Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

Automating Collusion Detection in Sequential Games

Parisa Mazrooei and **Christopher Archibald** and **Michael Bowling**
Computing Science Department, University of Alberta
Edmonton, Alberta, T6G 2E8, Canada
{*mazrooei,archibal,mbowling*}*@ualberta.ca*

Abstract

Collusion is the practice of two or more parties deliberately cooperating to the detriment of others. While such behavior may be desirable in certain circumstances, in many it is considered dishonest and unfair. If agents otherwise hold strictly to the established rules, though, collusion can be challenging to police. In this paper, we introduce an automatic method for collusion detection in sequential games. We achieve this through a novel object, called a collusion table, that captures the effects of collusive behavior, i.e., advantage to the colluding parties, without assuming any particular pattern of behavior. We show the effectiveness of this method in the domain of poker, a popular game where collusion is prohibited.

1 Introduction

Many multi-agent settings provide opportunities for agents to collude. We define *collusion* to be the practice of two or more parties deliberately cooperating to the detriment of the other parties. While in some multi-agent settings such cooperation may be allowed or even encouraged, in many human-settings collusion is frowned upon (e.g., auctions), forbidden (e.g., poker), or even illegal (e.g., financial market manipulation). However, it is often hard to identify and regulate. If agents are not otherwise violating the established rules (e.g., sharing their own private information or the proverbial "ace up the sleeve"), policing is restricted to observing the actions of agents. These actions can be collusive in combination despite being individually legal.

In settings where collusion is prohibited or illegal, collusion detection can be a serious problem. Real-world collusion policing always relies on some form of human intervention. In online poker, participants who feel they have been colluded against may report their suspicions (PokerStars 2012). Such a report may result in an in-depth investigation by human experts. In the financial market setting, "detection of suspect trades" and "distinguishing manipulative from legitimate trading" is considered a major challenge (D'Aloisio 2010), that has been addressed in part by automated algorithms that identify suspicious patterns of behavior. For example, financial regulators employ automated surveillance systems that identify "trading activity which exceeds the parameters" of normal activity (Watson 2012). Human intervention in these systems comes in the form of comparisons with human-specified models of suspect behavior, and the systems can naturally only identify collusive behavior matching the model. These models are typically domain specific, and new models are required for any new domain addressed. Since any model of collusive behavior will be based on past examples of such behavior, new forms of collusion typically go undetected. As a community's size increases, detecting collusion places a huge burden on human experts, in terms of responding to complaints, investigating false positives from an automated system, or continually updating the set of known collusive behaviors.

In this paper, we present a novel approach to automating collusion detection and give proof-of-concept experimental results on a synthetic dataset. Our approach has three main advantages. First, we focus on the actual decisions of the participants and not just on secondary factors, such as commonly playing together in poker or high trading volume in markets. Second, our approach avoids modelling collusive behavior explicitly. It instead exploits our very definition that collusive behavior is jointly beneficial cooperation. Third, our approach is not specific to any one domain, and could be employed to any new problem easily.

Before going on, it is important to distinguish between collusion and *cheating*. We define cheating to be the use of actions or information that is not permitted by the decision and information-set structure of the underlying game-theoretic model of the interaction. For example, if two poker players share their private card information then this would be cheating, since the players have access to information outside of the information-set structure of the game. In contrast, collusion occurs when two players employ strategies that are valid in the game-theoretic model, but which are designed to benefit the colluding players. An example of this in poker might be a colluding player raising, regardless of private cards held, following any raise by a colluding partner. This would double the effect of the initial raise, forcing non-colluders out of the hand and securing the pot more frequently for the colluding pair. In this work we focus on the problem of detecting collusion, although we hypothesize that our techniques could work to detect cheating as well.

We proceed by giving in Section 2 the definitions and background necessary to understand the paper. The pro-

posed approach to detecting collusion, the main component of which is a novel data structure called a *collusion table*, is described in Section 3. The dataset used for our experimental results and its creation are described in Section 4, while Section 5 demonstrates the effectiveness of our approach on the experimental dataset. A discussion of related work is given in Section 6, followed by concluding remarks.

2 Definitions and Background

The focus of this work is detecting collusion in a known *zero-sum extensive-form game with imperfect information* (Osborne and Rubinstein 1994). Extensive-form games, a general model of sequential decision making with imperfect information, consist of a game tree whose nodes correspond to *histories* (sequences) of actions $h \in H$. Each non-terminal history, h, has an associated *player* $p(h) \in N \cup \{c\}$ (where N is the set of players in the game and c denotes chance) that selects an action $a \in A(h)$ at h. When $p(h) = c$, chance generates action a at h with fixed probability $\sigma_c(a \mid h)$. We call h a *prefix* of history h', written $h \sqsubseteq h'$, if h' begins with the sequence h. Each *terminal history* $z \in Z \subset H$ has associated *utilities* for each player $i \in N$, $u_i(z)$. Since the game is zero-sum, for all $z \in Z$, $\sum_{i \in N} u_i(z) = 0$. In imperfect information games, each player's histories are partitioned into *information sets*, sets of game states indistinguishable by that player. To avoid confusion, in what follows we will refer to a player in the game, as defined above, by the term *position*, corresponding to the index into the player set N. We use the term *agent* to refer to a strategic entity that can participate in the game as any player, or equivalently, in any position. For example, in chess, the two players/positions are black and white. A *positional strategy* (Section 4.2), is a strategy for only the black or white pieces, while an agent is able to participate in chess games as either black or white, and must therefore have a suitable positional strategy for each of the game's positions.

We are given a population of agents M who play the extensive-form game in question, and a dataset D consisting of *game episodes*. Each game episode is a record of a set of agents from M playing the extensive-form game a single time. Precisely, a game episode g is a tuple (P_g, ϕ_g, z_g), where $P_g \subseteq M$ is the set of agents who participate in game episode g, $\phi_g : P_g \mapsto N$ is a function which tells which agent was seated in each game position during g, and z_g denotes the terminal history that was reached during g. The latter term includes the entire sequence of actions taken by players and chance during the episode.

2.1 [2-4] Hold'em Poker

We experimentally validate our proposed approach using a small version of limit Texas Hold'em called three-player limit [2-4] Hold'em (Johanson et al. 2012) As in Texas Hold'em Poker, in [2-4] Hold'em each of between two and ten agents are given two private cards from a shuffled 52-card deck. In Texas Hold'em the game progresses in four betting rounds, which are called the **pre-flop**, **flop**, **turn**, and **river** respectively. After each round, public cards are revealed (three after the preflop, one after the flop and one after the turn). In each betting round, agents have the option to **raise** (add a fixed amount of money to the pot), **call** (match the money that other players put in the pot), or **fold** (discard their hand and lose the money they already put in the pot). The '2' in [2-4] Hold'em refers to the number of betting rounds, and the '4' refers to the number of raises allowed in each betting round. Thus, [2-4] Hold'em ends with only three public cards being part of the game, whereas full limit Texas Hold'em – which would be called [4-4] Hold'em using the same nomenclature – continues for all four rounds. After the final betting round, the agent who can make the best five-card poker hand out of their private cards and the public cards wins the pot.

2.2 Counterfactual Regret Minimization

In this work we utilize counterfactual regret minimization (CFR) (Zinkevich et al. 2008), an algorithm which creates strategies in extensive-form games, although any automatic strategy creation technique could be substituted in its place. CFR constructs strategies via an iterative self-play process. In two-player zero-sum games the strategy produced by CFR converges to a Nash equilibrium of the game (Zinkevich et al. 2008). While three-player games lack this guarantee, CFR-generated strategies have been shown to perform well in these games in practice (Risk and Szafron 2010).

Games like Texas Hold'em are far too large for CFR to maintain strategies for the full game, so *abstraction* techniques are typically employed. These techniques create a new game by merging information sets from the real game together. This means that a player is unable to distinguish some states that are distinguishable in the real game. CFR is run in the abstract game, and the strategy it learns can be played in the real game. Strategies created by CFR vary widely in their effectiveness, depending largely upon the quality and size of the abstraction used. Although pathologies exist (Waugh et al. 2009), in general as an abstraction is refined, the resulting strategy is stronger in the full game.

3 Detecting Collusion

Given a set D of game episodes, described above, our method assigns each pair of agents in M a score, derived from their behavior during the game episodes comprising D. A higher score indicates more suspicion of collusion by our method. The goal of this paper is to demonstrate that among the highest scores assigned by our method will be those of the actual colluding agents from D. We note that our approach will not decide with statistical significance the question of whether any two specific agents are in fact colluding. In this sense it can be likened to the task of *screening*, as that term is used in Harrington's survey (2008). For example, a system built using our approach could be used by human investigators to prioritize their investigations into the activities of suspected colluders, as opposed to simply relying on user complaints as is current practice.

3.1 Collusion Tables

The centerpiece of our approach for detecting collusion is the *collusion table*. Collusive strategies may take many different forms but one thing is common among all methods of

collusion: colluders play to benefit themselves, to the detriment of the other players. A collusion table captures the effect of each player's actions on the utility of all players. We first define a collusion table's semantics, describe how table entries are computed in Section 3.2, and show how collusion table entries can be used in Section 3.3.

	A	B	C	Chance	Sum
A	-3	$+13$	$+2$	-20	-9
B	$+8$	-6	$+2$	-25	-21
C	-5	-7	-3	$+45$	$+30$

Table 1: Example of a collusion table

Consider the example collusion table for three agents (A, B, and C), from a single game episode g, shown in Table 1. Each cell of a collusion table contains the *collusion value*, $C_g(j,k)$, for that cell's row agent (j) and column agent (k). The collusion value $C_g(j,k)$ captures the impact on agent j's utility caused by agent k's actions, where positive numbers denote positive impact, and negative numbers indicate negative impact, or harm. For example, the first column of Table 1 describes the effect that agent A had on all agents. We can see that A impacted his own utility by -3, B by $+8$, and C by -5. For any agent $a \in P_g$, $\sum_{b \in P_g} C_g(b,a) = 0$, i.e., the sum of each agent's impact on all agents must equal 0, as the game is zero-sum. Additionally, if we include chance as a column and record the impact of chance's "actions" on the agents, then in a symmetric game the sum over agent j's row will equal her utility in the episode, $u_{\phi_g(j)}(z_g)$.

3.2 Generating Collusion Values

While the cell values in a collusion table could conceivably be generated by many different methods, in this work we utilise the methodology which has been successful in the agent evaluation setting (Billings and Kan 2006; Zinkevich et al. 2006; White and Bowling 2009) and employ value functions. Assume we are given, for each position i in our game, a real-valued function $V_i : H \mapsto \mathbb{R}$, with the constraint that for all $z \in Z$, $V_i(z) = u_i(z)$. These functions are value functions, and $V_i(h)$ is an estimate of how much an agent in position i might expect to win at the end of a game that began at history h.

The impact that an agent has on another's utility can be computed given these value functions. Since agents can only affect the game through their actions, when describing an acting agent's impact on some target agent's utility, it suffices to consider the change in value that the target agent experienced as a result of the acting agent's actions. Precisely, for a given game episode g, and two agents $j, k \in P_g$, the value functions V are used to calculate the collusion value $C_g(j,k)$ describing the impact of agent k on agent j in game episode g, as follows:

$$C_g^V(j,k) = \sum_{\substack{ha \sqsubseteq z_g \\ p(h)=\phi_g(k)}} V_{\phi_g(j)}(ha) - V_{\phi_g(j)}(h)$$

Noting that $C_g^V(j,c)$ describes the impact of chance on the utility of agent j, and taking into consideration the starting value of each position $j \in N$ (i.e. $V_j(\emptyset)$) we can rewrite an agents utility as follows, effectively determining to what an agent's final utility is due.

$$u_{\phi_g(j)}(z_g) = V_{\phi_g(j)}(\emptyset) + \sum_{k \in P_g \cup \{c\}} C_g(j,k)$$

Given a set of game episodes $G = \{g_1, g_2, \ldots\}$ such that $P_{g_1} = P_{g_2} = \cdots = P_G$, we can naturally construct a collusion table which summarizes the collusion tables for all of the game episodes in the set. Each entry $C_G(j,k)$ in the summary collusion table is computed as the average of all $C_{g_i}(j,k)$ values from the individual game episodes as $C_G(j,k) = \frac{1}{|G|} \sum_{g_i \in G} C_{g_i}(j,k)$.

Creating value functions Several different methods have previously been used to create value functions in extensive-form games. Billings and Kan (2006) used a hand-created deterministic strategy to implicitly define the value of any history in the game. For a given history, the game is played from that state, with all players employing the same hand-created strategy to select actions during the remainder of the game. The value of a given state for a player is then the expected utility that player would receive if all players were to use the specified strategy for the remainder of the game. White and Bowling (2009) define features of a game history and learn a value function based on those features. In this paper, combining characteristics of each of these previous approaches, we utilise base agent strategies to implicitly define the two value functions we investigate. These base strategies, however, are not human-specified. They are instead automatically determined by applying CFR to two different abstractions of the game, described in Section 5.

3.3 Collusion Scores

Our main hypothesis is that collusion tables constructed in the described manner contain the information necessary to identify pairs of colluding agents. We now give two methods of deriving *collusion scores* from collusion tables and show in Section 5 that they can identify colluders. A collusion score is a number which designates, for a pair of agents in a collusion table, the degree of collusion exhibited by that pair in the collusion table. We do not claim that these two scores are the only ones that can be derived, but we use them to to confirm our hypothesis and show that it is possible to produce evidence of collusion from a collusion table.

Total Impact Score The *Total Impact* (TI) score is based on the idea that colluding agents are trying to maximise the total positive impact that they have on their joint utility. The total impact score for a game episode g is computed by simply summing the four collusion values from g's collusion table which show the impact of the pair on itself, as follows:

$$S_g^{TI}(a,b) = \sum_{i \in \{a,b\}} \sum_{j \in \{a,b\}} C_g(i,j)$$

Marginal Impact Score The *Marginal Impact* (MI) score is based on the idea that a colluding agent will act differently toward his colluding partner than towards other agents, and specifically calculates this difference as follows:

$$S_g^{MI}(a,b) = \left(C_g(b,a) - \frac{1}{|N|-2} \sum_{\substack{i \in P_g \\ i \notin \{a,b\}}} C_g(i,a) \right) + \left(C_g(a,b) - \frac{1}{|N|-2} \sum_{\substack{j \in P_g \\ j \notin \{a,b\}}} C_g(j,b) \right)$$

Each of the two terms in this formula computes for one of the pair the difference between their impact on their partner compared to the average impact on all other agents.

Agent i	Agent j	$S^{TI}(i,j)$	$S^{MI}(i,j)$
A	B	$+12$	$+33$
A	C	-10	-14
B	C	-14	-19

Table 2: Collusion scores for all pairs of agents in Table 1.

Table 2 shows both collusion scores for each pair of agents from the example collusion table shown in Table 1.

4 Experimental Dataset

To demonstrate the effectiveness of our approach, we needed a large dataset that included collusion. We are not aware of a suitable human dataset with known colluders, so we were compelled to construct and use a synthetic dataset . Our synthetic dataset was generated to be large enough so that obtained results are statistically significant, proving that our proposed approach can in principle be effective.

Creation of this dataset required first creating collusive strategies, a topic not previously discussed in the literature for general sequential games. We describe in Section 4.1 our novel approach for automatically constructing effective collusive strategies. The remainder of the section describes the agent population and creation of the dataset.

4.1 Colluding Strategies: Learning to Collude

To design collusive strategies for our agents, we return to the informal definition of collusion, given in Section 1, which is that colluders play so as to benefit each other, i.e., a colluder will play as if the colluding partner's utility is valuable as they may, in fact, be sharing their winnings afterward. We capture this idea by modifying the utility function of the game and creating a new game, for which CFR can determine effective positional strategies. For two colluders, seated in positions i and j in the game, the utility function for position i is modified for all $z \in Z$ to be $\hat{u}_i(z) = u_i(z) + \lambda u_j(z)$, where λ specifies exactly how much each colluder values their partner's utility. In what follows $\lambda = 0.9$ was used, which gives the colluders an average[1] advantage of 53.08 milli-big-blinds per game episode (mbb/g) against a non-colluding strategy, determined by CFR in the unmodified game. A *milli-big-blind* is one-thousandth of a big blind, the ante that the first player posts at the start of the game. This advantage is considered sufficient to ensure sustainable long-term profit by human poker players. The λ value to use was determined over a million-hand match of limit [2-4] Hold'em and $\lambda = 0.9$ yielded the maximum advantage over all tested λ values.

[1]Over all possible permutations of the agents

4.2 Base Positional Strategies

As we wanted to simulate imperfect agents of differing skill levels, we employed two different abstraction techniques (strong and weak) in developing strategies for three-player [2-4] Hold'em. Each abstraction employs different means of aggregating disparate information sets to control how much information each strategy can employ when making decisions. Strong agents used a large abstraction which employs no abstraction in the pre-flop round, and on the flop round divides the information sets into 3700 different groups, or buckets, using k-means clustering based on hand strength information. Weak agents used an abstract game with only five different buckets in the first betting round and 25 in the second. We employed CFR to create three different types of strategies for each position in both of these abstractions.

Collusive As described in Section 4.1, for each pair of positions we create strategies that collude, with $\lambda = 0.9$.

Defensive When CFR creates a pair of collusive strategies, the third position's strategy is created so as to minimise its losses when playing against two colluders.

Normal CFR creates this strategy in the unmodified game.

4.3 Agent Population

To create a realistic poker ecosystem, with various agents of differing ability, the previously described base positional strategies are combined in seven different ways. Agents differ in the base strategy they employ given the other agents participating in the game instance. Both weak (W.X) and strong (S.X) versions of each of the following kinds of agents were created using the abstractions described in Section 4.2.

Colluder A (CA) : If colluding partner, Colluder B, is present, utilise the appropriate collusive positional strategy. Otherwise, use a normal strategy.

Colluder B (CB) : Symmetric to Colluder A.

Non-colluder (NC) : Always employ a normal positional strategy.

Smart non-colluder / Defender (DF) : If playing with colluding pair, use defensive positional strategy, otherwise, use normal.

Paranoid (PR) : Always utilise defensive positional strategy.

Accidental collude-right (CR) : Always use positional strategy that colludes with the position to the right.

Accidental collude-left (CL) : Same as CR, but left.

These seven agent types represent diverse styles of play and create different challenges for collusion detection. The accidental colluders are designed to emulate poor players that may appear to be colluding, but any advantage gained is purely coincidental. This will allow us to see if our method can separate accidental and intentional collusion.

4.4 Creation of the Dataset

Each possible three-player configuration from the 14 agent population played in a one-million-hand three-player limit [2-4] Hold'em match. The full history of each hand, or game episode, was saved. The average utility won per-game-episode by each agent is reported in Table 3. This shows that, indeed, each strong agent gains more utility on average than any of the weak agents, and furthermore, the colluding agents are gaining an unfair advantage due to their collusion.

Agent	Utility (mbb/g)	Agent	Utility (mbb/g)
S.CA	36.63173	W.CA	-11.46571
S.CB	36.10692	W.CB	-11.89301
S.DF	32.44212	W.DF	-14.67359
S.NC	32.07846	W.NC	-14.80859
S.PR	20.13365	W.PR	-16.90147
S.CL	3.49873	W.CL	-28.67179
S.CR	-10.65571	W.CR	-51.82154

Table 3: Average per-game-episode utility gained.

It seems, intuitively, that it would be easy to identify the strong colluders due to their financial success. However, in general, the most successful poker agents will not be colluding, since they do not need to. Additionally, any focus primarily on money won will overlook the weak colluders. By comparison with the other weak agents it is clear that the weak colluders do gain an advantage, and a successful method should detect this, suggesting the need for a method that looks deeper into the game.

5 Experimental Results

We now examine the effectiveness of two different versions (A and B) of our collusion detection method on the synthetic dataset just described. So that results obtained are not specific to a particular value function, versions A and B differ in the value function used to create the collusion values. Neither version has any knowledge of the collusive strategies that compose the population.

Version A utilises a determinized, or purified (Ganzfried, Sandholm, and Waugh 2012) version of the S.NC strategy, described in Section 4.2, to implicitly define the value function, while version B uses a determinized version of a CFR strategy created using an abstraction of un-modified [2-4] Hold'em with 10 buckets in the pre-flop round and 100 buckets in the flop round. This abstraction is larger than the 5 bucket (weak) abstraction, but significantly smaller than the strong abstraction (see Section 4.2).

(a) Total Impact

Agent i	Agent j	$S^{TI}(i,j)$
S.CA	**S.CB**	**100.56**
W.CA	**W.CB**	**38.58**
S.CR	W.CR	37.52
S.CL	S.CR	34.93
W.CL	S.CR	9.17
S.CL	W.CR	7.98
S.CL	S.DF	4.32
S.CR	S.PR	4.18
S.CL	S.NC	3.94
S.CA	S.DF	3.77

(b) Marginal Impact

Agent i	Agent j	$S^{MI}(i,j)$
W.CR	S.CR	119.568
S.CA	**S.CB**	**109.20**
W.CA	**W.CB**	**45.43**
S.CL	S.CR	21.95
S.CL	W.CR	5.36
S.PR	W.PR	4.33
S.CL	S.DF	2.74
S.DF	W.PR	2.40
S.CL	S.NC	2.18
S.DF	W.NC	1.94

Table 4: Version A: top 10 scoring agent pairs (mbb/g)

The collusion table for each trio of agents is created from all game episodes involving those three agents, as described in Section 3.1. A collusion score for each pair of agents is then computed using each of the two scoring functions described in Section 3.3. The final collusion score for each pair of agents is computed by averaging the collusion scores for that pair across all collusion tables containing both agents.

5.1 Results

Tables 4 and 5 show the top ten scoring agent pairs from all 91 possible pairs, according to each scoring function, and using the A and B versions. The 95% confidence interval for all collusion scores is approximately ± 15 mbb/g, indicating that our dataset achieved its goal of being sufficiently large to ensure statistical significance. It also provides hope that colluding strategies could be distinguished with far fewer hands, which we explore in Section 5.3.

The total impact score with version A scores the two actual colluding pairs highest, while A's marginal impact score ranks them in spots 2 and 3. Version B's total impact score ranks the actual colluders in places 1 and 4 using the total impact score, and in places 2 and 3 using the marginal impact scores. In all cases the strong colluders received a higher score than the weak colluders.

To show the distribution of collusion scores across all 91

(a) Total Impact

Agent i	Agent j	$S^{TI}(i,j)$
S.CA	**S.CB**	**88.9617**
S.CL	S.CR	36.6128
W.CR	S.CR	34.8189
W.CA	**W.CB**	**30.6378**
S.DF	S.NC	13.3063
W.CL	S.CR	13.2091
S.DF	S.CB	13.101
S.DF	S.CA	12.9752
S.CA	S.NC	11.7711
S.NC	S.CB	11.317

(b) Marginal Impact

Agent i	Agent j	$S^{MI}(i,j)$
W.CR	S.CR	123.848
S.CA	**S.CB**	**110.728**
W.CA	**W.CB**	**41.6537**
S.CL	S.CR	35.3203
S.CL	W.CR	13.9641
W.CL	S.CL	4.95004
W.CL	S.CR	4.20684
S.PR	W.PR	3.32129
W.CL	S.DF	2.73517
W.CL	S.NC	1.93659

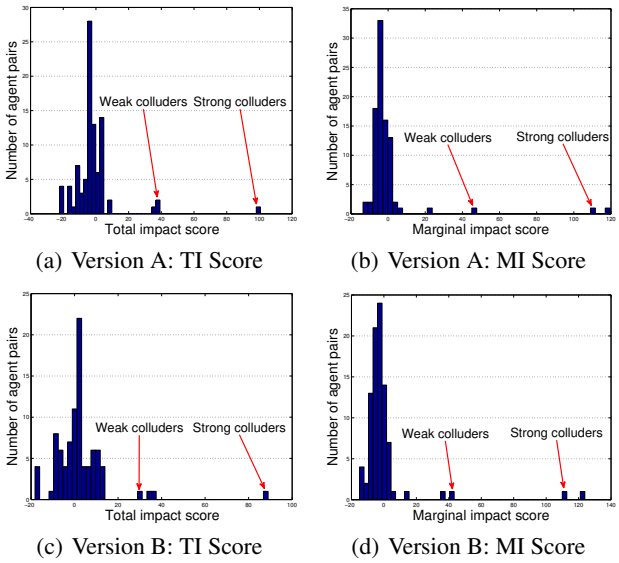
Table 5: Version B: top 10 scoring agent pairs (mbb/g)

agent pairs, the histograms for all scores are shown in Figure 1. The histograms give a sense of how distinguished the highly ranked pairs are from the majority of the pairs. For each scoring function only 4 pairings could be considered anomalous at all, and all of these pairs consist solely of colluding players, both intentional and accidental.

5.2 Discussion

These results demonstrate several points about the collusion detection method presented in this paper. This method ranks actual colluding pairs at or near the top of a list of all agent pairs from the population, and not only are the colluders at or near the top, but they are clear outliers, as shown in Figure 1. Our method detects the strong colluders, a minimal requirement given how much this pair stood out in the money results presented in Section 4.4. More importantly, our method is also able to detect the weak colluders who do not even make a profit over all of the games, placing them near the top of the list of suspicious agents.

The fact that both versions A and B are able to detect both colluding pairs suggests that the system is robust to the choice of value function. Specifically, it is successful when using a value function close in performance to the better members of the population (A) and is also successful using a value function implicitly defined by a strategy significantly



(a) Version A: TI Score (b) Version A: MI Score

(c) Version B: TI Score (d) Version B: MI Score

Figure 1: Collusion Score Histograms

different than any in the population (B).

In both versions A and B, the top four highest scoring agent pairs according to both scoring functions are either intentionally or accidentally colluding. While not intentionally colluding, the behavior of accidental colluders is also suspicious, and our method detects this. In this case, further human expert investigation might be required to fully determine that they are not in fact colluding.

5.3 Limited Data

Having shown that our proposed approach can successfully detect colluders given sufficient data, we briefly examine how the method might fare with a more limited amount of data. As mentioned in Section 5.1, the confidence intervals for the collusion scores that were generated indicate that the strong colluders should be highly ranked (top 4 with statistical significance) with as few as 90,000 hands per game episode. To confirm this, we ran Version B of the collusion detection system using only 100,000 hands from each configuration of three players. The top ten scoring agent pairs for each scoring function are presented in Table 6. Both the strong and weak colluders still stand out from the majority of the population with the smaller dataset.

While this may still seem like a large number of hands, a study of professional online poker players revealed that they often play over 400,000 hands annually (McCormack and Griffiths 2012). Indeed, to reliably profit from playing poker, either through skill or collusion, players must play enough hands to overcome the variance of the game. Not surprisingly, this should also provide our method enough data to also overcome the variance when detecting collusion.

6 Related Work

Previous research into collusion has typically focused on specific domains, with no general domain-independent treat-

(a) Total Impact

Agent i	Agent j	$S^{TI}(i,j)$
S.CA	**S.CB**	**87.67**
S.CL	S.CR	38.45
S.CR	W.CR	36.64
W.CA	**W.CB**	**31.41**
S.DF	S.CA	14.16
S.PR	S.CR	13.87
W.CL	S.CR	13.75
S.NC	S.CB	12.89
S.DF	S.CB	12.39
S.DF	S.NC	12.32

(b) Marginal Impact

Agent i	Agent j	$S^{MI}(i,j)$
S.CA	**S.CB**	**73.01**
W.CR	S.CR	39.20
W.CA	**W.CB**	**29.60**
S.CL	S.CR	20.85
W.CL	S.CR	4.27
S.DF	S.PR	0.07
W.DF	S.PR	-2.38
S.CL	S.DF	-2.49
W.CL	S.DF	-2.71
S.CL	S.NC	-3.36

Table 6: Version B: top 10 scoring agent pairs (mbb/g) using only 100,000 hands per game episode

ment. Collusion has long been an important topic in auctions (Robinson 1985; Graham and Marshall 1987; McAfee and McMillan 1992; Klemperer 2002), although such work is often focused on designing auctions with no (or reduced) incentive for participants to collude. Detecting colluders *post-hoc* in auctions has received less attention, but examples do exist. Hendricks and Porter (1989) and Porter and Zona (1993) identify collusive bids in auctions, and suggest that success requires domain-specific tailoring. A survey of collusion and cartel detection work is given by Harrington (2008). Kocsis and György (2010) detect cheaters in single agent game settings.

Due to the anonymity provided by the internet, recent work has started to consider collusion in online games. In 2010, collusion detection in online bridge was suggested as an important problem for AI researchers to focus on (Yan 2010). Smed et al. (2006; 2007) gave an extensive taxonomy of collusion methods in online games, but never attempted to detect collusive behavior. Lassonen et al. extended that work by determining which game features might be informative about the presence of collusion (2011). Yampolskiy (2008) gave a taxonomy of online poker cheating methods and provided a CAPTCHA-like mechanism for preventing bot-assistance in poker. Johansson et al. (2003) examined cheating in poker and developed strategies in a simplified version of 3-player Kuhn poker that share information and thus cheat.

Most of this work in online games actually deals with what we term cheating. Even so, the proposed approaches are based on the recognition of human-specified cheating patterns. To the best of our knowledge, none of the proposed approaches in these online game domains has been implemented and shown to be successful. Our work presents the first implemented and functional collusion detection technique which does not require human operators to specify collusive behavior.

The problem of detecting collusion from agent behavior is not that dissimilar from the problem of evaluating agent behavior, which has received a fair bit of recent attention in extensive-form games (notably poker). The principle of advantage-sum estimators (Zinkevich et al. 2006) has been used in several novel agent evaluation techniques, like DIVAT (Billings and Kan 2006) and MIVAT (White and Bowling 2009), which both aim to reduce the variance inherent in evaluating an agent's performance from sampled outcomes of play. Our method extends these approaches to do more than just evaluate an agent in isolation.

7 Conclusion

This paper presents the first implemented and successful collusion detection technique for sequential games. Using automatically learned value functions inter-player influence is captured in a novel object called a collusion table. We demonstrate the ability of our method to identify both strong and weak colluders in a synthetic dataset, whose creation required designing the first general method for creating collusive strategies. Because our collusion detection method does not rely on hand-crafted features or human-specified behavior detection we argue that it is equally applicable in any sequential game setting.

Acknowledgements

We would like to thank the members of the Computer Poker Research Group at the University of Alberta for their helpful suggestions throughout this project. This work was supported by Alberta Innovates through the Alberta Innovates Center for Machine Learning and also by the Alberta Gambling Research Institute.

References

Billings, D., and Kan, M. 2006. A tool for the direct assessment of poker decisions. *International Computer Games Association Journal*.

D'Aloisio, T. 2010. Insider trading and market manipulation. Retrieved May 30, 2012 from: http://www.asic.gov.au/asic/pdflib.nsf/LookupByFileName/speech-insider-trading-market-manipulation-August-2010.pdf/$file/speech-insider-trading-market-manipulation-August-2010.pdf.

Ganzfried, S.; Sandholm, T.; and Waugh, K. 2012. Strategy purification and thresholding: Effective non-equilibrium approaches for playing large games. In *Proceedings of the 11th*

International Conference on Autonomous Agents and Multiagent Systems, AAMAS '12.

Graham, D. A., and Marshall, R. C. 1987. Collusive bidder behavior at single-object second-price and english auctions. *Journal of Political Economy* 95(6):1217–39.

Hendricks, K., and Porter, R. H. 1989. Collusion in auctions. *Annales d'Economie et de Statistique* 217–230.

Johanson, M.; Bard, N.; Lanctot, M.; Gibson, R.; and Bowling, M. 2012. Efficient nash equilibrium approximation through monte carlo counterfactual regret minimization. In *Proceedings of the Eleventh International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Johansson, U.; Sönströd, C.; and König, R. 2003. Cheating by sharing information–the doom of online poker. *Proceedings of the 2nd International Conference on Application and Development of Computer Games* 16–22.

Joseph E. Harrington, J. 2008. Detecting cartels. *Handbook in Antitrust Economics*.

Klemperer, P. 2002. What really matters in auction design. *Journal of Economic Perspectives* 16(1):169–189.

Kocsis, L., and György, A. 2010. Fraud detection by generating positive samples for classification from unlabeled data. In *ICML 2010 Workshop on Machine Learning and Games*.

Laasonen, J.; Knuutila, T.; and Smed, J. 2011. Eliciting collusion features. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, SIMUTools '11, 296–303. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

McAfee, R. P., and McMillan, J. 1992. Bidding rings. *American Economic Review* 82(3):579–99.

McCormack, A., and Griffiths, M. 2012. What differentiates professional poker players from recreational poker players? a qualitative interview study. *International Journal of Mental Health and Addiction* 10(2):243–257.

Osborne, M. J., and Rubinstein, A. 1994. *A course in game theory*. MIT Press.

PokerStars. 2012. Data privacy for our poker software: Collusion. Retrieved May 30, 2012 from: http://www.pokerstars.com/poker/room/features/security/.

Porter, R. H., and Zona, J. D. 1993. Detection of bid rigging in procurement auctions. *Journal of Political Economy* 101(3):pp. 518–538.

Risk, N. A., and Szafron, D. 2010. Using counterfactual regret minimization to create competitive multiplayer poker agents. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*, AAMAS '10, 159–166. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Robinson, M. S. 1985. Collusion and the choice of auction. *The RAND Journal of Economics* 16(1):pp. 141–145.

Smed, J.; Knuutila, T.; and Hakonen, H. 2006. Can we prevent collusion in multiplayer online games? In *Proceedings of the Ninth Scandinavian Conference on Artificial Intelligence 2006*, 168–175.

Smed, J.; Knuutila, T.; and Hakonen, H. 2007. Towards swift and accurate collusion detection. *Security*.

Watson, M. J. 2012. The regulation of capital markets: Market manipulation and insider trading. Retrieved May 30, 2012 from: www.icclr.law.ubc.ca/Publications/Reports/wats_pap.pdf.

Waugh, K.; Schnizlein, D.; Bowling, M.; and Szafron, D. 2009. Abstraction pathologies in extensive games. In *Proceedings of the Eighth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 781–788.

White, M., and Bowling, M. 2009. Learning a value analysis tool for agent evaluation. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, 1976–1981.

Yampolskiy, R. V. 2008. Detecting and controlling cheating in online poker. In *2008 5th IEEE Consumer Communications and Networking Conference*, 848–853. Ieee.

Yan, J. 2010. Collusion detection in online bridge. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*.

Zinkevich, M.; Bowling, M.; Bard, N.; Kan, M.; and Billings, D. 2006. Optimal unbiased estimators for evaluating agent performance. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI)*, 573–578.

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2008. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems 20 (NIPS)*, 905–912. A longer version is available as a University of Alberta Technical Report, TR07-14.