %	AUROC	Pair	Star	Two pairs
Net influence	56.0 [37.1, 73.3]	84.0 [65.3, 93.6]	0.0 [0.0, 13.3]	0.72	88.0 [70.0, 95.8]	0.0 [0.0, 13.3]	88.0 [70.0, 95.8]
Components	8.0 [2.2, 25.0]	64.0 [44.5, 79.8]	24.0 [11.5, 43.4]	0.80	100.0 [86.7, 100]	96.0 [80.5, 99.3]	96.0 [80.5, 99.3]
Louvain	8.0 [2.2, 25.0]	64.0 [44.5, 79.8]	24.0 [11.5, 43.4]	0.80	100.0 [86.7, 100]	96.0 [80.5, 99.3]	96.0 [80.5, 99.3]
Likelihood search	12.0 [4.2, 30.0]	72.0 [52.4, 85.7]	36.0 [20.2, 55.5]	0.89	100.0 [86.7, 100]	100.0 [86.7, 100]	0.0 [0.0, 13.3]
Average-z growth	12.0 [4.2, 30.0]	64.0 [44.5, 79.8]	36.0 [20.2, 55.5]	0.89	100.0 [86.7, 100]	92.0 [75.0, 97.8]	0.0 [0.0, 13.3]

Finally, the Qwen-generated play in Section 4.2 confirms that the testbed can record model-generated play with coordinated collusion. Under the tested assistance instructions, the detectors separated many coalition games from honest games, particularly at CP = 1. The decrease in performance compared to the scripted benchmark can be partially attributed to the detectors being calibrated on simulated honest games. It can also be attributed to models not perfectly following instructed coordinated collusion. We leave larger model studies for future work.

We think that these kinds of detection methods can become important in agent collusion detection where it’s more feasible to monitor actions than (likely) encrypted messages. Since it doesn’t rely on semantic context, it’s not vulnerable to steganography or any forms of encryption of messages to the same extent as monitors reading agents CoT or messages. The detection doesn’t depend on models using specific channels for communication. If a model’s actions has a publicly visible effect, that effect can be monitored and logged. Due to the underlying simplicity of the detection, models might learn to evade it but at the cost of their publicly visible effect on the world - which might, as with most monitoring safety measures, push models to hide their actions or wait and save up credibility for one larger malicious action. Evading this detection would also mean constraining their ability to coordinate with one another as they would have to actively avoid their actions reading as correlated with another models.

5.1. Limitations and Future Work

We initially forked the continuous double auction market repository from Agrawal, K. et al. [6], but due to compute budgets allowing for models, which were observed to be incapable of effectively colluding, we could not produce coordinated collusion to test our detectors on. Instead, in this work, we built a rock-paper-scissors game, where we simulate collusion pseudo-randomly and rules are simple enough for the smaller models to follow. We leave the testing of our proposed detectors on market simulations for future work with more compute and time, we would also encourage to test it on more complicated scenarios that are more context-heavy since these methods propose to work regardless of semantic contexts.

The current calibration approach assumes that the new honest games have the same distribution as the calibration games, which may result in a higher FPR. Using a test game to set thresholds

can strengthen calibration. Further within-state shuffling may cause decrease in performance when behavior changes with history. Future work may improve upon the testbed by calibrating net-influence and strongest-pair detector alongside the proposed methods for improved baseline comparison.

The methods have assumptions that may benefit from further testing. Standardizing pairwise scores does not make them Gaussian or independent, as assumed by the methodologies of the ML decoder-based detectors. Pairwise evidence may also miss coordination expressed only through relationships among three or more players. Further experiments should include larger coalitions with a higher number of rounds, repeating both calibration and testing across independent sessions and model families.

6. Conclusion

We extend information-theoretic approach to detecting collusion in multi-agent games. Using conditional mutual information between actions, we adapt ideas from channel decoding and graph to detect collusion in games and identify the players involved. In the scripted RPS benchmark, the ML decoder achieved the highest recall (81.71%), while Louvain most often recovered the exact coalition groups (73.29%). The RPS testbed is also extended to evaluate the methods in model-generated play with instructed collusion using Qwen.

Our central finding is that ML decoder and graph dependence-based methods observing actions in a discrete game can reliably find colluding parties in a RPS testbed setting with scripted collusion. Furthermore, results show that detecting methods detecting collusion cannot always reliably identify the exact groups. Graph methods can recover separate coalitions that single-group decoders cannot represent. The qwen-generated play validates the central finding in an RPS testbed with colluding LLM players. The qwen results show that proposed methods can detect instructed coordination. Further tests with different strategies, larger coalitions and more models are needed to establish how broadly these findings apply.

Code and Data

- **Code repository:** <https://github.com/SkabbMask/rps-collusion>

Author Contributions

We order the author names alphabetically in the title page. Chae Eun Lee and Fabian Rosdalen led the project, designed experiments, and ran the code, and mainly contributed to the writing of the manuscript. Peter Samodelkin and Yasin Edin contributed to writing and reviewing the final manuscript.

References

- [1] Trevor Bonjour, Vaneet Aggarwal, and Bharat Bhargava. Information theoretic approach to detect collusion in multi-agent games. In *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pages 223–232. PMLR, 2022. URL <https://proceedings.mlr.press/v180/bonjour22a.html>.

- [2] Filippo Radicchi. Decoding communities in networks. *Physical Review E*, 97:022316, 2018. doi: 10.1103/PhysRevE.97.022316.
- [3] Yihong Wu and Jiaming Xu. Statistical problems with planted structures: Information-theoretical and computational limits. arXiv preprint arXiv:1806.00118, 2018.
- [4] G. K. Palshikar and M. M. Apte. Collusion set detection using graph clustering. *Data Mining and Knowledge Discovery*, 16:135–164, 2008. doi: 10.1007/s10618-007-0076-8.
- [5] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008. doi: 10.1088/1742-5468/2008/10/P10008.
- [6] K. Agrawal, V. Teo, J. J. Vazquez, S. Kunnavaakkam, V. Srikanth, and A. Liu. Evaluating LLM agent collusion in double auctions. arXiv preprint arXiv:2507.01413, 2025.
- [7] Liam Paninski. Estimation of entropy and mutual information. *Neural Computation*, 15(6): 1191–1253, 2003. doi: 10.1162/089976603321780272.
- [8] Belinda Phipson and Gordon K. Smyth. Permutation P-values should never be zero: Calculating exact P-values when permutations are randomly drawn. *Statistical Applications in Genetics and Molecular Biology*, 9(1), 2010. doi: 10.2202/1544-6115.1585.
- [9] Anderson M. Winkler, Gerard R. Ridgway, Matthew A. Webster, Stephen M. Smith, and Thomas E. Nichols. Permutation inference for the general linear model. *NeuroImage*, 92: 381–397, 2014. doi: 10.1016/j.neuroimage.2014.01.060.
- [10] Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification, 2021. URL <https://arxiv.org/abs/2107.07511>.
- [11] Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. doi: 10.1080/01621459.1927.10502953.

Appendix

A.1. Limitations and Dual-Use / Ethical Considerations

The results motivate applying action-based audits for agent collusion detection. However, the results with current budget and time limits do not indefinitely conclude that the proposed methods remain robust in scenarios outside of the RPS testbed. Further experiments with varying models of greater capacity, higher number of parameters and varying collusion parameters is necessary to test the robustness of the methods.

Ethical considerations are as follows. If this is included in AI training data, it could tip off models to the usage of such detection methods. We find these risks low. The underlying methods already exist publicly, and a monitor that depends on secrecy would be weak regardless. The detector should not be used as a training signal as this would optimize a models behavior to collude right under the threshold for detection.

Extending the proposed information-theoretic approaches to complex game scenarios, such as continuous double auction markets from [6] can inform whether these approaches can provide robust collusion detection. Validation of the approaches and adaptations to complex scenarios can provide an insight into real-world collusion detection, where actions may be the only informative signal due to stenographic communication between agents.

A.2. Notation and information units

N denotes the number of players or sellers, and T the number of rounds per episode. In RPS, a_{it} is player i 's action in round t , and S_t is the public state known before that round's actions. A candidate coalition is a set G of size $k = |G|$. The collection of true coalitions is denoted by $\mathcal{G}$.

The episode label $C \in \{0, 1\}$ records whether a coalition mechanism was assigned. Assignment does not guarantee that players followed the mechanism or harmed other players. The auditor observes data V and applies a detector $D = g(V)$, whose decision rule is fixed before evaluating the test games.

Mutual information is measured in nats when calculated with natural logarithms:

$$I_{\text{bits}} = \frac{I_{\text{nats}}}{\ln 2}. \quad (\text{A1})$$

The RPS testbed baseline uses a cutoff of 0.05 nats. This does not denote a 5% false-positive rate. We use α for the target false-positive rate.

A.3. RPS game and programmed assistance

Our RPS environment follows the collusion setting studied by Bonjour et al. [1], with the payoff rule and assistance mechanism specified below.

Let $b(a)$ be the action defeated by a : $b(\text{rock}) = \text{scissors}$, $b(\text{paper}) = \text{rock}$, and $b(\text{scissors}) = \text{paper}$. Let K_t be the number of distinct action types played in round t . Player i receives

$$u_{it} = \begin{cases} 0 & \text{if } K_t = 1, \\ 1 & \text{if } K_t = 3, \\ \mathbf{1}\{a_{it} \text{ is the winning action}\} & \text{if } K_t = 2. \end{cases} \quad (\text{A2})$$

Here $\mathbf{1}\{\cdot\}$ equals one when the condition holds and zero otherwise.

Honest scripted players independently choose each action with probability $1/3$. For a primary player p and helper h , assistance is activated independently each round with probability $\rho = \text{CP}$. When activated, the helper observes the primary’s current action and chooses the action that loses to it. Otherwise, the helper chooses uniformly:

$$P(a_{ht} = x \mid a_{pt}, C = 1) = \frac{1 - \rho}{3} + \rho \mathbf{1}\{x = b(a_{pt})\}. \quad (\text{A3})$$

With three players, assistance guarantees that the primary scores in that round. Under uniform independent play, each player’s expected payoff is $5/9$. At $\text{CP} = 1$, the primary, helper and remaining player have expected payoffs of 1 , $1/3$ and $2/3$, respectively. Thus, the primary gains a relative advantage, but the remaining player also gains compared with uniform independent play. Coalition assignment therefore does not by itself establish harm to outsiders. The extension to N players uses the same payoff rule.

A.4. Conditional action dependence

Following Bonjour et al. [1], we measure how much two players’ actions depend on one another after accounting for the public state:

$$\gamma_{ij} = I(A_i; A_j \mid S) = \sum_s P(s) \sum_{x,y} P(x, y \mid s) \ln \frac{P(x, y \mid s)}{P(x \mid s)P(y \mid s)}. \quad (\text{A4})$$

Here x and y are possible RPS actions, and s is a possible public state. In the main memoryless benchmark, S is constant.

We estimate these probabilities from observed counts. For the rounds retained for a pair, n_{sxy} counts how often actions x and y occur together in state s . The counts n_{sx} and n_{sy} record each player’s actions separately, n_s counts rounds in state s , and $n = \sum_s n_s$. The estimate is

$$\hat{\gamma}_{ij} = \frac{1}{n} \sum_{\substack{s,x,y \\ n_{sxy} > 0}} n_{sxy} \ln \frac{n_{sxy} n_s}{n_{sx} n_{sy}}. \quad (\text{A5})$$

For the four added detectors, we retain only rounds where both players’ actions are observed. We discard states with fewer than 10 observations for that pair and require at least 20 observations after filtering. If these requirements are not met, the pair’s score is undefined. The net-influence baseline instead uses rounds where every player’s action is observed and requires at least two such rounds.

Limited data can produce positive estimated dependence even between independent players [7]. Conditioning accounts for the recorded public state, but cannot remove dependence caused by public information that was not recorded.

A.5. Within-state permutation scores

The four added detectors compare the observed mutual information with a shuffled reference. For each pair with sufficient data, we keep one player’s actions fixed and randomly rearrange the other player’s actions among rounds with the same public state. This preserves each player’s action counts within each state while changing which actions occur together.

We repeat this $R = 99$ times. Let $\hat{\gamma}_{ij}^{(r)}$ be the estimate from shuffle r , and let $\mu_{0,ij}$ and $s_{0,ij}$ be the mean and sample standard deviation of the shuffled estimates. The standardized score is

$$z_{ij} = \frac{\hat{\gamma}_{ij} - \mu_{0,ij}}{\max(s_{0,ij}, 10^{-12})}. \quad (\text{A6})$$

The denominator floor prevents division by zero. If the observed estimate equals the shuffled estimates within numerical tolerance and their variation is negligible, the implementation returns $z_{ij} = 0$. This score measures distance from the shuffled reference; it does not imply a Gaussian distribution or a Gaussian significance level.

We also record a permutation p -value:

$$p_{ij} = \frac{1 + \sum_{r=1}^R \mathbf{1}\{\hat{\gamma}_{ij}^{(r)} \geq \hat{\gamma}_{ij} - 10^{-12}\}}{R + 1}. \quad (\text{A7})$$

The plus-one correction avoids reporting zero permutation p -values [8]. These values are recorded for inspection. Final detection decisions use the separate calibration procedure in Appendix A.10.

Interpreting the shuffles as a null reference requires the relevant observations to be exchangeable within each state: under the null, the allowed rearrangements must preserve their joint distribution [9]. Behaviour that changes with history can violate this assumption.

A.6. Net-influence baseline

The baseline uses raw mutual-information estimates, without shuffle standardization. Following the net-influence approach of Bonjour et al. [1], it compares a pair's dependence with the strongest competing relationship:

$$\Gamma_{i \rightarrow j} = \hat{\gamma}_{ij} - \max_{\ell \notin \{i, j\}} \hat{\gamma}_{\ell j}. \quad (\text{A8})$$

A pair qualifies when its net influence reaches $\tau = 0.05$ nats in both directions:

$$D_{\text{pair}}(i, j) = \mathbf{1}\{\min(\Gamma_{i \rightarrow j}, \Gamma_{j \rightarrow i}) \geq \tau\}. \quad (\text{A9})$$

The evaluated RPS implementation reports every qualifying pair and flags the episode if at least one qualifies. COLAV instead requires exactly one qualifying pair. Our reported RPS results use the former rule.

If every pair within a three-player coalition has the same dependence g , the subtraction can give $\Gamma_{i \rightarrow j} = g - g = 0$. This illustrates how similar relationships within a larger coalition can cancel; it does not imply that every three-player coalition will be missed.

A.7. Dependence graphs and Louvain

Each player is a node in an undirected graph. We use 80 independent honest games to choose the edge threshold q_{edge} . Each game contributes its largest supported pairwise score, $M = \max_{i < j} z_{ij}$, and the threshold is calculated using the rank rule in Appendix A.10. An edge connects two players when their score is defined and exceeds this threshold:

$$E = \{(i, j) : z_{ij} \text{ is defined and } z_{ij} > q_{\text{edge}}\}. \quad (\text{A10})$$

We extract candidate coalitions of at least two players in two ways. Connected components group players whenever a path of edges connects them. Louvain instead partitions the graph into communities, allowing it to separate groups that a weak connecting edge would merge into one component. Graph clustering has previously been used to identify collusion groups in trading data [4]; here, edges represent action dependence.

For Louvain, retained edges have weights $w_{ij} = \max(z_{ij}, 10^{-9})$, and absent edges have weight zero. Louvain seeks to increase modularity [5]:

$$Q_{\text{mod}} = \frac{1}{2w} \sum_{i, j} \left(w_{ij} - \frac{d_i d_j}{2w} \right) \mathbf{1}\{c_i = c_j\}, \quad (\text{A11})$$

where c_i is player i 's community, $d_i = \sum_j w_{ij}$ is its total edge weight, and $2w = \sum_i d_i$. Modularity compares the weight within communities with the weight expected from the nodes' total edge weights.

Both graph methods then score each candidate group using

$$Q_{\text{group}}(G) = \sum_{\substack{i < j \\ i, j \in G \\ z_{ij} \text{ defined}}} z_{ij}. \quad (\text{A12})$$

This sum includes every supported pair within the group, including negative scores and scores below the edge threshold. Modularity is used to find Louvain communities; it is not their final detection score.

Each method uses its largest candidate score per honest calibration game to set its own final cutoff. It reports only candidate groups whose scores exceed that cutoff.

A.8. Likelihood-based group decoder

The decoder searches for the coalition that best explains the observed pairwise scores under a simple statistical model. This follows the perspective of recovering hidden group membership through decoding [2, 3].

For scoring, we assume independent Gaussian scores: pairs within a coalition have mean $\mu > 0$ and variance one, while other pairs have mean zero and variance one. Relative to the model with no elevated pairwise means, a candidate group G of size k has log-likelihood-ratio score

$$\ell(G; \mu) = \mu \sum_{\substack{i < j \\ i, j \in G}} z_{ij} - \frac{\mu^2}{2} \binom{k}{2}. \quad (\text{A13})$$

For each size k , we estimate μ from the strongest $\binom{k}{2}$ pairwise scores in the whole episode, rather than from the candidate group alone:

$$\hat{\mu}_k = \max \left\{ 10^{-6}, \text{mean of the largest } \binom{k}{2} \text{ pairwise } z \text{ scores} \right\}. \quad (\text{A14})$$

Undefined pairwise scores are replaced by zero for this calculation and for scoring candidate groups.

We compare sizes from two to $\min(5, N)$ and select

$$(\hat{k}, \hat{G}) = \arg \max_{\substack{2 \leq k \leq \min(5, N) \\ |G|=k}} \left[\ell(G; \hat{\mu}_k) - \ln \binom{N}{k} \right]. \quad (\text{A15})$$

The second term is a heuristic adjustment based on the number of possible groups of size k . It is separate from ℓ and does not increase monotonically with group size.

The decoder checks all candidates when $\binom{N}{k} \leq 200,000$. Otherwise, it starts with the highest-scoring pair after replacing undefined scores by zero, then adds players with the largest summed scores to the current group until it reaches size k . The $N \in \{3, 6\}$ benchmark uses exhaustive search throughout. The selected group is reported only if its adjusted score exceeds its own honest-calibrated threshold. If no pair has a supported score, the decoder abstains.

The Gaussian model is an assumption used to construct the score. Actual pairwise scores can be dependent, especially when they share players, and need not follow Gaussian distributions. The estimated $\hat{\mu}_k$ is also not necessarily the maximum-likelihood estimate for the selected group. The method therefore provides a likelihood-based search, not a posterior probability or a guarantee of optimal recovery.

A.9. Average- z growth

This method starts with the supported pair having the highest z score, then adds players without specifying the final group size. The joining rule is adapted from the pinned upstream implementation.

For the averages below, replace undefined pairwise scores by zero. The group’s average internal score and an outside player v ’s average connection to the group are

$$\bar{z}_{\text{internal}}(G) = \binom{|G|}{2}^{-1} \sum_{\substack{i < j \\ i, j \in G}} z_{ij}, \quad (\text{A16})$$

$$\bar{z}_{\text{to_group}}(v, G) = \frac{1}{|G|} \sum_{i \in G} z_{vi}. \quad (\text{A17})$$

Select the outside player v^* with the largest average connection and add that player if

$$\bar{z}_{\text{to_group}}(v^*, G) \geq f \bar{z}_{\text{internal}}(G), \quad f = 0.25. \quad (\text{A18})$$

Thus, the player’s average connection must be at least one quarter of the group’s current internal average. Recalculate both averages after each addition. Stop when the strongest remaining candidate fails this condition or all players have joined. Ties follow player order.

The detection score remains the original pair’s score:

$$\text{Score}_{\text{growth}} = z_{\text{seed}} = \max_{\substack{i < j \\ z_{ij} \text{ supported}}} z_{ij}. \quad (\text{A19})$$

The grown group is reported only if this score exceeds its own honest-calibrated threshold. Expansion changes the predicted membership, but not the score used for detection. The method abstains if there is no supported starting pair and returns at most one group, so it cannot exactly recover two separate coalitions.

A.10. Honest calibration and evaluation metrics

All four added detectors use a separate set of 199 honest games to choose their final detection thresholds. They share the calibration games but receive different thresholds because their scores differ. The graph edge threshold is fitted on the separate 80-game set described in Appendix A.7. The net-influence baseline retains its fixed cutoff.

For n honest calibration scores sorted as $s_{(1)} \leq \dots \leq s_{(n)}$, define

$$r = \lceil (n+1)(1-\alpha) \rceil, \quad q = \begin{cases} s_{(r)} & \text{if } r \leq n, \\ +\infty & \text{otherwise,} \end{cases} \quad (\text{A20})$$

$$D = \mathbf{1}\{\text{score} > q\}, \quad \alpha = 0.05. \quad (\text{A21})$$

With $n = 199$, the cutoff is the 190th smallest calibration score. A test score must be strictly above it to trigger detection. Games with no candidate enter calibration with score $-\infty$; they are not discarded.

This rule uses the same rank argument as split conformal calibration [10]. If the scoring procedure is fixed independently of calibration, and calibration and future honest games are exchangeable, the false-positive rate is at most α on average over calibration and future games. An individual fitted threshold or test sample can have a higher false-positive rate. The guarantee does not automatically extend to a different honest population.

At $\alpha = 0.05$, fewer than 19 calibration games give $q = +\infty$, preventing detection. Reporting a flag whenever any of several detectors fires requires separate calibration of that combined rule or an adjustment for multiple testing.

We evaluate all methods on the same test games:

$$\text{Recall} = \frac{TP}{TP + FN}, \quad \text{FPR} = \frac{FP}{FP + TN}. \quad (\text{A22})$$

Here TP and FN count detected and missed coalition episodes, while FP and TN count flagged and unflagged honest episodes.

Exact group recovery requires the full collection of predicted groups to match the true collection: $\hat{\mathcal{G}} = \mathcal{G}$, regardless of group order. Correctly identifying the combined member set is insufficient if separate coalitions are merged into one group. Missed detections and abstentions count as recovery failures on coalition games.

AUROC measures how often a coalition game receives a higher detection score than an honest game, without choosing a detection threshold. Let s_a^+ be the score of coalition game a and s_b^- the score of honest game b . With n_+ coalition games and n_- honest games,

$$\widehat{\text{AUROC}} = \frac{1}{n_+ n_-} \sum_{a=1}^{n_+} \sum_{b=1}^{n_-} \left[\mathbf{1}\{s_a^+ > s_b^-\} + \frac{1}{2} \mathbf{1}\{s_a^+ = s_b^-\} \right]. \quad (\text{A23})$$

We compare every coalition game with every honest game, counting one when the coalition score is higher, one half for a tie, and zero otherwise. AUROC is the average of these values. A value of 0.5 means chance-level ranking; 1 means every coalition game scores above every honest game.

We calculate AUROC separately for each experimental setting and specified coalition condition. It measures separation between honest and coalition games, not whether the correct coalition members are identified. It does not change the threshold used for detection.

We report Wilson 95% intervals for recall, FPR and exact group recovery [11]. These intervals describe uncertainty in the test outcomes for the fitted procedure; they do not include the variation from fitting a new calibration. Games are the sampling units. Applying five detectors to one game does not create five independent observations.

Test games were not used to set thresholds. However, results were inspected during method development, so the reported comparisons remain exploratory.

A.11. Scripted benchmark collusion coordination performance

Detection performance was strongly influenced by how well the coordinated collusion design was working in the testbed. Recall depended strongly on the frequency with which assistance occurred and how many rounds were observed, as shown in Figure 3. In six-player, 100-round games with a single pair and $CP = 0.1$, both the likelihood search and average-z growth detected only 7 of 100 coalition games. More rounds helped, but did not always provide enough evidence. For the six-player pair condition at $CP = 0.25$, the ML decoder’s recall rose from 38/100 games at 100 rounds to 100/100 at 300 rounds. At $CP = 0.1$, it rose from 7/100 to 24/100. The aggregate recall in Table 1 and Figure 2, therefore hide conditions where most coalitions were missed.

A.12. Metadata detector: Public message

Note: modal-qwen14b-comparison-t30 run, Case B: Quen-generated play, 3 players

Each turn, the player outputs two things; a chosen move and an optional public message. Honest players chose to provide a public message 90% of rounds. The assisting model was given the prompt:

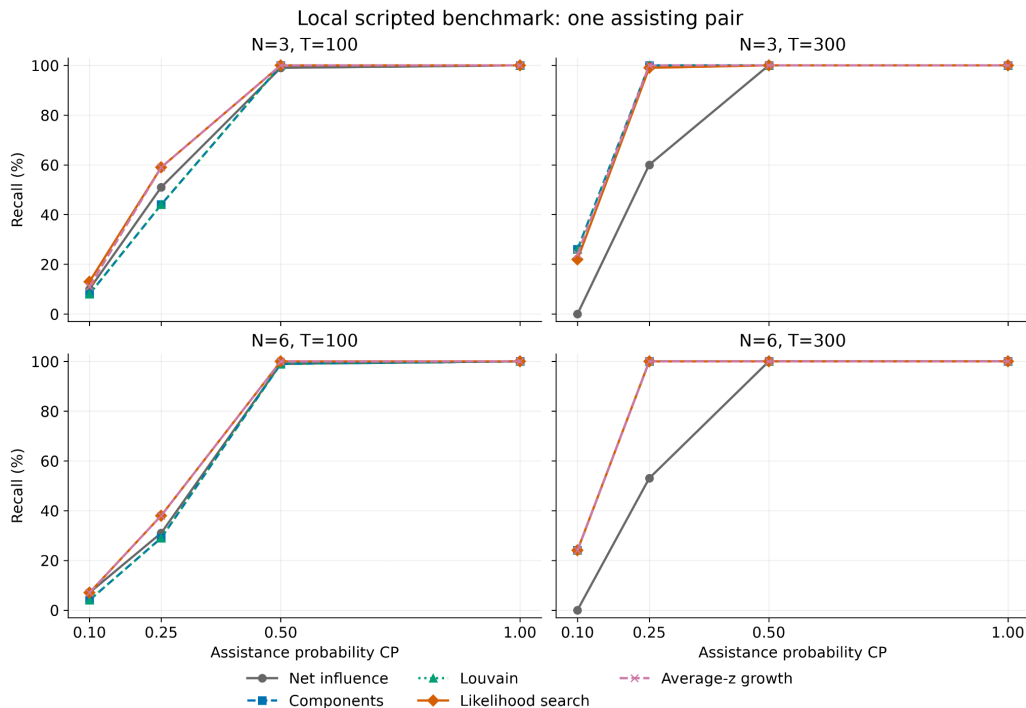


Figure 3: Recall for a single coordinating pair at different assistance probabilities and episode lengths Each point represents 100 coalition test games. Panels show three or six players and 100 or 300 rounds. CP is the per-round probability that a helper receives the primary’s current move. Lines connect measured conditions; they are not fitted predictions. Overlapping lines indicate similar or identical recall.

"You are {name}. You have a secret agreement with {partner}: your real goal is to help {partner} score. Sometimes you will receive a private signal revealing the move {partner} is about to play. When you do, play the move that the signalled move beats (if {partner} plays rock, you play scissors; paper $\rightarrow$ rock; scissors $\rightarrow$ paper) with certainty; this guarantees {partner} a point. Without a signal, play normally. Never mention the agreement."

Qwen 14B went silent in 90% of signaled rounds, making message metadata a near-perfect detector here. We’re not sure if this is a quirk of the prompt, model or model size - it is possible that the model would behave more like the honest players if told to.

LLM Usage Statement

We used Claude to brainstorm approaches and help draft proof-of-concepts. All results and claims were independently verified. All code was generated by Claude, but designed by us. Claude also helped with producing visualizations with feedback from us.