University of Alberta

COLLUSION DETECTION IN SEQUENTIAL GAMES

by

Parisa Mazrooei

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Fall 2012
Edmonton, Alberta

To my dear family,
for being there.

Abstract

Collusion is the deliberate cooperation of two or more parties to the detriment of others. While this behaviour can be highly profitable for colluders (for example, in auctions and online games), it is considered illegal and unfair in many sequential decision-making domains and presents many challenging problems in these systems.

In this thesis we present an automatic collusion detection method for extensive form games. This method uses a novel object, called a collusion table, that aims to capture the effects of collusive behaviour on the utility of players without committing to any particular pattern of behaviour. We also introduce a general method for developing collusive agents which was necessary to create a dataset of labelled colluding and normal agents. The effectiveness of our collusion detection method is demonstrated experimentally. Our results show that this method provides promising accuracy, detecting collusion by both strong and weak agents.

Acknowledgements

I spent two great years of my life studying at University of Alberta. During these two years I had been supported by many exceptional people that I would like to thank.

First and for most, I would like to thank my supervisor, Michael Bowling, for being such a wonderful supervisor. I appreciate his support, guidance and encouragement during all the time I have worked with him. More than supervising me in my research, his good personality is inspirational. I have learned from him how to be more positive, patient and a better human being. I also appreciate the wonderful moments that he and his family shared with me.

I would also like to thank Christopher Archibald, who co-supervised me in this work. He has provided a friendly atmosphere for research and has always been there to support and guide me whenever I was lost. He has thought me how to stay strong when a storm is passing.

Next, I would like to thank the members of the University of Alberta Poker Research Group, particularly Michael Johanson, Richard Gibson and Neil Burch for their help with the ideas and implementation issues.

It is said that friends are those who nourish your spirit. I am grateful to experience this with friends that I made during this journey in Edmonton and those who I left in Iran. In particular, I like to thank Zohreh, Omid and Kaveh for their support during these two years. I will always be indebted to them. Many thanks goes to Shadi, Iraj, the 7s, Afarin and Akram for their constant remote supports. I made wonderful friends in Edmonton that I could never imagine when coming here. Among those are, Carol, Bernardo, Hossein, Saeed and Afra. You have all touched my heart. Thank you all for our discussions and all the fun.

I am blessed to have a great family. They give me both the freedom to fly and the support to lean against. I humbly thank my parents, Shahnaz and Habib and my brother, Amir who always support me and encourage me. I thank my dear aunt, Maryam and my entire family for always being there. Last but not least, I would like to thank my fiance, Shervin. He is the family when it comes to supporting me and the friend when I need someone to talk. Thank you.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Collusion is the practice of two or more parties deliberately cooperating to the detriment of other parties. While this cooperation is allowed and encouraged in some multi-agent settings, in other domains it is illegal or prohibited and can provide colluders an unfair advantage. Because of this, detecting and preventing collusion is a challenge of major interest in many different settings.

One real world system in which collusion presents a challenging problem is financial markets. Collusive behaviour in these domains includes insider trading and market manipulation to set prices or limit production. Previous research has introduced some collusion detection methods that utilize pre-established parameters in computerized trading systems as an indicator of normal versus suspicious activities. If these parameters' normal values are exceeded in any trade, the system identifies and reports the trade [21]. This mechanism is based on parameters which are domain specific and thus cannot be generalized to other settings where collusion detection is desired. Additionally, the system does not completely determine that illegal activity has occurred, but further consideration of trained staff is needed to fully detect that unlawful interaction occurred.

Another setting in which collusive behaviour is illegal is games such as poker. On online poker websites participants are instructed to report any suspicious behaviour to the website administrators. Additionally, some detection methods are used which warns security personnel if any pre-specified unusual play patterns occur [15]. These methods are also based on known patterns of collusion and require human experts to examine the history of the games in which suspected colluders participated. When the number of players increases or the volume of games is large, in-depth examination of data for collusion patterns costs an enormous amount of resources, both in terms of human experts' time and money.

Some detection methods build a model of normal or collusive behaviour based on previous interactions of agents [25, 26]. However, such models can only detect collusive behaviour that matches the known collusive model or differs from the normal model and cannot identify other collusive behaviours.

While no collusion detection system can completely replace human experts, automated detection methods can assist them and improve the efficiency of investigations. Ideally, an automated collusion detection system should be able to detect collusive behaviour based not only on a specific pattern but also be able to detect unknown patterns that are also collusive. The method should depend on the actions taken by agents and not on any domain specific factors. The goal of this thesis is to design an automated general-purpose collusion detection method that can be used in different multi-agent systems.

Note that we distinguish between the concept of *cheating* and colluding. We define *cheating* as using information or actions that is not permitted by the structure and rules of the game. For example, when players share their private cards using a back channel, it is cheating. However, collusion is a series of valid actions which leads to an unfair advantage for colluders. For example, when a player reraises his partner to increase the leverage of his bet, it is colluding. The focus of this work is collusion detection, although we hypothesize that our technique could detect cheating as well.

Our approach to developing a general-purpose collusion detection method focuses fundamentally on our definition of collusion: that colluders act to increase their joint utility. First, we define the collusion detection problem in the domain of extensive form games, a general model of sequential multi-agent decision making systems. We then focus on how the actions taken by players affect the utility of all participants. This information is summarized in a novel data structure called a *collusion table*. We then evaluate our method by a series of experiments in the domain of poker, where collusive behaviour can give colluders a great advantage and collusion detection is a real world challenge. The experiments validate our approach for detecting collusion in the domain of poker. This method can also be applied to other domains since it does not use any domain specific knowledge.

This thesis will proceed as follows. First, in Chapter 2, we describe work related to collusion detection, which can be divided into two categories: the first is previous work based on known patterns of collusion or domain specific methods. The second is work in the domain of operations research and game theory for evaluating agent behaviour. In 2006, Zinkevich and colleagues proposed *advantage sum* estimators for constructing low-variance unbiased estimates of an agent's performance [27]. This approach has been utilized in agent evaluation techniques like DIVAT and MIVAT which will be explained in Chapter 2. Unlike other methods of estimating agent's performance, this method examines the whole history of players' interactions. We also employ the same underlying idea as a basis for our method of evaluating the effect of agents' actions on each other in games.

Chapter 4 is devoted to the construction of collusion tables. First, the design and semantics of collusion tables are described, followed by some examples of collusion tables. We also introduce a number of different scoring methods which are designed to evaluate the likeliness that a pair of agents are colluding, given a collusion table. While collusion tables could be constructed in many different ways, in this work we use value functions from histories of the game to real numbers. This approach has been shown to be successful in the field of agent evaluation [2, 23, 27]. We propose three techniques for constructing a value function in Chapter 5.

Ideally, a dataset of play by both colluding and non-colluding agents, where each agent is labelled as such, would be used to validate our approach. It would be most desirable for this dataset to be constructed from human play. However, as such a dataset is not available, we instead created a synthetic dataset of matches using various "bots". This requires developing poker bots that collude with partners, which we did by modifying the utility function of the colluders so that they consider their partner's utility as their own. This technique is described in Chapter 3 and is accompanied by the results of experiments that verify the effectiveness of our method at creating profitable colluding bots. The different kinds of agents which we developed to represent players with different skill levels and a description of our experimental design is described in Chapter 6. The results of our experiments are presented in Chapter 7. Finally, we conclude this work in Chapter 8 and discuss possible future work.

Chapter 2

Background and Related Work

In this chapter, we define the general domain in which collusion detection is investigated. Then, the features of poker which make it a proper experimental domain are explained. After presenting the problem formulation, previous work on collusion detection and other related work in the field of game theory is described. Finally, a state of the art technique for developing poker agents is explained which will be used in our experiments.

2.1 Definitions and Notation

In this section the preliminary definitions used in this thesis are presented.

2.1.1 Extensive form games

Definition 1. *(Adapted from Osborne and Rubinstein [14]) A zero-sum **Extensive form game** is defined to have the following components:*

- *A finite set N (the set of **players**).*
- *A finite set H of sequences such that : (1) the empty sequence $\emptyset$ is a member of H. (2) If h' is a prefix of h and $h \in H$, then h' is also a member of H. (3) If every subsequence h' of an infinite sequence h is a member of H, then $h \in H$.*
  *Each member of H is called a **history**. $Z \subseteq H$ is the set of **terminal histories** which are sequences that are infinite or are not a subsequence of any other sequence in H.*

- *For each player $i \in N$ a function $u_i : Z \mapsto \mathbb{R}$ which assigns a **utility** for player i to each terminal history $z \in Z$.*

- *A finite set A consisting of all possible **actions** a player can take. $A(h) = \{a \in A : ha \in H\}$ denotes the set of actions*[1] *that a player can take after a history $h \in H$.*

- *A **player function** $P : H \setminus Z \mapsto N \cup \{c\}$ which assigns a player (from set $\{1, 2, \ldots, N\}$ or the chance player c) to each non-terminal history h. $P(h)$ is the player who takes an action after history h. If $P(h) = c$ then chance determines the action taken after the non-terminal history h.*

- *A function σ_c which associates with every history in $\{h \in H : P(h) = c\}$ a **probability distribution** $\sigma_c(.|h)$ on $A(h)$. $\sigma_c(a|h)$ is the probability that action a occurs after the history h.*

- *For each player $i \in N$ a partition $\mathcal{I}_i$ of $\{h \in H : P(h) = i\}$ such that $A(h) = A(h')$ whenever h and h' are in the same member of the partition. $\mathcal{I}_i$ is the **information partition** of player i; a set $I_i \in \mathcal{I}_i$ is an **information set** of player i.*

A game is called **a zero-sum game** if we have $\sum_{i \in N} u_i = 0$ for all terminal histories in Z. Extensive form games can be represented as a game tree. Each node represents a player (or chance player) and each edge represents a valid action. The path from root to each node determines a history of the game. All possible sequences of actions are represented in a game tree (terminal histories with their corresponding utilities). In extensive form games with perfect information (like chess), a player can determine the state of the game and has complete information about all players' actions taken previously (by looking at the board). However, in extensive form games with imperfect information, a player may not have information about previous actions that are taken by players or chance. This means that players may not be able to differentiate between game states. Consider the example shown in Figure

[1] ha refers to a history consisting of action a concatenated to history h.

2.1. In this game tree, player 1 is the first player to act and can take action A or action B. Then player 2 is the player to act who can choose action C or action D. There are five terminal histories in this game which are marked as black. In this figure the dashed line indicates that histories AC and AD are in the same information set of player 1 which means this player cannot distinguish between history AC and history AD. In either case, it is player 1's turn to play and choose an action from set $\{E, F\}$.

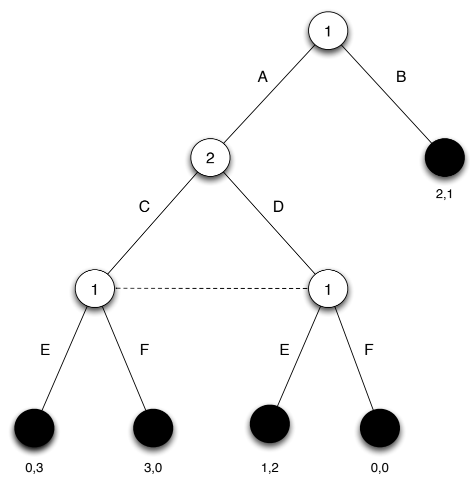


Figure 2.1: An example of an extensive game with imperfect information.

2.1.2 Poker and its properties

In this thesis, poker is used as an experimental domain to validate our collusion detection methods. Some of the properties that make poker a suitable domain for this research are as follows:

- Poker is a multiplayer game with imperfect information. Important information which a player cannot access is the private cards of opponents. Therefore, a player cannot make decisions based on this information unless he illegally accesses it by cheating or colluding.

- Poker is a popular game; not only because it is entertaining, but also because money is involved in poker. Many poker experts view the game as a way to gain money. This means that players may be more motivated to collude or cheat in poker than other games.

- Poker players have many different skill levels. Also, there is no single, known winning strategy in this game. This results in a wide variety of poker playing styles which complicate the task of detecting unusual behaviours which might be signs of collusion or cheating.

- Finally, poker provides us a good representation of real world environments. Settings like auctions and market places are very similar to poker in their structure. Therefore, techniques that we develop for poker should be extendable to these real-world settings as well.

2.1.3 Strategy and Nash Equilibrium

The outcome of an extensive form game is highly dependent on the strategies of the players. Informally, the actions that a player chooses at different states of a game is called that player's strategy. Each strategy is simply a set of action distributions for each information set in the game. Since a player's strategy affects the outcome of the game, a player would like to choose a strategy which increases the player's utility at reached outcomes.

Definition 2. *(Adapted from Osborne and Rubinstein [14]) A **behavioural strategy of player** i is a collection $(\beta_i(I_i))_{I_i \in \mathcal{I}_i}$ of independent probability measures, where $\beta_i(I_i)$ is a probability measure over $A(I_i)$.*

β indicates a profile of behavioural strategies such that for each player i, $\beta_i \in \beta$ is the behavioural strategy of player i. β_{-i} indicates a strategy profile for all the players in a game except for player i. The expected utility of player i if all the players in the game follow the strategy profile β is $u_i(\beta)$.

Definition 3. *(Adapted from Osborne and Rubinstein [14]) A **Nash equilibrium** in behavioural strategies of an extensive game is a strategy profile β^* of behavioural strategies with the property that for every player $i \in N$ we have*

$u_i(\beta^*_{-i}, \beta^*_i) \succsim_i u_i(\beta^*_{-i}, \beta_i)$ *for every behavioural strategy* β_i *of player* i.

In this thesis, we refer to a behavioural strategy as a **strategy**. An entity with a strategy to play as a player or position in the game is called **positional agent** or **player**. A set of positional agents which has at least one strategy for each position in the game forms an **agent**. Thus, an agent can participate as any player in a game.

2.1.4 The Collusion detection problem

In this section we introduce the problem of collusion detection. Assume that we have a dataset of past interactions between agents from some population of agents M.

Definition 4. *A* ***game episode*** g *is a tuple* $\langle P_g, \phi_g, z_g \rangle$*, in which* $P_g \subseteq M$ *is the subset of agents participating in* g*. A mapping function* $\phi_g : P_g \mapsto N$ *that associates each agent with a player or position in the game and* z_g *is the terminal history reached at the end of game episode* g*.*

Definition 5. *Given a set of agents M participating in a dataset of game episodes D, the* ***Collusion Detection Problem*** *is to find a function* $\vartheta : M^2 \mapsto \mathbb{R}$ *that measures the possibility of collusion for each pair of agents in* M *and ranks them accordingly.*

That means, if $\vartheta(a, b) > \vartheta(c, d)$, then it is more likely that agents (a, b) are colluding than agents (c, d) are. In other words, the goal of collusion detection is to rank the pair of players participating in a dataset in order of their likeliness of collusion.

2.2 Experimental Domain

As described in section 2.1.2, poker has properties which make it a suitable experimental domain for the purpose of this research. In this section, we explain the variation of poker that is used in our experiments.

2.2.1 [2-4] Limit Hold'em Poker

[2-4] Hold'em Poker is a small version of Texas Hold'em Poker. In Texas Hold'em, we have 2 to 10 players. Each player is given a hand of 2 private cards from a

shuffled deck of 52 cards. There are four rounds in the game, **pre-flop**, **flop**, **turn** and **river**. A specific number of public cards are revealed after each round (three after the pre-flop, one after the flop and one after the turn). After each deal, players have the options to **bet** (add more money to the **pot**), **check/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 objective is to win the money in the pot. After the last round, player who has not folded that can make the best 5-card poker hand out of his private cards and the public cards wins the pot.

Texas Hold'em is usually played using **small** and **big blind bets**. These are the first two bets which the first two players must begin with. A dealer button is used to specify the position of the **dealer**. The player to the left of dealer bets the small blind and the big blind is bet by the player to the left of the small blind. The dealer button rotates clockwise after each game episode to change the position of the dealer and blinds. In this way, the effect of position on the utility of the players is averaged out over the hands.

There are two variants of Texas Hold'em Poker and [2-4] Hold'em Poker called **limit** and **no-limit**. In the limit version, four raises are allowed in each round of the game. Additionally, the size of the bets is fixed. On the other hand, in each round of no-limit games, players can raise any number of times by any arbitrary amount which is greater than or equal to the minimum bet size and less than their total number of chips. Therefore, the size of the game tree in no-limit games is much larger than in limit games.

[2-4] Hold'em Poker is exactly like limit Texas Hold'em except for the number of rounds. The game ends after the flop betting round, and there are no turn or river rounds. Therefore, the number of public cards in total is three (The name, [2-4], refers to two rounds and four raises that are allowed in each round). The game that is used in our experiments is [2-4] Limit Hold'em poker because of its similarity to Texas Hold'em and its smaller size. The size of the small and big blinds are set to 5 and 10, respectively.

2.3 Previous Work

There have been a few previous studies on detecting and preventing collusion in different domains. Many suggested methods are based on patterns of collusion which are specified by human experts. The others design methods based on features of the game which are mostly domain specific. In this section an overview of this work in the domain of auctions and games is presented .

Auctions are one of the important areas in which collusion is prohibited and thus has received great attention by scientists. Most of this work, however, discusses the problem of collusion prevention and is aimed at designing auctions so that collusion is not a profitable strategy for bidders [7, 10, 13]. In 1989, Hendricks and Porter [8] argued that the presence of collusive behaviour is highly dependent on the object being auctioned and the auction rules.

Robinson [17] compared the stability of cartels in oral auctions versus sealed high-bid auctions. The results shows that cartels are stable in *oral* auctions but not in *sealed high-bid* auctions, but this doesn't mean that collusion can never happen in the latter. Bachrach et al. [1] investigated collusive behaviour of bidders in a class of auctions called *Vickrey-Clarke-Groves* auctions using a cooperative game theory approach. They examined different auctions with respect to the possibility of a *stable* agreement between colluders. They showed that in some auctions participants may not be able to form a long-lasting agreement even if they have complete information about each other's preferences.

Another area in which collusion is a real-world challenge and has been studied is online games. In 2010, Yan introduced the problem of detecting collusion in online bridge as a hard problem due to anonymity and the benefit of sharing information through back channels that players have when playing over internet [26]. The motivation was to solve the general problem of detecting the usage of prohibited information in decision making, which is categorized in our termination as cheating not colluding. The approach suggested by Yan is to focus on critical parts of the play in which traces of cheating may appear more and compare the sequence of decisions of the players with a model of non-colluding play. However, their approach is completely based on the "critical parts" they introduced in bridge and is not applicable to other domains.

Smed et al. [19,20] also investigated the problem of collusion in different games and gave an extensive classification of collusion based on types of agreement that colluders can have. They also introduced a simplified version of the game of Pac-Man for evaluating collusion detection methods. Laasonen et al. [11] utilized this game for their experiments to detect features that are indications of collusion. The features they suggested are secondary factors which are domain specific. Finally, they gave an analysis of the utility functions of different groups of colluders in the designed experimental settings.

2.4 Related Work on Agent Performance Evaluation

Performance evaluation is a well-studied problem which has some similarities to the problem of detecting collusion. This problem arises in any multi-agent system in which designing better agents is needed. Because of the stochastic nature of the environment and agent's decisions, designing a low variance estimator of agent performance has been the target of several recent studies. These issues also arise in detecting collusion and makes the problem of designing a collusion detector method a challenge. In 2006, Zinkevich et al. [27] introduced a low variance estimator called an *advantage sum*. The principle of this method has been used in two other agent evaluation techniques on which the idea of our collusion detection method is built. In this section we briefly explain these methods (for a complete description of these methods, see the original papers [2,23,27]).

2.4.1 Advantage Sum Estimators and DIVAT

In 2006, Zinkevich et al. [27] showed that given any value function on the histories of a finite extensive form game, an unbiased estimator can be generated. They do this as follows: Assume one is given a real valued function on histories $V_i : H \mapsto \mathbb{R}$ such that $V_i(z) = u_i(z) \ \forall z \in Z$ for any given player $i \in N$. Define the following real-valued functions on terminal histories:

$$S_{V_i}(z) = \sum_{\substack{ha \sqsubseteq z \\ P(h) \neq c}} V_i(ha) - V_i(h) \tag{2.1}$$

$$L_{V_i}(z) = \sum_{\substack{ha \sqsubseteq z \\ P(h) = c}} V_i(ha) - V_i(h) \tag{2.2}$$

$$\text{Pos}_{V_i} = V_i(\emptyset) \tag{2.3}$$

We write $ha \sqsubseteq z$ to denote that history ha is a prefix of terminal history z. We call these values **skill**, **luck** and **position** of player i (note that in this definition skill is computed with respect to the players participating in the game and will change if the players change). The utility can be rewritten as a function of skill, luck and position because terms cancel when summing skill, luck and position.

$$\begin{aligned} S_{V_i}(z) + L_{V_i}(z) + \text{Pos}_{V_i} = \\ &V_i(\emptyset) \\ &+ V_i(a_1) - V_i(\emptyset) \\ &+ V_i(a_1 a_2) - V_i(a_1) \\ &+ \cdots + V_i(z) - V_i(a_1 \cdots a_{|z|-1}) \\ &= V_i(z) \\ &= u_i(z) \end{aligned} \tag{2.4}$$

The advantage sum estimator is then defined as follows:

$$\hat{u}_{V_i}(z) = S_{V_i}(z) + \text{Pos}_{V_i} \tag{2.5}$$

To have an unbiased estimator, Zinkevich et al. [27] suggest choosing the value function for player i so that the expected value of luck for player i equals zero. This is called the *zero-luck constraint*. In this case, we have:

$$
\begin{aligned}
E[\hat{u}_{V_i}(z)|\sigma] &= E[S_{V_i}(z) + \mathrm{Pos}_{V_i}|\sigma] \\
&= E[S_{V_i}(z) + L_{V_i}(z) + \mathrm{Pos}_{V_i}|\sigma] \qquad \text{if } \; E[L_{V_i}(z)|\sigma] = 0 \\
&= E[u_{V_i}(z)|\sigma]
\end{aligned}
$$

Specifically, they suggested that if the value function is chosen such that the value of histories before a chance node is equal to the expected value of the histories right after a chance node for player i, then the zero-luck constraint is satisfied. This approach will be used later in the computation of *collusion values* in Chapter 4.

Based on this technique, Billings and Kan proposed the *Ignorant Value Analysis Tool* (DIVAT), a low variance estimator of agents in two player Texas Hold'em poker which uses a hand designed value function which satisfies the zero-luck constraint [2].

2.4.2 MIVAT

Although DIVAT has been shown to have very good performance in two player Texas Hold'em limit poker, it cannot be expanded to other domains since it uses a hand crafted value function. White and Bowling proposed a more general approach to utilize the advantage sum technique called the *Informed Value Assessment Tool*. Instead of using a domain specific hand designed value function, they learn a value function based on the features of the domain, given sample data of previous matches played by players, and so is informed by data of past interactions [23].

To satisfy the zero-luck constraint, they assume that the value function is only learned for histories following a chance node. Then, the value of the game for player i at histories where chance is next to act is defined to be computed from a weighted sum of the other histories. The value function is then:

$$
V_i = \begin{cases} \displaystyle\sum_{a \in A(h)} \sigma_c(a|h) V_i(ha) & \text{if } P(h) = c, \\ \text{learned function} & \text{Otherwise.} \end{cases}
$$

Given T samples of outcomes, they minimized the sum of the estimated utility variance over T samples. To make this optimization tractable, they focused on the class of linear value functions and proposed a closed form solution for such value functions. Our method extends these approaches to the problem of collusion detection.

2.5 Counterfactual Regret Minimization

Regret minimization is a well-known concept in online learning [3]. In 2008, Zinkevich et al. [28] proposed a technique for solving extensive form games based on regret minimization. They introduced a new concept called *counterfactual regret* and proved that minimizing counterfactual regret leads to minimizing overall regret which results in an approximate Nash equilibrium strategy in two-player zero-sum games. This approach also requires only memory linear in the number of information sets instead of game states. As the agents used in our experiments are built using this technique, we briefly describe it in this section.[2]

The concept of *regret* is similar to *opportunity cost* in economics. We can informally define it as follows: suppose one has taken action a and gained utility $u(a)$. Regret is the different between the utility that you gained and the maximum utility that could have been possible for you to gain if you had taken the right action a^* or $u(a^*) - u(a)$. A strategy selection algorithm is called *regret minimizing* if the average overall regret of player i playing the chosen strategy σ_i^t in iteration t goes to zero as the number of repeated choices played goes to infinity.

Counterfactual Regret Minimization is a regret minimizing algorithm for solving zero-sum extensive form games. In this algorithm, positional strategies play repeated games against each other. The algorithm's action probability distribution is initiated uniformly from all possible choices in all information sets. In each round and for each information set, a positional agent improves its strategy (by changing the probability distribution over actions) so as to minimize the regret of its subtree given that strategies of other participants are fixed. In a two player zero-sum game, it is shown that in self-play as the number of games increases, the regret

[2] Agents used in the experiments of this thesis are developed by Computer Poker Research Group in University of Alberta (CPRG). The author's contribution is to modify them to collude and develop different kinds of players and agents.

minimizing behaviour of positional strategies will cause them to approach a Nash equilibrium [9] . CFR-generated strategies have been shown to have good performance in multiplayer poker games (e.g. it achieved first place in the three-player limit Texas Hold'em category at the Annual Computer Poker Competition in 2012 [1] [6]). We used CFR in our experiments due to having the advantage of its low memory requirement and possibility of parallel computation [9] which improves the speed of the program.

[1] http://www.computerpokercompetition.org/

Chapter 3

Colluding Bots

In order to evaluate our proposed collusion detection methods, a synthetic dataset of labeled colluding and non-colluding agents is required. Since a labeled human dataset of poker games is not available, we used a population of "labeled" bots to play a set of game episodes. While there have been a number of non-colluding bots developed by the CPRG and other research groups [4], to our knowledge no colluding agent has ever been developed. Thus, to create a synthetic dataset, we first develop colluding bots.

There are many opportunities in most multi-agent settings for players to collude. Some of these collusion methods depend on the system's characteristics and cannot be utilized in other domains. Here, we propose a collusion method which is not dependent on any system properties or constrained to any particular pattern of collusion. Our method centers on the very definition of collusion: that colluding is jointly beneficial for all colluding partners.

After describing our method we evaluate it in the domain of [2-4] Hold'em poker and the results show that agents are indeed successfully colluding. However, this method could be used in any multi-agent system, including any other variations of poker. In this chapter our technique for developing colluding bots is introduced and the experimental results which validates the method are presented.

3.1 Learning to Collude: Modified Utility Function

In a multi-agent system players can collude using various methods. Many of these techniques are domain dependent. That is, colluders take advantage of characteristics of the game to increase their utilities. Other methods utilize the environment to collude. For instance, using an independent communication channel to share private information is one method of colluding in online poker. However, all of these methods have one thing in common: the purpose is to increase the joint utility of the colluders. One way this can be interpreted is that each party involved in collusion partially considers his partner's utility as his own and plays accordingly. For instance, consider two colluding players in poker, one with a weak and one with a strong hand. One example of collusive behaviour would be to gain the pot by raising and re-raising until all non-colluding players with marginal hands fold. After that, the colluder with the weak hand folds to a raise and the colluder with the strong hand gets the pot. This method is known as *active collusion* in which the weak colluder considers the utility of the strong colluder as his own completely [24]. In contrast, if he were to play normally, the best strategy would be to fold at the beginning of the game to avoid a huge loss.

We use a utility function to capture the notion of colluding agents (partially) considering their partner's utility as their own. This is done by modifying the utility function. Once the utility function is altered, we can use any strategy creation method to build the strategies for each position in the modified game. We define the modified utility function for colluding player i who colludes with player j as:

$$\hat{u}_i(z) = u_i(z) + \lambda u_j(z) \qquad \text{for all } z \in Z,$$

where parameter λ specifies how much a player considers his partner's utility as his own. We then apply the Counterfactual Regret Minimization (CFR) method [16,28] to the modified game to create positional collusive strategies. CFR is used since it has been shown to generate winning computer poker agents for multiplayer games [16]. For each combination of positions in the game, a pair of colluding positional strategies were created. Due to the huge number of game states in poker, we first abstract the game and then develop colluding strategies in the abstracted game. In general the less abstraction is used, the higher the quality of the created strategy will be in the full game (although some examples of abstraction pathologies in toy poker games exists [22]).

3.2 Results and Discussion

To determine which value of λ was most beneficial for the colluders, we ran several sets of experiments. First, positional strategies for different λ values were created for the game of [2-4] Hold'em poker. To ensue the consistency of the developed positional strategies, different random seeds were used when building the strategies using CFR. Once the strategies were created, a set of experiments was ran to verify that players can collude. Each experiment consists of three players, two colluders and one normal player who played one million hands (game episodes) of [2-4] Hold'em poker. The dealer button did not rotate in these game sets since the players are position dependent. Table 3.1 shows the results of this set of games: sixteen matches, each of one million hands. For each value of $\lambda > 0$, a one million hand match was played for each of the three different combinations of colluders' and non-colluder's positions. The performance of players is measured in milli-big-blind/game (mbb/g), where milli-big-blind is 0.001 big blind. The name of the players in this table shows the player index followed by its partner in that match. When a player does not have a partner, N is used to show that it is a normal non-colluding player. For instance, player 1,2 on the first row of the table indicates that player 1 is colluding with player 2 in this match.

We also ran a one-million-hand match in which nobody is colluding (that is when $\lambda = 0$) for the sake of comparison. In this set rotating positional strategies does not provide further information. The results for this match is presented in the last three rows of Table 3.1.

Table 3.1 demonstrates the average winnings of each player during a one-million-hand set. As described in Section 2.4, the utility of a player can be divided into the utility achieved from skill of the player, luck, and the benefit of position. Table 3.1 shows that the player in position three (the last player to act) always has a higher

λ	Player	Mean (mbb/g)	Sample Standard Deviation	*Estimated SD 95% CI
0.3	1,2	-93.7215	2711.04	5.31364
	2,1	-177.131	2974.27	5.82958
	3,N	270.852	2487.06	4.87464
	1,3	-92.152	2742.66	5.37561
	2,N	-186.153	2960.24	5.80207
	3,1	278.305	2563.64	5.02473
	1,N	-96.8015	2709.56	5.31075
	2,3	-179.526	2945.62	5.77342
	3,2	276.327	2456.56	4.81486
0.5	1,2	-93.2045	2723.4	5.33787
	2,1	-174.442	2989.94	5.86029
	3,N	267.646	2484.89	4.87039
	1,3	-86.3735	2779.6	5.44801
	2,N	-192.334	2976.09	5.83313
	3,1	278.708	2644.37	5.18296
	1,N	-104.814	2707.04	5.3058
	2,3	-170.766	2912.22	5.70796
	3,2	275.579	2386.75	4.67802
0.9	1,2	-83.2175	2823.85	5.53474
	2,1	-176.58	3092.45	6.0612
	3,N	259.798	2510.98	4.92153
	1,3	-298.743	4026.31	7.89157
	2,N	-282.812	2995.33	5.87085
	3,1	581.554	3793.39	7.43504
	1,N	-153.058	2794.93	5.47806
	2,3	-88.525	2981.18	5.84311
	3,2	241.583	2448.42	4.79891
0.99	1,2	-69.4655	2901.35	5.68665
	2,1	-186.345	3153.23	6.18032
	3,N	255.811	2528.09	4.95505
	1,3	-182.321	3728.09	7.30706
	2,N	-259.239	3057.01	5.99173
	3,1	441.561	3639.56	7.13355
	1,N	-155.818	2812.48	5.51246
	2,3	-72.5558	3077.89	6.03267
	3,2	228.374	2576.63	5.0501
0.999	1,2	-67.7375	2908.19	5.70005
	2,1	-187.542	3159.99	6.19357
	3,N	255.28	2530.73	4.96024
	1,3	-175.559	3706.38	7.26451
	2,N	-260.049	3060.46	5.9985
	3,1	435.608	3626.23	7.1074
	1,N	-153.228	2815.2	5.5178
	2,3	-71.6845	3079.1	6.03503
	3,2	224.913	2573.54	5.04414
0	1,N	-94.722	2707.84	5.30737
	2,N	-181.684	2956.63	5.79499
	3,N	276.406	2491.3	4.88295

Table 3.1: Performance of Players for a set of matches with different lambda values. (* Estimated Standard Deviation for a 95% Confidence Interval.)

value in comparison to positions one and two. This reveals the effect of position on the utility of a player. Remarkably, the player in position three wins the most even when the other two players are colluding against him. However, collusion helps players in positions one and two to lose less than when they are not colluding. A summary of the colluders' average winnings when in the different seats is presented in Table 3.2. These results reveals that the most beneficial collusion occurs when players are in positions one and three.

λ	Colluders' Average Wining for Positions:			Average Colluders' Wining with Rotation
	1 & 2	1 & 3	2 & 3	
0	-276.406	181.684	94.722	0.00000
0.3	-270.8525	186.153	96.801	4.03383
0.5	-267.6465	192.3345	104.813	9.83367
0.9	-259.7975	282.811	153.058	58.69050
0.99	-255.8105	259.24	155.8182	53.08257
0.999	-255.2795	260.049	153.2285	52.66600

Table 3.2: Summary of colluders' average winnings for different positions.

The last column of Table 3.2 shows the average winnings of a pair of colluding agents playing one-million-hand matches of [2-4] Hold'em poker with different λ values. Two colluding agents and one non-colluding agent were created by combining positional strategies. In these games, the dealer button rotates so that all of the players have equal utility gained from their position in the game. These experiments along with our previous results for positional strategies demonstrates that collusion is more beneficial when the λ value is higher. However, the colluders must have a preference for their own utility versus their partners' when all non-colluding players fold. In such cases, setting λ equal or very close to one would result in the colluding players acting randomly. This is shown in Table 3.2, specifically when we compare the average collusion values for $\lambda = 0.9$ and $\lambda = 0.999$. Therefore, λ must be set such that colluders can differentiate between their own utility and their partners' to ensure that they behave believably when the money transitions will be between members of the colluding pair.

Chapter 4

Collusion Tables

In an extensive form game, every action that a player takes has the potential to affect not only on his own utility but also on the utility of all the other players in the game. As each strategy is a distribution over actions and differs from other strategies by the actions it chooses during a game, every strategy also affects the utility of players differently. Collusive strategies not only have an effect on the utility of all players in the game, but since they aim to increase the joint utility of colluders, they explicitly affect the utility of colluders in a manner different from other strategies.

Our idea for detecting collusion takes advantage of this difference. The main component of our approach is called a *collusion table*, which is a data structure that aims to capture the effect of players on the utility of other players. Using collusion tables, one can investigate if colluding and non-colluding strategies affect players utilities differently.

In this chapter, the semantics of a collusion table is described, followed by some examples of collusion tables. A method for creating entries in a collusion table based on a given value function is introduced. Finally, different scoring methods are presented for detecting unusual behaviours that might be signs of collusion using a collusion table.

4.1 Semantics

A *collusion table* is a data structure that is designed to capture the effect of each player's actions on all the players' utilities in a game episode. Each element $C_g(i, j)$

of a collusion table is called a *collusion value* and is meant to represent the effect of column player j's actions on the utility of row player i, during game episode g. If $C_g(i,j)$ is positive, it shows that player j benefited player i by his decision making during game episode g while a negative value for $C_g(i,j)$ indicates that player j's actions during g decreased player i's utility. Table 4.1 shows an example of a collusion table for four players. In this table, the third column represents the effect of player 3's actions on the utility of all players during the game. $C_g(1,3)$ is equal to 5 which means that player 3's actions favor the utility of player 1 while $C_g(4,3) = -3$ reveals that actions taken by player 3 had a negative impact on player 4's utility by 3 units.

Collusion values in a collusion table must satisfy the following two properties:

- The sum of the collusion values in each column of a collusion table must sum up to zero. This demonstrates that a player's actions may increase or decrease the utility of all the players but its overall effect must sum up to zero.

- The sum of the collusion values in each row of a collusion table represents the expected money that the row player can gain.

One can also define a collusion table for a dataset of game episodes D. Every player who participates in at least one game episode of the games in D has a row and a column in this collusion table. Then, a collusion table will be generated for each game episode $g \in D$ and all $C_g(i,j)$ values will be computed (which will be discussed in the next section). Finally, $C_D(i,j)$ will be computed using $C_g(i,j)$ values for all game episodes g, in which both players i and j participated using the following formula:

$$C_D(i,j) = \frac{1}{|G|} \sum_{g \in G} C_g(i,j) \qquad where \qquad G = \{g \in D | i,j \in g\} \tag{4.1}$$

4.2 Design of Collusion Tables

To build the collusion tables as described in the previous section, one must have a method for generating collusion values from game episodes. Our method focuses

Collusion Table	Player 1	Player 2	Player 3	Player 4
Player 1	-3	2	5	-2
Player 2	-2	-3	-4	1
Player 3	4	3	2	-4
Player 4	1	-2	-3	5

Table 4.1: An example of a collusion table

on the change of the game value between two consecutive states for each player during a game episode. We assume that we are given a real-valued function for each agent i on histories $V_i : H \mapsto \mathbb{R}$. We will later describe sources for this value function. Then, given a game episode g, the contribution of agent j to the utility of agent i on a terminal history z_g is defined as:

$$C_g(i,j) = \sum_{\substack{ha \sqsubseteq z_g \\ p(h)=j}} V_i(ha) - V_i(h) \tag{4.2}$$

Similarly, the impact of luck on agent i's performance is as follows:

$$L_g(i) = \sum_{\substack{ha \sqsubseteq z_g \\ p(h)=c}} V_i(ha) - V_{i)}(h) \tag{4.3}$$

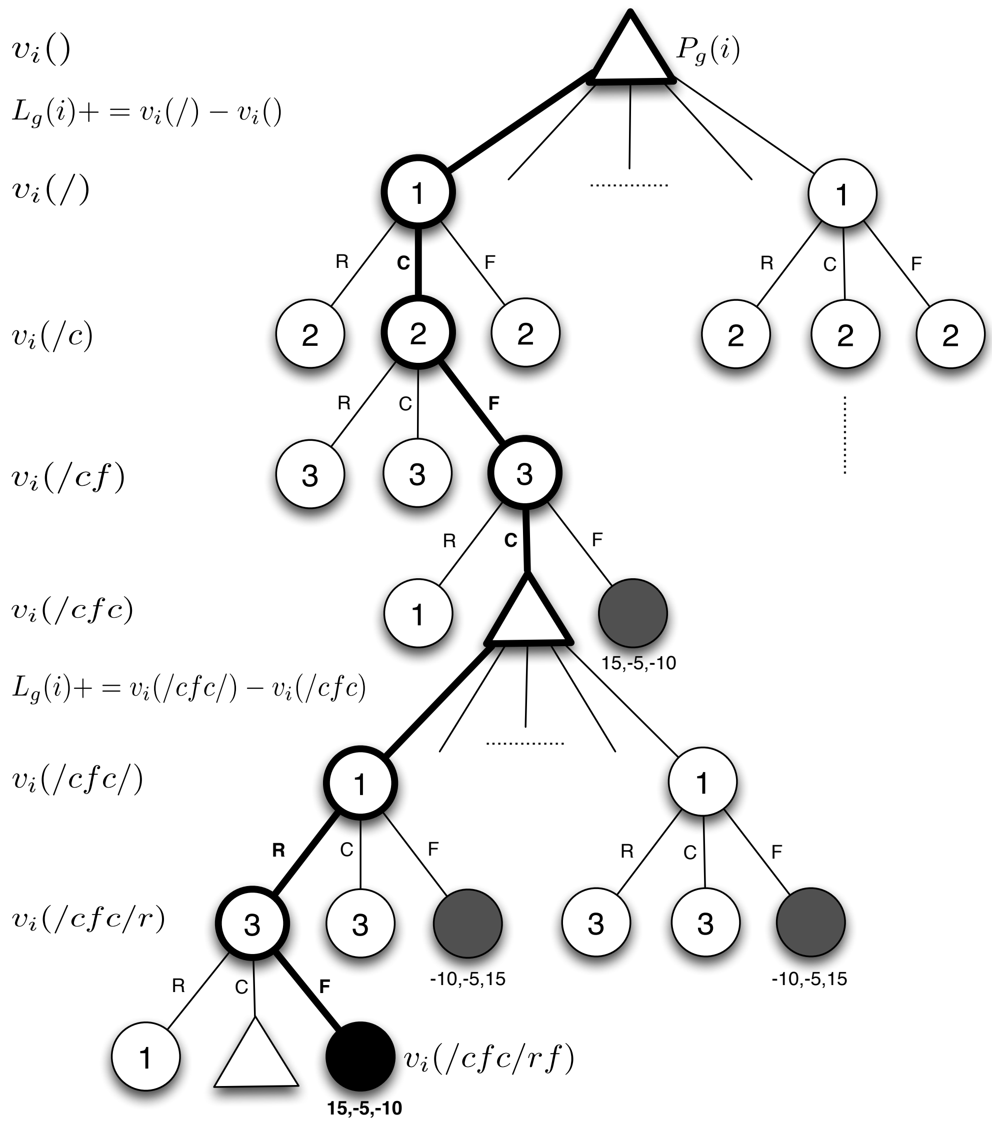
As described in Section 2.4, a player's utility can be divided into skill, luck and position. Based on the definition of $C_g(i,j)$ we have:

$$S_g(i) = \sum_{j \in P_g} C_g(i,j) \tag{4.4}$$

where, instead of the player's skill, $S_g(i)$ now indicates the sum effect of all the players' behaviours on player i's utility. The expected utility of player i in history z_g is now as follows:

$$u_g(i) = \Big[\sum_{j \in P_g} C_g(i,j)\Big] + L_g(i) + \mathrm{Pos}_g(\phi_g(i)) \tag{4.5}$$

Using $C_g(i,j)$ values, we build collusion tables and ignore the effect of luck and position on the utility of players for this study of collusion since a player's only



$$C_g(3,1) = v_3(/c) - v_3(/) + v_3(/cfc/r) - v_3(/cfc/)$$
$$C_g(1,3) = v_1(/cfc) - v_1(/cf) + v_1(/cfc/rf) - v_1(/cfc/r)$$

Figure 4.1: Example of a game episode and the value function for each state. Chance nodes are shown as triangles. In the sequence of actions / represents actions which are taken by chance.

means of colluding is their actions and they cannot use the effect of luck or position for profit. Figure 4.1 shows an example of a poker game tree. In this tree, a game episode is traced in which player 2 folded as his first action and left the game. Player

1 and 3 played until the flop cards are revealed (that is after the second chance node which is shown as a triangle). In this example the value of small blind is 5 and big blind is 10. We denote the players actions as f (*fold*), c (*check* or *call*), and r (*bet* or *raise*). Figure 4.1 shows how the value of luck and the collusion values are computed during a game episode. The effect of player 1 on player 3's utility and the effect of player 3 on player 1's utility are computed as two instances.

4.3 Collusion Scores

Assuming that we have a collusion table which accurately reflects the influence of the agents on each other, we now focus on the problem of determining the degree of collusion exhibited by a pair of players in a collusion table. To do this, we introduce *collusion scores*, the functions θ from pairs of players to real numbers, where the higher a score is, the greater degree of collusion exhibited by the pair.

Collusion scores evaluate collusion tables for any suspicious patterns which might be a sign of collusion. For this step, we propose different functions θ that give scores to each pair of players in accordance with the degree of collusion illustrated by that pair.

Total Impact Score. This scoring function is designed based on the primary objective of colluding players: to increase their joint utility. Hence if two players are colluding during a game episode, their actions specifically affect their joint utility. The Total Impact score computes the total effect of the alleged colluders behaviour on their utility during a game episode g. It is formulated as:

$$\vartheta_g^{TI}(a,b) = \sum_{i \in \{a,b\}} \sum_{j \in \{a,b\}} C_g(i,j) \tag{4.6}$$

In other words, total impact score simply sums up the four values that a pair of players gained in total, due to their own activity.

Marginal Impact Score. When colluding, players differentiate between their opponents and their partners. Therefore, we can expect a gap between a specific player's effect on his partner and the impact he has on his opponents. Marginal Impact score is designed to examine this difference and is computed as follows:

$$\vartheta_g^{MaI}(a,b) = \left(C_g(b,a) - \tfrac{1}{|N|-2}\sum_{\substack{i\in P_g\\ i\notin\{a,b\}}} C_g(i,a)\right) + \left(C_g(a,b) - \tfrac{1}{|N|-2}\sum_{\substack{j\in P_g\\ j\notin\{a,b\}}} C_g(j,b)\right) \tag{4.7}$$

In the above formula, the first term computes the difference between player a's effect on player b and player a's effect on other players in the game on average. The same is computed for player b in the second term. Hence, the marginal impact score investigates how much a pair of players stands out based on the comparison of their behaviour toward each other and other players in the game episode.

Mutual Impact Score. This function is designed to investigate how much positive utility two players transfer to each other. The idea is that a pair of colluders should gain more mutual utility in comparison to any other pair of players in the game. This function is calculated from a game episode g collusion table as follows:

$$\vartheta_g^{MuI}(a,b) = C_g(a,b) + C_g(b,a) \tag{4.8}$$

Minimum Impact Score. The fourth scoring function is based on the idea that both players must be colluding for collusion to really take place. In this measure, the possibility of collusion is investigated based on a player's individual effect on their partnership rather than the total utility that they gain together (i.e. in total impact score). This function is formulated as:

$$\vartheta_g^{MiI}(a,b) = \min_{i\in\{a,b\}} \Big\{ \sum_{j\in\{a,b\}} C_g(j,i) \Big\} \tag{4.9}$$

For each pair of players, this measure reflects the minimum utility that each partner contributed to their joint utility. In this way, the pair of players with the maximum minimum contribution from both of the partners would be ranked first. Therefore, accidental collusion in which the joint utility increases by just one of the partners would not be ranked highly.

	Total Impact	Marginal Impact	Mutual Impact	Min Impact	Differential Total Impact
Player 1-Player 2	-6	-3	0	-5	-14
Player 1-Player 3	8	13	9	1	7
Player 1-Player 4	1	-0.5	-1	-2	-7
Player 2-Player 3	-2	-2	-1	-2	-10
Player 2-Player 4	1	-0.5	-1	-5	-7
Player 3-Player 4	0	-7	-7	-1	-8

Table 4.2: Different collusion scores for a sample collusion table shown in Table 4.1.

Differential Score. This scoring method tries to determine the difference between the score of the most suspicious pair and the rest of the pairs of players in the game. This method can be defined using any other measure of collusion as its base score. Here, we use the Total Impact score as the base scoring method. Differential Total Impact score is computed as follows:

$$\vartheta_g^{DI}(a,b) = \vartheta_g^{TI}(a,b) - \max\left\{ \max_{\substack{d\in P_g \\ d\notin\{a,b\}}} \left\{\vartheta_g^{TI}(a,d)\right\}, \max_{\substack{d\in P_g \\ d\notin\{a,b\}}} \left\{\vartheta_g^{TI}(b,d)\right\} \right\} \quad (4.10)$$

Using this method, the top value among the base scores stands out and all the rest of the pairs of players' values will be negated. Therefore, this method can be used when there is a need to detect the most suspicious pair of players rather than to order the pairs of agents according to their likeliness of collusion.

Table 4.2 demonstrates the different scoring methods for the collusion table example shown in Table 4.1. Based on all of the measures, player 1 and player 3 have the maximum collusion scores which shows that their behaviour is the most suspicious for collusion. The data structure and measurement methods that were described in this chapter provide the basis for our collusion detection system which is experimentally verified in Chapter 7 .

Chapter 5

Creating Value Functions for Sequential Games

To create collusion tables, a method of determining the value for each agent in every history in the game is needed. This value function will be used to determine the effect of each player's actions on all the players' utilities.

We have already seen the similarity between the underlying methods of the advantage sum estimator described in Chapter 2 and the generation of collusion values in a collusion table. These methods can utilize any given value function. Some advantage sum estimators use a hand crafted domain specific value function (like DIVAT [2]) while others use a learned value function using features of the domain (like MIVAT [23]).

In this chapter, three value functions are introduced: *Always call*, *Purified CFR*, and *PIVAT*. The first two methods take advantage of a base strategy to implicitly define the value function, while the last method learns the value function based on the features of the domain. Note that each of these value functions are learned for each position (player) in the game. Then, a set of value functions is used to evaluate a specific agent in different positions.

5.1 Value Functions Based on a Strategy

Any strategy can be used as the basis of defining a value function for a game. The value function then is simply the expected utility gained from any game state for each player if all the players were to use the base strategy to make their decisions during the remainder of the game episode. If a good non-colluding strategy is used as a value function and it reveals that the utility gained by a group of players is unexpectedly high, those players are more likely to be colluding. In this section, two strategies are described that are used in our experiments as a value function. The first one is a very simple strategy while the second one is one of the state of the art strategies that have been shown to have good performance in large extensive form games.

5.1.1 Always Call

The first strategy that we utilized to define a value function is called *always call*. This strategy selects the "call" action in all game states and for all players without considering the private or public cards. In order to compute the value of the game in a history for a player i, it assumes that all the participants would call for the rest of the game. This strategy is naive but has been shown in other domains to yield significant reduction in variance [18].

5.1.2 Purified CFR

The second value function used in this work is defined by a strategy that is generated using Counterfactual Regret Minimization method (CFR), a no-regret learning algorithm [16, 28]. In order to increase the speed of evaluation in each game state, we de-randomized the base CFR strategy by deterministically choosing the action assigned the highest probability by the CFR base strategy in each information set, a technique which is introduced by Ganzfried et al. [5]. This value function is named *purified CFR*.

5.2 Implementation of Value Functions

The Always call value function was implemented enumerating all the possible outcomes at a chance node. As an example, a small experiment is performed consisting of 3 positional agents which play using identical non-colluding CFR strategies for a set of one million hands. Table 5.1 shows the collusion table for these games, as

	Agent 1	Agent 2	Agent 3
Agent 1	332.389024	-214.061441	-213.230632
Agent 2	-243.602725	420.087003	-358.769791
Agent 3	-88.786299	-206.025562	572.000423

Table 5.1: Collusion table for 3 non-colluding positional agents evaluated by always call policy (mbb/g).

	Agent 1	Agent 2	Agent 3
Agent 1	-1.08161	-0.274575	0.443451
Agent 2	0.785583	0.804984	1.06696
Agent 3	0.296089	-0.530411	-1.51039

Table 5.2: Collusion table for 3 non-colluding positional agents evaluated by determinized CFR policy (mbb/g).

evaluated using the always call defined value function.

We also implemented a simplified version of the Purified CFR value function to speed up the evaluation process. In this implementation, to determine the value of a chance node, only a single chance outcome is sampled, instead of enumerating all outcomes as with the always call value function. We also evaluated another set of one million game episodes using purified CFR. In these games, 3 positional agents played using identical non-colluding CFR strategies. Table 5.2 shows the resulting collusion table for this set of games.

The values in Table 5.1 are rather large, demonstrating that the strategy that players actually used in the game episodes was quite different from the policy used to define the value function. In such a case, the policy would compute an expected value of player i for a history h based on the strategy used as the value function. The player then would take action a and reach history ha while the value function assumed that it would choose another action a' and reaches ha'. Therefore, the collusion value computed from the values of the game in histories h and ha may become large. On the other hand, Table 5.2 shows much lower values. This is due to the greater degree of similarity between the value function strategy and the strategies used by the players in the game episodes.

5.3 PIVAT

The two policies discussed earlier use a base strategy. The quality of evaluation depends on how close the base strategy and the players' strategies are. Therefore, a better method may be to use a dataset of games previously played by agents and learn a value function for that population. Using such a value function improves collusion detection technique since it is tailored to the target population and can also reduce the variance as shown by White and Bowling [23].

The third value function examined in this thesis is called the *PIVAT* value function which learns an estimator of collusion values. This method is based on similar ideas from the DIVAT and MIVAT assessment tools. In this approach, a value function is designed for each player based on a set of features for the game.

5.3.1 Finding the Value Function

The main goal for the PIVAT value function is to minimize the variance of collusion values so as to better estimate the effect of players' actions on the utilities of all players. Assume we are given a value function $V_i : H \mapsto \mathbb{R}$ for each player i. We first reformulate the impact of player j's actions on player i's utility during game episode g as:

$$C_g(i,j) = \mathbf{1}\left[P_{-1}(z_g) = j\right]\left(u_i(z_g) - V_i(z_g)\right) + \sum_{\substack{ha \sqsubseteq z_g \\ P(h)=j}} V_i(ha) - V_i(h) \tag{5.1}$$

in which $P_{-1}(z_g)$ indicates the player who made the last move before the game reached terminal history z. In this formula, the indicator function $(1\left[P_{-1}(z_g) = j\right])$ is equal to 1 when the last player was player j, otherwise it equals 0.

Note that the variance of collusion values will depend on the strategies which players use in the game episodes. Our goal is to learn a better value function using the past interactions of players. Therefore, to estimate the variance, some samples of game episodes $G = \{g_1, g_2, \ldots, g_T\}$ are utilized. Our objective is then to solve the following optimization problem given a dataset of game episodes G:

$$\min_{V_i \in \mathbb{V}_i} \sum_{j \in N} \text{Var}(C_G(i,j)) \tag{5.2}$$

where $\mathbb{V}_i$ is the class of all value functions we are considering. The variance of a collusion value in a game episode g can be written as:

$$\begin{aligned} \text{Var}(C_G(i,j)) &= \mathbb{E}_{g \sim G}\left[(C_G(i,j) - \bar{C}(i,j))^2\right] \\ &\approx \tfrac{1}{T}\sum_{t=1}^{T}\left[\left(C_{g_t}(i,j) - \frac{1}{T}\sum_{t'=1}^{T} C_{g_{t'}}(i,j)\right)^2\right] \end{aligned} \tag{5.3}$$

To ensure that the estimator using this value function satisfies the properties of a collusion table, the zero-luck constraint must be satisfied. Thus means that $E[L_{V_j}|\sigma] = 0$. This constraint is equivalent to requiring that for each player, the value of a state right before chance is to act must be equal to the expected value of the states right after the chance node. If this constraint is satisfied, the expected money gained by a player i will be equal to the sum effect of all the players' behaviour on player i's utility. As explained in Chapter 2, in order to satisfy this constraint, we define the value of a chance node as follows:

$$V_i(h \;\; s.t. \;\; P(h) = c) = \sum_{a \in A(h)} \sigma_c(a|h) V_i(ha) \tag{5.4}$$

This guarantees that the value function will be unbiased.

5.3.2 Linear Value Functions

We focus on the class of linear value functions, since we want the optimization to be tractable. Define a feature mapping $\lambda : H \mapsto \mathbb{R}^n$ which maps histories to a vector of n real-valued features. We then consider value functions which are a linear combination of these features:

$$V_i(h) = \theta_i^T \lambda(h)$$

for some $\theta_i \in \mathbb{R}^n$. We can now rewrite the definition of collusion values using the new value function.

$$C_g(i,j) = \mathbf{1}\left[P_{-1}(z_g) = j\right]\big(u_i(z_g) - \theta_i^T\lambda(z)\big) + \theta_i^T\left(\sum_{\substack{ha\sqsubseteq z\\P(h)=j}} \lambda(ha) - \lambda(h)\right) \tag{5.5}$$

additionally, the zero-luck constraint can be rewritten as:

$$\lambda(h) = \sum_a \sigma_c(a|h)\lambda(ha)$$

5.3.3 Linear Optimization Objective

In this section, the full optimization objective function is written out and then simplified. Given a dataset of game episodes $G = \{g_1, g_2, \ldots, g_t\}$ and for each player $i \in N$, our goal is:

$$\begin{aligned}\textbf{Minimize:}_{\theta_i\in\mathbb{R}^n} \sum_{j\in N}\Bigg\{\frac{1}{|G|}\sum_{g\in G}\Bigg[\theta_i^T\Bigg(&\sum_{\substack{ha\sqsubseteq z_g\\P(h)=j}} \lambda(ha) - \lambda(h)\Bigg)\\ &+ \mathbf{1}[P_{-1}(z_g) = j]\Big(u_i(z_g) - \theta_i^T\lambda(z_g)\Big)\\ &- \frac{1}{|G|}\sum_{g'\in G}\Bigg[\theta_i^T\Bigg(\sum_{\substack{ha\sqsubseteq z_{g'}\\P(h)=j}} \lambda(ha) - \lambda(h)\Bigg)\\ &\qquad + \mathbf{1}[P_{-1}(z_{g'}) = j]\Big(u_i(z_{g'}) - \theta_i^T\lambda(z_{g'})\Big)\Bigg]\Bigg]^2\Bigg\}\end{aligned} \tag{5.6}$$

This expression can be simplified as follows

$$
\begin{aligned}
\sum_{j\in N}\Bigg\{\frac{1}{|G|}\sum_{g\in G}\Bigg[\theta_i^T\Bigg(&-\mathbf{1}[P_{-1}(z_g)=j]\lambda(z_g)+\sum_{\substack{ha\sqsubseteq z_g\\P(h)=j}}\lambda(ha)-\lambda(h)\Bigg)\\
&+\mathbf{1}[P_{-1}(z_g)=j]\Big(u_i(z_g)\Big)\\
-\frac{1}{|G|}\sum_{g'\in G}\Bigg[\theta_i^T\Bigg(&-\mathbf{1}[P_{-1}(z_{g'})=j]\lambda(z_{g'})+\sum_{\substack{ha\sqsubseteq z_{g'}\\P(h)=j}}\lambda(ha)-\lambda(h)\Bigg)\\
&+\mathbf{1}[P_{-1}(z_{g'})=j]\Big(u_i(z_{g'})\Big)\Bigg]\Bigg]^2\Bigg\}
\end{aligned}
\tag{5.7}
$$

By defining the following shorthand notation,

$$
A_g(i,j)=-\mathbf{1}[P_{-1}(z_g)=j]\lambda(z_g)+\sum_{\substack{ha\sqsubseteq z_g\\P(h)=j}}\lambda(ha)-\lambda(h)
$$

$$
\begin{aligned}
\bar{A}(i,j)&=\frac{1}{|G|}\sum_{g'\in G}\Big[\,(-\mathbf{1}[P_{-1}(z_{g'})=j]\lambda(z_{g'}))+\sum_{\substack{ha\sqsubseteq z_{g'}\\P(h)=j}}\lambda(ha)-\lambda(h)\Big]\\
&=\frac{1}{|G|}\sum_{g'\in G}A_{g'}(i,j)
\end{aligned}
\tag{5.8}
$$

$$
B_g(i,j)=\mathbf{1}[P_{-1}(z_g)=j]u_i(z_g)-\frac{1}{|G|}\sum_{g'\in G}\mathbf{1}[P_{-1}(z_{g'})=j]u_i(z_{g'})
$$

we get the following optimization:

$$
\underset{\theta_i\in\mathbb{R}^n}{\textbf{Minimize:}}\sum_{j\in N}\sum_{g\in G}\left[\theta_i^T\left(A_g(i,j)-\bar{A}(i,j)\right)+B_g(i,j)\right]^2
$$

5.3.4 Linear Optimization Solution

As our objective is convex in θ_i, we can solve this optimization by setting the objective's partial derivative to 0.

$$
\nabla_{\theta_i}J(\theta_i)=\nabla_{\theta_i}\left[\sum_{j\in N}\sum_{g\in G}\left[\theta_i^T\left(A_g(i,j)-\bar{A}(i,j)\right)+B_g(i,j)\right]^2\right]
$$

$$\nabla_{\theta_i} J(\theta_i) = 2 \sum_{j \in N} \sum_{g \in G} \left[A_g(i,j) - \bar{A}(i,j) \right] \left[\theta_i^T \left(A_g(i,j) - \bar{A}(i,j) \right) + B_g(i,j) \right]$$

$$\nabla_{\theta_i} J(\theta_i) = \left[\sum_{j \in N} \sum_{g \in G} \left(A_g(i,j) - \bar{A}(i,j) \right) \left(A_g(i,j) - \bar{A}(i,j) \right)^T \right] \theta_i + \sum_{j \in N} \sum_{g \in G} (A_g(i,j) - \bar{A}(i,j)) B_g(i,j)$$

if we set $\nabla_{\theta_i} J(\theta_i) = 0$ and solve for θ_i^*

$$\theta_i^* = \left[\sum_{j \in N} \sum_{g \in G} \left(A_g(i,j) - \bar{A}(i,j) \right) \left(A_g(i,j) - \bar{A}(i,j) \right)^T \right]^{-1} \times \left[\sum_{j \in N} \sum_{g \in G} (\bar{A}(i,j) - A_g(i,j)) B_g(i,j) \right]$$

The linear value function defined by these weights is the PIVAT value function. The experimental results in this thesis will focus on the other value function methods though. We leave experimental evaluation of PIVAT for future work.

Chapter 6

Experimental Design

6.1 Introduction

In the previous chapters, we described our collusion detection method. To evaluate the effectiveness of this method, a dataset of games played by different types of players is needed. The most desirable dataset for evaluating collusion detection methods would be one composed of real-world, labelled human data. However, due to the unavailability of such a dataset, we designed and constructed a synthetic dataset of poker games with different types of agents. In this chapter, different strategies and the abstractions used to create weak and strong strategies are explained. Finally, the different types of agents that used our designed strategies to create the dataset of games are introduced.

6.2 Creating Different Strategies

The next step in building an experimental platform is to design and create agents. Each agent is constructed by combining a set of positional strategies (one positional strategy for each position in the game). In these experiments, all agents are designed to play 3-player poker games since it is the smallest multi-player poker game in which collusion is possible. Every agent employs a set of strategies to play poker. Positional strategies were built using the Chance Sampling CFR algorithm (a variant of CFR which samples one set of chance outcomes per iteration) [12,28].

Due to the large size of the poker game tree, strategies are built in an abstracted version of the game. An abstracted game shares the same strategic properties with the full game while having fewer game states and information sets [9]. During the running time of the CFR algorithm, the strategies for the players are updated during a set of self-play games. Therefore, in each run of CFR, 3 positional strategies will be generated (that is: one for each position in the game). First, an abstracted game tree is built and initialized for each player in the game. Then, in each iteration and for each information set, the action probabilities for that player are calculated using the accumulated counterfactual regret of playing each possible action. Finally, due to the large number of choices in a chance node, a sampling method will be used instead of enumerating all the possible outcomes. In general, the higher the number of iterations are, the closer to optimal the developed positional strategy will be.

Each created strategy is position dependent and also created in the context of a specific situation, depending on whether the other players are its partners, colluding opponents or normal players. An agent is constructed from three positional strategies. We developed three different kinds of strategies to construct our agents from:

- **Colluder.** This kind of strategy is built using the modified utility function that is described in Chapter 3. For each combination of two out of three positions, a pair of colluding strategies is developed.

- **Defender.** As mentioned before, during iterations of the CFR algorithm a strategy is developed for each position in the game. Therefore, when a pair of positional colluding strategies are being generated, the algorithm also produces a third strategy for the third position. This strategy plays to minimize its regret while playing against colluding opponents. We call this strategy the defender.

- **Normal.** When the CFR algorithm is run without modifying the players' utility functions, three positional strategies are developed which are playing unmodified [2-4] Holdem. These strategies minimize their regret assuming that their opponents are also normal players.

6.2.1 Abstractions

Computing a strategy in a large extensive form game like poker usually requires an enormous amount of memory. Although [2-4] Hold'em is a smaller variation of Texas Hold'em, it still needs a huge amount of computer memory to build and store a strategy for it since it has $\sim 10^{17}$ game states and 1.677×10^{12} information sets. Therefore, abstraction is utilized to reduce the size of the game tree. An abstracted version of the game is created combining similar information sets into one bucket. A strategy will be created then in this abstracted game.

Beside managing the memory size issue, we also use abstractions as a tool to create strategies with different strengths. Based on the method of abstraction and the number of buckets used, the resulting strategies will perform with different strength in the real game. In these experiments, we used abstractions which vary both in size and in the method of abstraction to develop weak and strong strategies. The stronger agents were created using a bigger abstraction while smaller abstractions were utilized to develop strategies for weak agents. These abstractions are introduced below:

- **3700-K-Means Abstraction.** In this abstraction, complete knowledge about the pre-flop is used. That means there is one information set in the abstracted game for every information set in the real game. On the flop round, information sets are grouped into 3700 different buckets. K-means clustering is used to create buckets based on hand strength. This abstraction uses imperfect recall, meaning that the exact pre-flop cards may be forgotten on the flop. We utilized this abstraction to construct strong agents.

- **Ns-Bucket Abstractions.** This abstraction divides the information sets in each round into N fixed buckets based on the hand strength information. These abstractions use perfect recall in contrast with the previous method of abstraction. Two sets of weak agents were developed using 2-bucket and 5-bucket abstractions. The first set of agents can distinguish between 2 buckets on the pre-flop round and 4 buckets on the flop. Similarly, the second set can distinguish between 5 buckets on the pre-flop round and 25 buckets on the flop which is extremely small in comparison with the actual number of information sets. As a result, strategies generated in this abstracted game may play weakly in the real game since they must use same action probabilities for all hands in a bucket, despite the individual hands having considerable differences.

6.3 Corpus of Games

Using the strategies and abstractions that are described in the previous section, we designed agents to form a population of diverse participants. In a real ecosystem of poker games such as an online poker game, users participate with various abilities. We created seven different types of agents by combining the positional strategies described earlier.

- **Colluder A. (CA)** This agent colludes with his partner colluder B when both of them are in a game. When this occurs, Colluder A applies an appropriate Colluder strategy based on his seat and his partner's seat in each game episode. If Colluder B is not one of the players participating in the game episode, Colluder A utilizes a Normal strategy.

- **Colluder B. (CB)** This player colludes with Colluder A when both of them are participating in a game and plays symmetrically.

- **Non-Colluder. (NC)** This agent applies a Normal strategy in all seats during a game with no dependence on who his opponents are.

- **Defender. (DF)** In a real game situation, there may be agents who are suspicious that their opponents are colluding and hence try to play their best response to collusion. The defender agent is designed to represent the group of smart agents who can detect and respond correctly if their opponents are colluding. This agent employs the Defender strategy in such situations. Otherwise, it uses a Normal strategy.

- **Paranoid. (PR)** While Defender agents can always detect occurrence of collusion correctly, there are other agents who are suspicious that their opponents are colluding. They always play a defensive strategy while they may be right or wrong. The Paranoid agent is designed to represent this group and it always employs a Defender strategy regardless of its opponents.

- **Accidental collude-right. (CR)** It is possible that agents are behaving as if they are colluding due to their lack of skill and this can benefit other players in a real game situation. The accidental colluding agents are designed to represent such implicit acts of collusion. This agent always utilizes a colluder strategy that benefits the agent on his right, regardless of that agent's strategy.

- **Accidental collude-left. (CL)** Similar to the Accidental collude-right agent, this agent always employs a colluder strategy designed to collude with the agent on his left.

Three sets of agents were created: one strong set and two weak sets each consisting of seven agents using $\lambda = 0.9$. Strong agents name are in the form of S.x and weak agents are in the form of WN.x in which N is the number of buckets used in their abstraction and x indicates the type of the agent. Every combination of three agents out of fourteen (one strong set and one weak set) played a one-million-hand match (364 sets in total).

Chapter 7

Results and Discussion

In this chapter we present the results of our collusion detection system, run with two of the proposed collusion value functions, for each of our test populations of agents. The results of experiments using the always-call value function are presented first. We do not include the complete experiments for this value function since it does not perform well. Then, the experiments using the purified CFR value function are described. Collusion tables were created for each game episode and combined into one collusion table using formula 4.1 for each match. Then collusion values for each pair of agents obtained in all of the matches in which both had participated were averaged and merged into one collusion value. Finally, we evaluated the final collusion table using all of the collusion scoring methods introduced in Chapter 5. The resulting set of rankings for pairs of agents are presented in this chapter.

7.1 Using the Always-Call Value function

Two one-million-hand matches were played by strong agents to evaluate the performance of our collusion detection method using the always-call value function described in Chapter 5. The first set was played by two colluding agents and one non-colluding agent. The second set was played by two colluding agents and one defending agent. The collusion tables created from these games using the Always-call value function are presented in Tables 7.1 and 7.2 and the resulting ranking from all scoring methods are shown in Tables 7.3 and 7.4.

None of the the scoring methods except the Minimum Impact score could detect

Agent	S.CA	S.CB	S.NC
S.CA	480.653	-208.739	-171.957
S.CB	-280.891	409.611	-175.271
S.NC	-199.761	-200.872	347.228

Table 7.1: Match1: Collusion table for 3 agents in mbb/g using the always-call value function.

Agent	S.CA	S.CB	S.DF
S.CA	461.828	-201.985	-164.893
S.CB	-276.89	388.476	-183.15
S.DF	-184.938	-186.49	348.044

Table 7.2: Match2: Collusion table for 3 agents in mbb/g using always-call value function.

Agent i	Agent j	Total Impact	Mutual Impact	Marginal Impact	Minimum Impact	Differential Total Impact
S.CA	S.NC	456.163	-371.718	84.444	175.271	55.529
S.CA	S.CB	400.634	-489.63	-88.997	199.762	-55.529
S.CB	S.NC	380.696	-376.143	4.553	171.957	-75.467

Table 7.3: Result of different scoring methods for Match1.

Agent i	Agent j	Total Impact	Mutual Impact	Marginal Impact	Minimum Impact	Differential Total Impact
S.CA	S.DF	460.041	-349.831	110.209	183.151	88.612
S.CA	S.CB	371.429	-478.875	-107.447	184.938	-88.612
S.CB	S.DF	366.88	-369.64	-2.762	164.894	-93.161

Table 7.4: Result of different scoring methods for Match2.

the colluding agents in either matches, even among only three pairs of agents. Also, the Minimum Impact score of the colluders is only a little bit higher than the next highest score which shows that it does not distinguish colluding agents reliably. These results show that, even though it has been shown to have good performance in agent evaluation, the always-call value function does not have good performance when used for collusion detection.

7.2 Strong vs. 5s Weak Agents Using Purified CFR Value Function

Since the always-call value function did not work, this led us to develop the Purified CFR value function described in Chapter 5, which we will show, can be used to detect collusion. We now describe the results of matches between strong agents and weak agents that were built based on 5s abstractions. The value function used in these experiments is Purified CFR. In these experiments all of the scoring methods ranked both the strong and weak colluders at or near the top of the rankings. The complete ranking of agents in this experiment is presented in the first section of the Appendix.

7.2.1 Money Winnings

Table 7.5 shows the top twelve pairs of agents ranked based on the money they gained in all the matches in which both participated. This table shows that the strong colluding agents (S.CA and S.CB) gained the most money when playing together, which is a very strong sign that they may have been colluding. However, it does not clearly identify the weak colluding agents (W5.CA and W5.CB) as they are ranked in the twelfth position. Also, two strong agents in a field of generally weak agents may appear to collude. Therefore, using the total money that pair of agents gained can be a sign of collusive behaviour but it has both false positive and false negative results and is not enough to always identify collusion. Figure 7.1 shows the distribution of agent pairs based on the total money they gained across all matches. This histogram reveals that collusion was beneficial for both pairs of colluders as even the weak colluding agents gained a considerable amount of money.

Rank	Agent i	Agent j	Money gained	Rank	Agent i	Agent j	Money gained
1	S.CA	S.CB	86.359	7	S.PR	S.CB	30.483
2	S.DF	S.CA	39.023	8	S.PR	S.CA	30.063
3	S.DF	S.NC	38.920	9	S.PR	S.NC	29.968
4	S.CA	S.NC	37.228	10	S.DF	S.PR	29.544
5	S.DF	S.CB	36.648	11	W5.CA	W5.CB	24.036
6	S.NC	S.CB	36.560	12	S.CL	S.DF	17.747

Table 7.5: Experiment 1: Top twelve ranking of pairs of agents based on money gained in mbb/g.

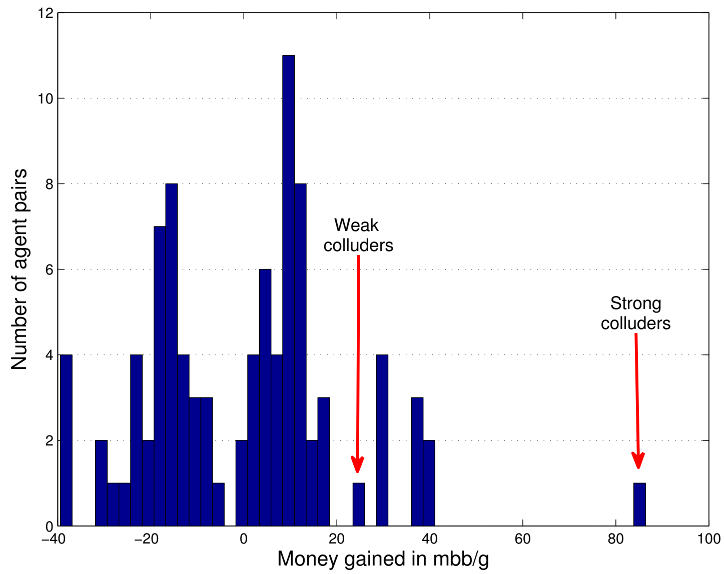


Figure 7.1: Experiment 1: Money gained by pairs of agents histogram.

7.2.2 Total Impact Score

The ranking of agent pairs based on their Total Impact score is presented in Table 7.6. In this table, the strong and weak colluders are ranked as the first and second most likely pairs of agents to be colluding. The strong colluders score is very high when compared with the rest of the agent pair scores. This difference can be seen in the distribution of scores which is shown in Figure 7.2. This shows that not only are both colluders at the top of the ranking, but the strong colluders are clear outliers.

Agent i	Agent j	TI
S.CA	S.CB	1.00555
W5.CA	W5.CB	0.385842
S.CR	W5.CR	0.375167
S.CL	S.CR	0.349336
W5.CL	S.CR	0.091698

Table 7.6: Experiment 1: Top five ranking of pairs of agents based on Total Impact score.

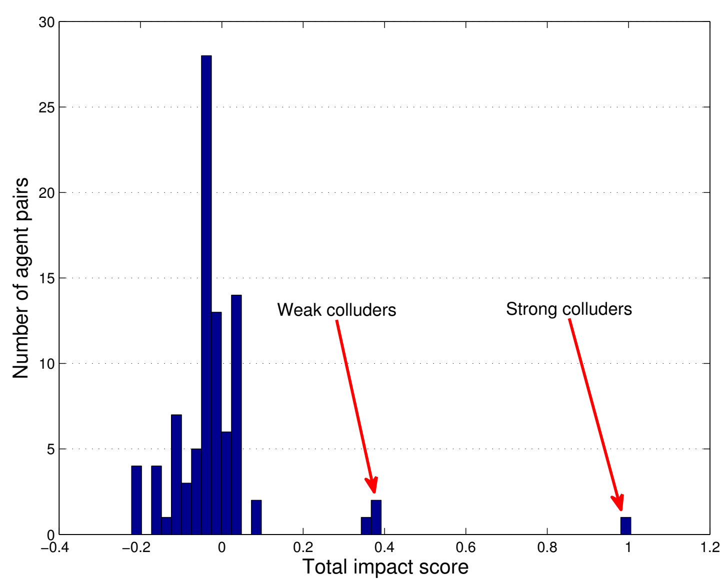


Figure 7.2: Experiment 1: Total Impact Score histogram.

Agent i	Agent j	MaI
W5.CR	S.CR	1.19568
S.CA	S.CB	1.09196
W5.CA	W5.CB	0.454279
S.CL	S.CR	0.219464
S.CL	W5.CR	0.0536221

Table 7.7: Experiment 1: Top five ranking of pairs of agents based on Marginal Impact Score.

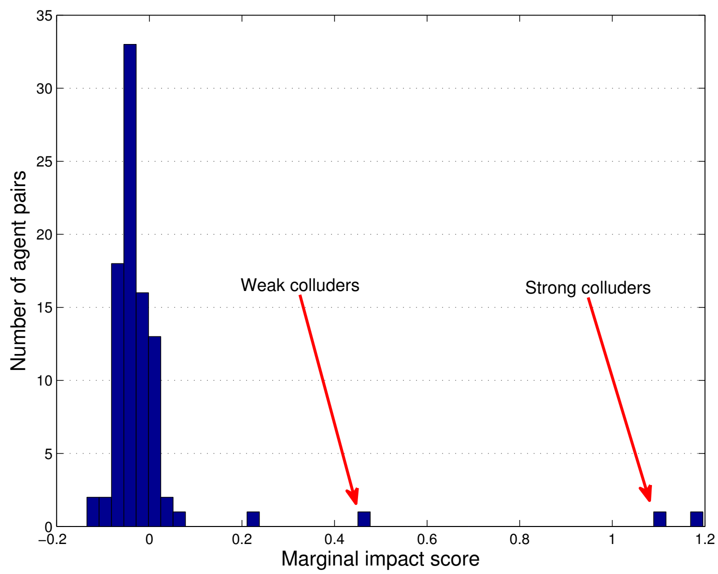


Figure 7.3: Experiment 1: Marginal Impact Score histogram.

7.2.3 Marginal Impact Score

Table 7.7 shows the top five ranking agent pairs based on the Marginal Impact score. Based on this measure, strong and weak colluding agents are ranked second and third respectively which shows that this method also can detect colluders. Strong and weak accidental colluders-right compose the pair of agents with the highest score in this table. The top four rankings seem to be outliers in Figure 7.3. Looking at Table 7.7 one can see that the agent pairs in the top five spots are either colluding agents or accidental colluders. These results show that although they are not intentionally colluding, their behaviour is suspicious. Our method with the Marginal Impact score detects this and further investigation by human experts would be needed to ensure they are not colluding.

7.2.4 Mutual Impact Score

Table 7.8 shows the top eighteen ranking pairs of agents based on the Marginal Impact score. As shown in this table and Figure 7.4, this measure does not appear to adequately detect collusion. This experiment demonstrates that although the colluding agents gain more money in total, our system does not detect that they are transferring utility to each other significantly more than other pairs of agents.

Rank	Agent i	Agent j	MuI	Rank	Agent i	Agent j	MuI
1	S.CR	W5.CR	0.8205	10	S.CB	W5.CR	0.0828874
2	W5.CR	W5.NC	0.177952	11	W5.CR	W5.PR	0.0819924
3	W5.CR	W5.DF	0.17314	12	S.DF	W5.PR	0.0799518
4	W5.CA	W5.CR	0.158265	13	W5.NC	W5.PR	0.0775214
5	W5.CB	W5.CR	0.148728	14	S.CA	W5.CR	0.0747294
6	W5.CR	S.DF	0.145593	15	S.NC	W5.PR	0.0720325
7	W5.CR	S.NC	0.14041	16	W5.DF	W5.PR	0.0711563
8	S.PR	W5.PR	0.0898325	17	W5.CA	W5.CB	0.0684351
9	S.CA	S.CB	0.0864108	18	W5.DF	W5.NC	0.0651412

Table 7.8: Experiment 1: Top eighteen ranking of pairs of agents based on Mutual Impact Score.

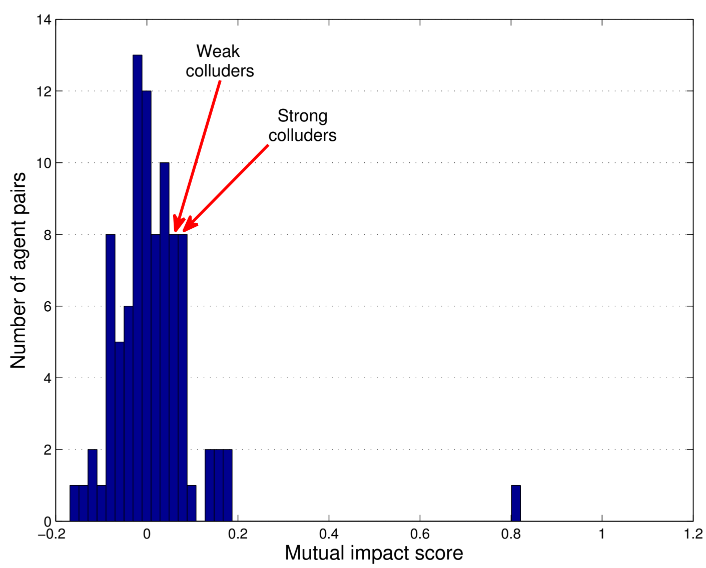


Figure 7.4: Experiment 1: Mutual Impact Score histogram.

7.2.5 Minimum Impact Score

Table 7.9 shows that colluders are close to top of the ranking based on Minimum Impact score. Again, using this measure, both pairs of colluding agents can be distinguished. One nice feature of this scoring function, as shown in this table and also Figure 7.5, is that only the top four pairs have positive scores among all 91 pairs and both colluding pairs are among those. So, this measure appears to attach meaning to the sign of a pair's collusion score.

Agent i	Agent j	MiI
S.CA	S.CB	0.414292
S.CL	S.CR	0.0964158
W5.CA	W5.CB	0.0406547
S.DF	S.NC	0.00535241
S.NC	S.CB	-0.00691819

Table 7.9: Experiment 1: Top five ranking of pairs of agents based on Minimum Impact Score.

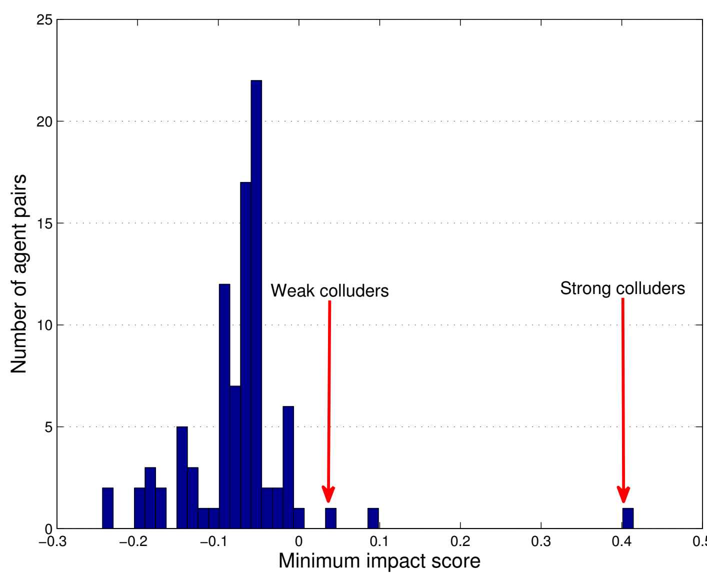


Figure 7.5: Experiment 1: Minimum Impact Score histogram.

Agent i	Agent j	DI
S.CA	S.CB	0.59348
W5.CR	S.CR	0.281338
W5.CA	W5.CB	0.262688
S.CL	S.CR	0.0502188
S.DF	S.PR	0.00218301

Table 7.10: Experiment 1: Top five ranking of pairs of agents based on Differential Total Impact Score.

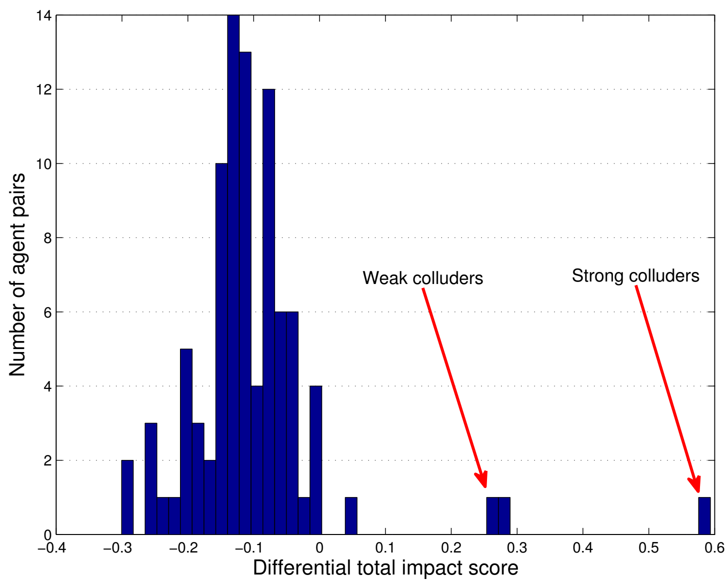


Figure 7.6: Experiment 1: Differential Total Impact Score histogram.

7.2.6 Differential Total Impact Score

The top five ranking pairs of agents based on the Differential Total Impact score are presented in Table 7.10. The strong colluders are detected as the most suspicious pair based on this measure. The complete ranking of pairs, presented in Table A.5, shows that only the first five values are positive. As explained before, this measure tries to magnify the difference between the most suspicious pairs and the rest of the pairs of agents. Figure 7.6 demonstrates that it was successful at this goal. The two colluding pairs are ranked in the first and third position. Also the accidental colluding agents appeared in the second and fourth position which again shows that their behaviour is suspicious and requires further investigation.

7.3 Strong vs. 2s Weak Agents Using Purified CFR Value Function

In this section, the result of a set of matches between strong agents and weak agents that were built using a 2s abstractions (which is described in Chapter 6) are presented. Each of the weak agents that were used in these experiments only had one bit of information (good/bad) about each chance event (i.e. good/bad private cards, good/bad flop cards given these private cards). The value function used in these experiments is Purified CFR. The complete ranking of agents in this experiment are presented in the second section of the Appendix.

Figure 7.7 demonstrates that the strong colluders gained the most money and appeared as an outlier among the other pairs of agents. However, the weak colluding agents did not gain a considerable amount of money. Table A.7 shows that not only did the weak colluders not gain money but also they lost money by –19.0389 mbb/g and were ranked in sixtieth place. This shows that their colluding strategies were not particularly beneficial.

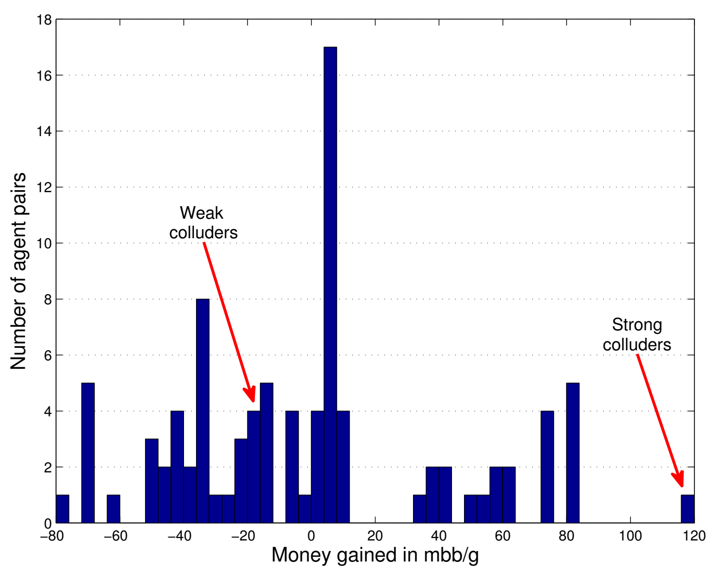


Figure 7.7: Experiment 2: Money gained by pairs of agents histogram.

The histograms in Figures 7.8-7.12 show that the strong colluders were detected by all scoring methods in the first or second place except for Mutual Impact score. This experiment also supports our previous idea that using, Mutual Impact score

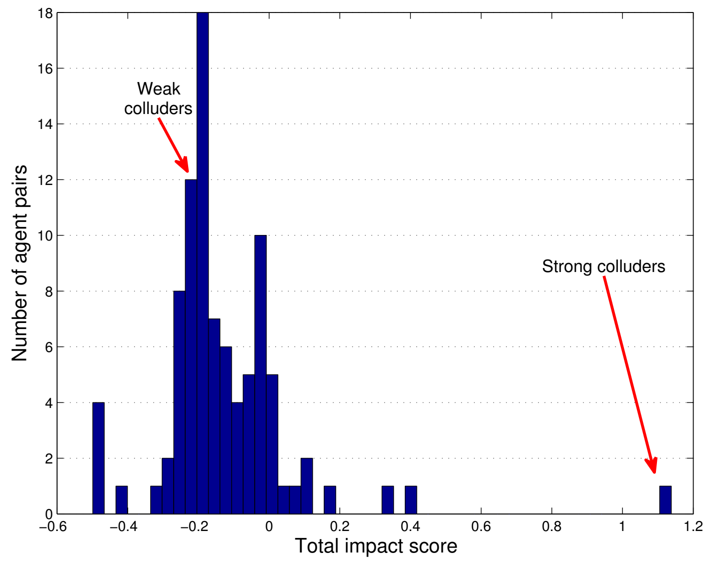


Figure 7.8: Experiment 2: Total Impact Score histogram.

does not well identify colluding agents. However, other scoring methods detected the strong colluders as an obvious outlier when compared with the rest of the pairs' scores.

None of the scoring methods could detect the weak colluders. We hypothesize that this is due to the weak abstractions that the weak colluding agents utilized. Their collusion did not even yield them a profit, so in some sense, they were attempting to collude, but they were not successful. These agents resemble beginner players who try to collude in poker but they just harm each other by trying to collude. Figures 7.8-7.12 show that our scoring methods cannot detect this type of colluders, or rather, those who fail to successfully collude.

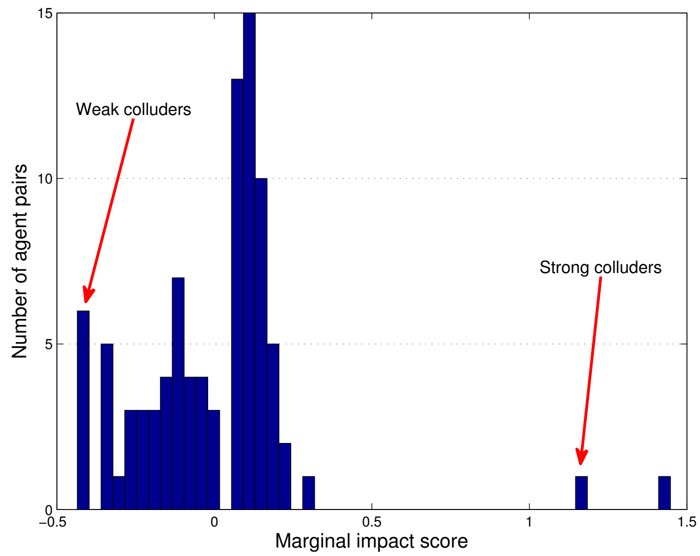


Figure 7.9: Experiment 2: Marginal Impact Score histogram.

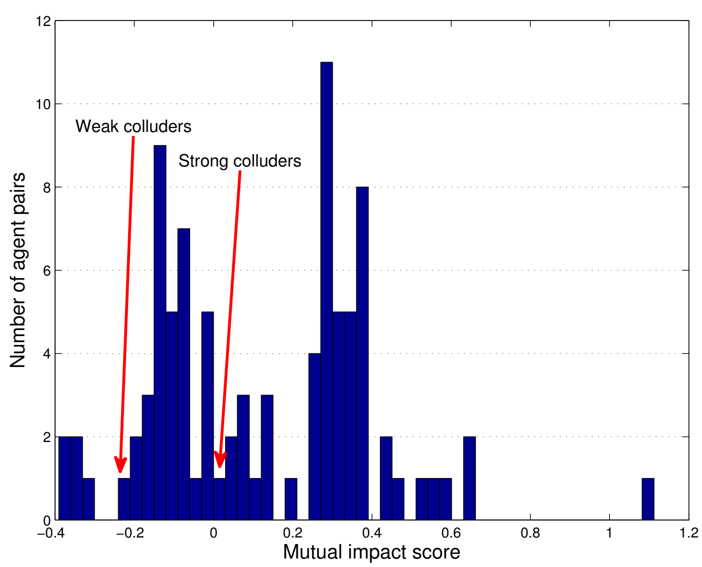


Figure 7.10: Experiment 2: Mutual Impact Score histogram.

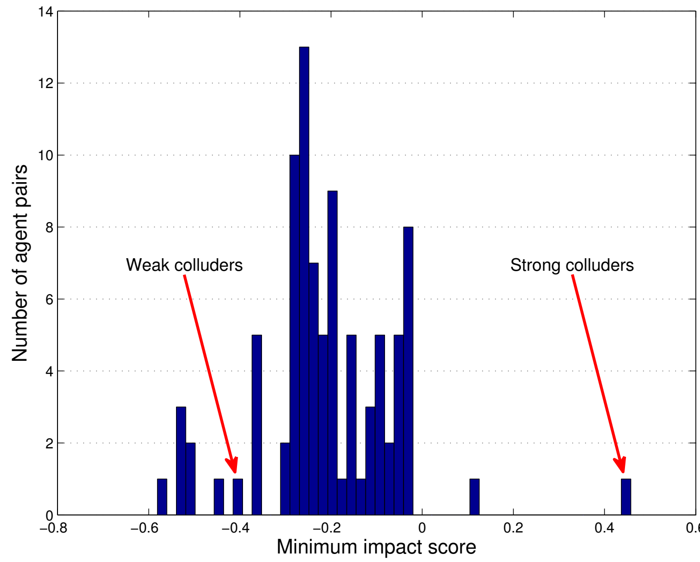


Figure 7.11: Experiment 2: Minimum Impact Score histogram.

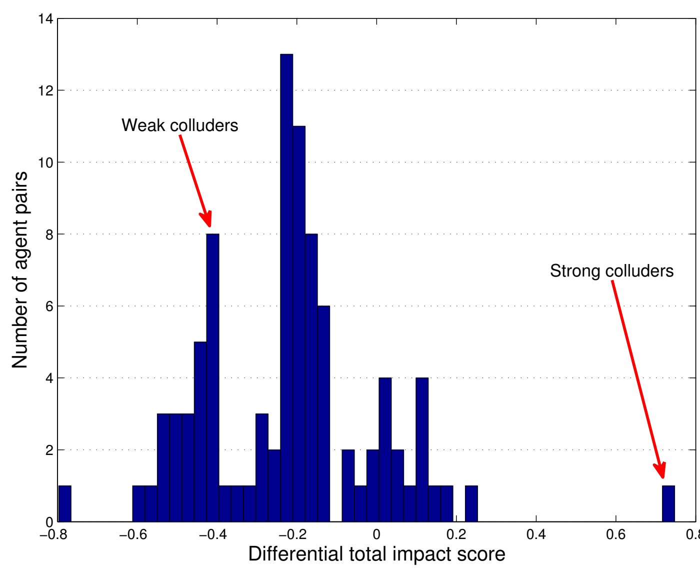


Figure 7.12: Experiment 2: Differential Total Impact Score histogram.

7.4 Discussion

The experiments presented in this chapter show that our collusion detection system using the purified CFR value function can successfully detect collusion if collusion is beneficial for colluding agents. Table 7.11 demonstrates a summary of the performance of the scoring methods on different datasets. It shows that all the scoring methods except the Mutual Impact score can detect strong colluding agents when Purified CFR is used as the value function. However, the Always-call value function does not show promising performance.

All of the scoring methods introduced in this thesis can be used in other multi-agent domains. Therefore, we recommend using a combination of these scoring methods in order to have a robust collusion detection method.

	Purified CFR value function		Always-call value function
Scoring method	Experiment 1	Experiment 2	
Total Impact	Yes	Yes	No
Marginal Impact	Yes	Yes	No
Mutual Impact	No	No	No
Minimum Impact	Yes	Yes	Yes
Differential Total Impact	Yes	Yes	No

Table 7.11: Can different scoring methods detect the strong colluding agents?

Chapter 8

Conclusion

In this section the contributions of this thesis is summarized and several avenues for future work are introduced.

8.1 Contributions

In this thesis we introduced the first implemented and successful automatic collusion detection system. This method ranks pairs of agents in order of the chance that they are colluding. This is done through a novel data structure, called a collusion table, which captures the impact that each agent has on the utility of all agents in the game. We showed how collusion tables can be created using value functions which estimate the expected value of each game state for each player. Three different methods for constructing a value function are introduced and two of them were implemented. We showed how to use collusion tables to detect collusion by introducing five different scoring methods which rank pairs of players in order of the possibility that they are colluding given the collusion values.

Poker was utilized as our test domain because of its well-defined formal structure and high similarities to a real-world system in which collusion is prohibited. We developed a population of colluding and non-colluding agents in this domain and created a synthetic dataset of collusive behaviour to evaluate our technique. We evaluated our collusion detection method and showed that our method can successfully detect the colluding agents in our dataset.

This work has two main contributions. First, we developed an automated collusion detection method and extended the applicability of the advantage sum idea to the domain of collusion detection. This technique does not require any hand-crafted features or depend on any characteristics of the domain. Second, we designed a general collusion method and constructed colluding agents that gain more utility through their collusion in our test domain. We created a synthetic dataset of agents using this method, which could be used in future research on collusion.

8.2 Future Work

There are three main directions for future work:

- We tested our technique using [2-4] Hold'em poker which is a smaller variation of Texas Hold'em poker. Therefore, it can be easily modified to evaluate corpus of games in Annual Computer Poker Competition in which bots are playing Texas Hold'em poker. This work was a first step to detecting collusion in human settings. The next step would be to evaluate this technique in a real-world corpus. Human datasets could be huge in size. A collusion detection system might need to employ a high level filtering of the data first, and then collusion tables could be constructed. Although finding sufficient labelled human data is a challenge, we hope to be able to find such data and apply this method in the near future.

- In this thesis, the PIVAT value function was introduced and the theoretical advantage of using it is shown. We are interested in implementing this technique and evaluating its effectiveness in future collusion detection methods.

- Finally, this method can be extended to other sequential decision making domains, since it does not rely on any characteristics of poker. It would be interesting to investigate reformulating our approach to be used in the regulation of market places in which detecting coalitions of parties is a real-world challenge. While there is a vast literature on preventing coalition by designing auctions in which collusion is not the preferable strategy, there have not been as many studies on detecting collusion when it might have occurred.

We are interested in extending our technique and similarly utilizing collusion tables and scoring methods to identify collusive behaviour in market places.

Bibliography

[1] Y. Bachrach, M. Zadimoghaddam, and P. Key. A cooperative approach to collusion in auctions. *ACM SIGecom Exchanges*, 10(1):17–22, 2011.

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

[3] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[4] A. C. P. Competition. Annual computer poker competition 2012 participants. http://www.computerpokercompetition.org/index.php?option=com_content &view=article&id=108%3Aparticipants-2012&catid=35%3Aparticipants& Itemid=62&limitstart=3, 2012.

[5] S. Ganzfried, T. Sandholm, and K. Waugh. 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, 2012.

[6] R. Gibson and D. Szafron. On strategy stitching in large extensive form multiplayer games. In *Proceedings of the Twenty-Fifth Conference on Neural Information Processing Systems (NIPS)*, 2011.

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

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

[9] M. B. Johanson. Robust strategies and counter-strategies: Building a champion level computer poker player. Master's thesis, University of Alberta, 2007.

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

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

[12] M. Lanctot, K. Waugh, M. Zinkevich, and M. Bowling. Monte carlo sampling for regret minimization in extensive games. *Advances in Neural Information Processing Systems*, 22:1078–1086, 2009.

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

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

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

[16] N. A. Risk and D. Szafron. 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, pages 159–166, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems.

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

[18] D. Schnizlein. *State translation in no-limit poker*. PhD thesis, Citeseer, 2009.

[19] J. Smed, T. Knuutila, and H. Hakonen. Can we prevent collusion in multi-player online games? In *Proceedings of the Ninth Scandinavian Conference on Artificial Intelligence 2006*, pages 168–175, 2006.

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

[21] M. J. Watson. 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.

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

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

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

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

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

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

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

Appendix A

Complete Results of Different Collusion Scores

A.1 Experiment 1: Strong v.s. 5s Weak Agents

Rank	Agent i	Agent j	Money
1	S.CA	S.CB	86.35875
2	S.DF	S.CA	39.023
3	S.DF	S.NC	38.92045833
4	S.CA	S.NC	37.22775
5	S.DF	S.CB	36.64808333
6	S.NC	S.CB	36.55991667
7	S.PR	S.CB	30.48295833
8	S.PR	S.CA	30.06370833
9	S.PR	S.NC	29.96825
10	S.DF	S.PR	29.54375
11	W5.CA	W5.CB	24.03591667
12	S.CL	S.DF	17.74708333
13	S.CL	S.NC	17.648625
14	S.CL	S.CR	17.43766667
15	S.CL	S.CA	15.15670833
16	S.CL	S.CB	14.83466667
17	W5.NC	S.NC	11.95775
18	S.DF	W5.DF	11.906375
19	S.CL	S.PR	11.8565
20	S.DF	W5.NC	11.709125
21	W5.DF	S.CB	11.52475
22	S.DF	W5.CB	11.12375
23	W5.NC	S.CA	11.053875
24	W5.DF	S.NC	11.02204167
25	W5.NC	S.CB	10.75641667
26	W5.CA	S.NC	10.53875
27	W5.DF	S.CA	10.34166667
28	S.DF	W5.CA	10.08333333
29	S.NC	W5.CB	9.952
30	S.DF	S.CR	9.400333333
31	S.NC	W5.PR	9.285958333
32	W5.CA	S.CB	9.182041667
33	S.DF	W5.PR	8.943875
34	S.CA	W5.CB	8.523208333
35	S.CR	S.NC	8.430416667
36	W5.CA	S.CA	8.293625
37	S.CB	W5.CB	7.840333333
38	S.CB	W5.PR	7.190625
39	S.CA	W5.PR	7.091333333
40	W5.DF	S.PR	4.324291667
41	W5.NC	S.PR	3.983295833
42	S.PR	S.CR	3.784120833
43	S.PR	W5.PR	3.681491667
44	S.CA	S.CR	3.419370833
45	S.CR	S.CB	3.393591667
46	S.PR	W5.CA	3.05115
47	S.PR	W5.CB	2.879204167
48	W5.CL	S.NC	1.334770833
49	W5.CL	S.DF	1.0584125
50	W5.CL	S.CA	0.400245833
51	W5.CL	S.CB	0.367283333
52	W5.CL	S.PR	-6.60775
53	S.CL	W5.DF	-6.988583333
54	S.CL	W5.PR	-7.10075
55	S.CL	W5.NC	-7.491333333
56	S.CL	W5.CA	-9.471291667
57	S.CL	W5.CR	-9.997125
58	S.CL	W5.CB	-10.02333333
59	W5.NC	S.CR	-11.97958333
60	W5.DF	S.CR	-13.15395833
61	S.CR	W5.PR	-13.854875
62	W5.DF	W5.NC	-14.18716667
63	W5.CR	S.NC	-14.3355
64	W5.CL	S.CR	-14.6155
65	S.DF	W5.CR	-15.23325
66	W5.DF	W5.CA	-15.98116667
67	W5.CA	S.CR	-16.04820833
68	S.PR	W5.CR	-16.14225
69	W5.DF	W5.CB	-16.2335
70	S.CR	W5.CB	-16.61233333
71	W5.NC	W5.CA	-16.78558333
72	W5.NC	W5.CB	-17.32033333
73	W5.NC	W5.PR	-17.53195833
74	W5.CA	W5.PR	-17.56933333
75	W5.PR	W5.CB	-17.79466667
76	W5.DF	W5.PR	-18.39758333
77	W5.CR	S.CA	-18.846
78	W5.CR	S.CB	-20.44754167
79	W5.CL	S.CL	-20.86691667
80	W5.CL	W5.DF	-22.57908333
81	W5.CL	W5.NC	-23.06370833
82	W5.CL	W5.PR	-23.479375
83	W5.CL	W5.CB	-24.25791667
84	W5.CL	W5.CA	-24.50579167
85	W5.CR	S.CR	-28.86554167
86	W5.CL	W5.CR	-29.55108333
87	W5.CR	W5.PR	-30.32220833
88	W5.DF	W5.CR	-36.97533333
89	W5.NC	W5.CR	-37.356375
90	W5.CA	W5.CR	-39.35104167
91	W5.CR	W5.CB	-39.417625

Table A.1: E1-Agent pairs ranked according to the total money gained (mbb/g).

Rank	Agent i	Agent j	TI
1	S.CA	S.CB	1.00555
2	W5.CA	W5.CB	0.385842
3	S.CR	W5.CR	0.375167
4	S.CL	S.CR	0.349336
5	W5.CL	S.CR	0.091698
6	S.CL	W5.CR	0.0797685
7	S.CL	S.DF	0.0432276
8	S.CR	S.PR	0.0418236
9	S.CL	S.NC	0.0393955
10	S.CA	S.DF	0.0376595
11	S.CB	S.DF	0.0374919
12	S.CL	S.CB	0.0366992
13	S.CL	S.CA	0.0331157
14	S.CL	S.PR	0.0308204
15	S.CA	S.NC	0.0284244
16	S.CA	S.PR	0.0283438
17	W5.CL	S.CB	0.0270478
18	S.CB	S.PR	0.0265555
19	S.CB	S.NC	0.0262797
20	S.DF	S.NC	0.0252142
21	W5.CL	S.NC	0.0227646
22	W5.CL	S.DF	0.0213757
23	W5.CL	S.CA	0.02031
24	S.DF	S.PR	0.00878357
25	S.NC	S.PR	0.0080518
26	S.CR	W5.PR	0.00407007
27	S.CL	W5.DF	-0.00850434
28	W5.CL	W5.DF	-0.00999859
29	W5.CL	S.PR	-0.0114101
30	W5.CL	W5.NC	-0.0128569
31	W5.CL	W5.PR	-0.0152997
32	S.CR	S.DF	-0.0153364
33	S.CL	W5.PR	-0.0172955
34	S.CL	W5.NC	-0.0182976
35	W5.CL	W5.CB	-0.0185488
36	S.CR	S.NC	-0.0198932
37	W5.CL	W5.CA	-0.0211479
38	S.CB	W5.DF	-0.0220208
39	S.CA	W5.DF	-0.0235
40	W5.CB	S.DF	-0.0278489
41	W5.CA	S.DF	-0.0286196
42	S.CB	W5.NC	-0.0290623
43	S.CA	W5.NC	-0.0316346
44	S.PR	W5.CA	-0.0318599
45	S.CL	W5.CA	-0.0319417
46	S.CL	W5.CB	-0.0324459
47	S.DF	W5.DF	-0.0326703
48	S.DF	W5.NC	-0.0337125
49	W5.CA	S.CB	-0.0350187
50	W5.CB	S.NC	-0.0358736
51	W5.CL	W5.CR	-0.0359578
52	S.NC	W5.NC	-0.0367932
53	W5.CA	S.NC	-0.0369764
54	S.CB	W5.CB	-0.0374961
55	S.PR	W5.CB	-0.0381553
56	S.CA	S.CR	-0.0386265
57	S.CA	W5.CB	-0.0391742
58	S.CB	W5.PR	-0.0403472
59	W5.DF	S.NC	-0.0415002
60	S.CA	W5.PR	-0.0418506
61	W5.DF	S.PR	-0.0418566
62	W5.CA	S.CA	-0.0422953
63	S.CR	W5.DF	-0.0446311
64	W5.NC	S.PR	-0.0449639
65	S.PR	W5.PR	-0.0465247
66	S.CB	S.CR	-0.0479099
67	S.CR	W5.NC	-0.0485878
68	W5.CL	S.CL	-0.0501718
69	S.DF	W5.PR	-0.055913
70	S.NC	W5.PR	-0.0573571
71	W5.CA	S.CR	-0.0605756
72	W5.CB	S.CR	-0.0642481
73	W5.CA	W5.DF	-0.0924057
74	W5.CB	W5.DF	-0.093758
75	W5.DF	W5.NC	-0.0960106
76	W5.CA	W5.NC	-0.101144
77	W5.CA	W5.PR	-0.103189
78	W5.CB	W5.NC	-0.104252
79	W5.NC	W5.PR	-0.10856
80	W5.CB	W5.PR	-0.111402
81	W5.CR	S.PR	-0.117305
82	W5.DF	W5.PR	-0.119467
83	W5.CR	W5.PR	-0.144366
84	S.CA	W5.CR	-0.156408
85	S.CB	W5.CR	-0.15718
86	W5.CR	S.NC	-0.162669
87	W5.CR	S.DF	-0.166082
88	W5.CR	W5.DF	-0.213907
89	W5.CA	W5.CR	-0.215197
90	W5.CR	W5.NC	-0.218744
91	W5.CB	W5.CR	-0.221886

Table A.2: E1-Agent pairs ranked according to the Total Impact Score.

Rank	Agent i	Agent j	MaI
1	W5.CR	S.CR	1.19568
2	S.CA	S.CB	1.09196
3	W5.CA	W5.CB	0.454279
4	S.CL	S.CR	0.219464
5	S.CL	W5.CR	0.0536221
6	S.PR	W5.PR	0.0433074
7	S.CL	S.DF	0.0274009
8	S.DF	W5.PR	0.0240394
9	S.CL	S.NC	0.0217852
10	S.DF	W5.NC	0.019442
11	S.DF	W5.DF	0.0187874
12	S.NC	W5.PR	0.0146756
13	W5.NC	S.NC	0.0113063
14	S.CR	W5.PR	0.00746623
15	W5.DF	S.PR	0.00708189
16	W5.CL	S.NC	0.00263166
17	W5.DF	S.NC	0.00261103
18	W5.NC	S.PR	-0.000762369
19	W5.CL	S.DF	-0.000915462
20	S.CL	W5.DF	-0.00140727
21	S.NC	W5.CB	-0.00310696
22	W5.DF	S.CR	-0.003712
23	S.DF	S.PR	-0.00618053
24	S.DF	W5.CA	-0.008558
25	S.DF	W5.CB	-0.00874559
26	W5.CA	S.NC	-0.0111532
27	S.PR	S.NC	-0.0113794
28	W5.NC	S.CB	-0.0117628
29	S.CL	W5.NC	-0.0139017
30	S.PR	W5.CA	-0.0141171
31	W5.NC	S.CR	-0.0186076
32	S.DF	S.NC	-0.0199818
33	S.DF	W5.CR	-0.0204909
34	S.PR	W5.CB	-0.0210138
35	W5.CR	S.NC	-0.0222606
36	W5.NC	S.CA	-0.0234142
37	S.CL	W5.CA	-0.0291203
38	W5.CL	W5.DF	-0.0296952
39	W5.DF	W5.NC	-0.0308692
40	W5.NC	W5.PR	-0.0310388
41	S.CL	W5.CB	-0.031309
42	W5.CL	S.PR	-0.0319363
43	W5.CA	S.CR	-0.0332808
44	W5.CL	S.CB	-0.0342883
45	W5.DF	S.CB	-0.0348782
46	S.CB	W5.PR	-0.0352628
47	S.CL	S.PR	-0.0369395
48	S.CL	S.CA	-0.0369653
49	S.CL	S.CB	-0.0373405
50	W5.CL	W5.NC	-0.0378956
51	S.CA	W5.PR	-0.0380722
52	W5.CA	S.CB	-0.0384728
53	W5.DF	S.CA	-0.0404635
54	S.CR	W5.CB	-0.0405878
55	W5.DF	W5.CR	-0.0407666
56	W5.NC	W5.CR	-0.0407915
57	W5.CL	S.CA	-0.0452298
58	S.CB	W5.CB	-0.0455843
59	S.CA	W5.CB	-0.0465935
60	S.PR	S.CR	-0.0467116
61	W5.DF	W5.PR	-0.0483111
62	S.CA	S.NC	-0.0494529
63	W5.CA	W5.PR	-0.051535
64	W5.CA	S.CA	-0.0518725
65	S.NC	S.CB	-0.0522755
66	W5.NC	W5.CB	-0.0527681
67	W5.DF	W5.CA	-0.0533426
68	W5.NC	W5.CA	-0.0535793
69	W5.PR	W5.CB	-0.0548547
70	W5.CL	W5.PR	-0.055735
71	W5.DF	W5.CB	-0.0561678
72	W5.CA	W5.CR	-0.0569326
73	S.CL	W5.PR	-0.057471
74	S.PR	S.CA	-0.0594118
75	W5.CL	S.CL	-0.0621232
76	W5.CR	W5.PR	-0.062374
77	W5.CL	W5.CB	-0.0634381
78	W5.CL	W5.CA	-0.0671638
79	S.PR	S.CB	-0.0684417
80	S.DF	S.CR	-0.0700778
81	S.PR	W5.CR	-0.0717068
82	W5.CR	W5.CB	-0.0731583
83	W5.CR	S.CB	-0.0742928
84	W5.CL	S.CR	-0.0772063
85	S.DF	S.CA	-0.0804722
86	S.DF	S.CB	-0.0806836
87	W5.CL	W5.CR	-0.0808094
88	W5.CR	S.CA	-0.0816774
89	S.CR	S.NC	-0.0852464
90	S.CA	S.CR	-0.112482
91	S.CR	S.CB	-0.134897

Table A.3: E1-Agent pairs ranked according to the Marginal Impact Score.

Rank	Agent i	Agent j	MiI
1	S.CA	S.CB	0.414292
2	S.CL	S.CR	0.0964158
3	W5.CA	W5.CB	0.0406547
4	S.DF	S.NC	0.00535241
5	S.NC	S.CB	-0.00691819
6	S.DF	S.CA	-0.00702479
7	S.CA	S.NC	-0.0071533
8	S.DF	S.CB	-0.00797061
9	S.DF	S.PR	-0.0136976
10	S.PR	S.NC	-0.0137525
11	S.PR	S.CB	-0.0218562
12	S.PR	S.CA	-0.0223335
13	S.CL	S.DF	-0.0413776
14	W5.CL	S.NC	-0.0442192
15	S.CL	S.NC	-0.0461937
16	W5.CL	S.CR	-0.0462927
17	W5.CL	S.DF	-0.0467339
18	S.DF	W5.NC	-0.0479013
19	W5.NC	S.NC	-0.0518318
20	S.DF	W5.DF	-0.0525409
21	W5.CL	S.CB	-0.0528611
22	S.DF	W5.CA	-0.0532672
23	S.DF	W5.CB	-0.0536039
24	S.PR	W5.CA	-0.0541288
25	S.NC	W5.CB	-0.0546656
26	W5.DF	S.NC	-0.0560043
27	W5.CA	S.NC	-0.0565428
28	W5.CA	S.CB	-0.0566923
29	W5.CL	S.CA	-0.0568909
30	W5.DF	S.PR	-0.0571369
31	S.PR	W5.CB	-0.0578862
32	W5.DF	S.CB	-0.0582174
33	S.CL	S.PR	-0.0585908
34	W5.NC	S.CA	-0.0585909
35	S.DF	S.CR	-0.0587645
36	W5.NC	S.PR	-0.0590812
37	S.CL	S.CB	-0.059765
38	W5.NC	S.CB	-0.0599747
39	S.CA	W5.CB	-0.0618488
40	W5.CL	W5.CB	-0.0619993
41	S.CR	S.NC	-0.0624509
42	W5.DF	S.CA	-0.0636885
43	S.PR	S.CR	-0.0637637
44	S.CB	W5.CB	-0.0638164
45	S.PR	W5.PR	-0.0650275
46	W5.CA	S.CA	-0.0650894
47	W5.CL	W5.NC	-0.06545
48	S.CL	S.CA	-0.0662283
49	W5.CL	W5.DF	-0.0677114
50	W5.CL	W5.CA	-0.0697317
51	W5.CL	W5.PR	-0.0708804
52	S.DF	W5.PR	-0.071911
53	W5.CL	S.PR	-0.0723838
54	S.NC	W5.PR	-0.0747727
55	S.CA	W5.PR	-0.0768113
56	S.CB	W5.PR	-0.0768821
57	W5.DF	S.CR	-0.0769805
58	S.CL	W5.DF	-0.0804949
59	W5.NC	S.CR	-0.082862
60	W5.DF	W5.CA	-0.0846052
61	W5.NC	W5.CA	-0.0869093
62	S.CL	W5.CA	-0.0871059
63	W5.DF	W5.CB	-0.0871849
64	S.CL	W5.CB	-0.0878555
65	S.CL	W5.NC	-0.0879052
66	W5.CA	W5.PR	-0.0882619
67	W5.NC	W5.PR	-0.0885982
68	S.CL	W5.PR	-0.089521
69	W5.PR	W5.CB	-0.08988
70	W5.DF	W5.NC	-0.0906928
71	W5.NC	W5.CB	-0.0907544
72	W5.DF	W5.PR	-0.0943725
73	S.CR	W5.PR	-0.0996067
74	W5.CA	S.CR	-0.119601
75	S.CR	W5.CB	-0.125679
76	S.CL	W5.CR	-0.129842
77	W5.CL	W5.CR	-0.129984
78	W5.CR	S.CR	-0.13921
79	W5.CR	W5.PR	-0.142895
80	S.CA	S.CR	-0.146055
81	S.CR	S.CB	-0.14733
82	W5.CL	S.CL	-0.150653
83	W5.DF	W5.CR	-0.168889
84	W5.NC	W5.CR	-0.176501
85	S.PR	W5.CR	-0.182682
86	W5.CR	S.NC	-0.183939
87	S.DF	W5.CR	-0.186674
88	W5.CR	W5.CB	-0.194768
89	W5.CA	W5.CR	-0.195015
90	W5.CR	S.CA	-0.242874
91	W5.CR	S.CB	-0.243398

Table A.4: E1-Agent pairs ranked according to the Min Impact Score.

Rank	Agent i	Agent j	DI
1	S.CA	S.CB	0.59348
2	W5.CR	S.CR	0.281338
3	W5.CA	W5.CB	0.262688
4	S.CL	S.CR	0.0502188
5	S.DF	S.PR	0.00218301
6	S.DF	S.NC	-0.00397064
7	S.PR	S.NC	-0.00730324
8	S.CL	S.DF	-0.012755
9	S.CL	S.NC	-0.0238906
10	W5.DF	S.PR	-0.0394925
11	S.DF	W5.DF	-0.0442297
12	W5.CL	S.CR	-0.0449244
13	S.DF	W5.NC	-0.0454422
14	S.CL	S.PR	-0.0463691
15	W5.NC	S.PR	-0.0484037
16	S.PR	W5.PR	-0.0513328
17	W5.NC	S.NC	-0.0535115
18	W5.DF	S.NC	-0.0563818
19	W5.CL	S.DF	-0.0587852
20	W5.CL	S.NC	-0.0607848
21	S.CA	S.NC	-0.0674986
22	S.DF	W5.PR	-0.0692615
23	S.PR	W5.CA	-0.0695348
24	S.DF	W5.CA	-0.0717318
25	S.NC	S.CB	-0.0724677
26	S.PR	W5.CB	-0.0740849
27	S.NC	W5.PR	-0.0777367
28	S.DF	W5.CB	-0.0785399
29	S.NC	W5.CB	-0.0795029
30	S.PR	S.CA	-0.0807079
31	S.PR	S.CB	-0.0817253
32	S.DF	S.CA	-0.0826129
33	S.DF	S.CB	-0.084323
34	W5.CA	S.NC	-0.0863162
35	W5.CL	S.PR	-0.0917511
36	S.CL	W5.DF	-0.0966285
37	W5.DF	W5.NC	-0.100456
38	S.CL	S.CB	-0.105683
39	W5.NC	S.CA	-0.107744
40	W5.NC	S.CB	-0.110044
41	S.CL	S.CA	-0.111206
42	S.CL	W5.NC	-0.112374
43	W5.DF	S.CR	-0.113211
44	S.PR	S.CR	-0.113321
45	S.CR	W5.PR	-0.115925
46	W5.DF	S.CB	-0.117887
47	W5.NC	S.CR	-0.118177
48	S.DF	S.CR	-0.118551
49	W5.CL	S.CB	-0.121519
50	W5.NC	W5.PR	-0.121797
51	W5.CL	W5.DF	-0.123223
52	W5.CL	S.CA	-0.125209
53	W5.CL	W5.PR	-0.125447
54	W5.DF	S.CA	-0.127031
55	S.CL	W5.CR	-0.127402
56	W5.CL	W5.NC	-0.127594
57	S.CR	S.NC	-0.128331
58	S.CL	W5.PR	-0.128791
59	W5.DF	W5.PR	-0.132668
60	W5.DF	W5.CA	-0.132824
61	W5.CL	W5.CR	-0.135054
62	W5.NC	W5.CA	-0.135085
63	W5.NC	W5.CB	-0.137446
64	W5.DF	W5.CB	-0.13931
65	S.CL	W5.CA	-0.142501
66	W5.CA	S.CB	-0.145612
67	S.CB	W5.CB	-0.147006
68	S.CA	W5.CB	-0.14764
69	S.CL	W5.CB	-0.148206
70	S.CA	W5.PR	-0.150043
71	W5.PR	W5.CB	-0.15075
72	S.CB	W5.PR	-0.152372
73	W5.CA	S.CA	-0.153924
74	W5.CA	W5.PR	-0.155094
75	W5.CL	W5.CA	-0.170172
76	W5.CL	W5.CB	-0.172048
77	W5.CA	S.CR	-0.185203
78	S.CR	W5.CB	-0.186936
79	W5.CL	S.CL	-0.190995
80	W5.CR	W5.PR	-0.194465
81	S.PR	W5.CR	-0.198404
82	W5.CR	S.NC	-0.203004
83	S.DF	W5.CR	-0.204006
84	W5.DF	W5.CR	-0.210277
85	W5.NC	W5.CR	-0.214023
86	S.CA	S.CR	-0.23342
87	S.CR	S.CB	-0.252221
88	W5.CA	W5.CR	-0.255744
89	W5.CR	W5.CB	-0.26223
90	W5.CR	S.CA	-0.2993
91	W5.CR	S.CB	-0.300669

Table A.5: E1-Agent pairs ranked according to the Differential Total Impact Score.

Rank	Agent i	Agent j	MuI
1	S.CR	W5.CR	0.8205
2	W5.CR	W5.NC	0.177952
3	W5.CR	W5.DF	0.17314
4	W5.CA	W5.CR	0.158265
5	W5.CB	W5.CR	0.148728
6	W5.CR	S.DF	0.145593
7	W5.CR	S.NC	0.14041
8	S.PR	W5.PR	0.0898325
9	S.CA	S.CB	0.0864108
10	S.CB	W5.CR	0.0828874
11	W5.CR	W5.PR	0.0819924
12	S.DF	W5.PR	0.0799518
13	W5.NC	W5.PR	0.0775214
14	S.CA	W5.CR	0.0747294
15	S.NC	W5.PR	0.0720325
16	W5.DF	W5.PR	0.0711563
17	W5.CA	W5.CB	0.0684351
18	W5.DF	W5.NC	0.0651412
19	W5.CB	W5.PR	0.0565469
20	S.DF	W5.NC	0.0531545
21	W5.CA	W5.PR	0.0516538
22	W5.CB	W5.NC	0.0514842
23	S.DF	W5.DF	0.0514578
24	W5.DF	S.PR	0.0489385
25	S.NC	W5.NC	0.0480998
26	W5.CA	W5.NC	0.0475637
27	W5.CR	S.PR	0.0455969
28	W5.NC	S.PR	0.0442014
29	W5.DF	S.NC	0.0441112
30	S.CR	W5.DF	0.0409191
31	W5.CA	W5.DF	0.0390632
32	W5.CB	W5.DF	0.0375906
33	W5.CB	S.NC	0.0327668
34	S.CR	W5.NC	0.0299801
35	W5.CA	S.CR	0.0272946
36	W5.CA	S.NC	0.0258231
37	W5.CB	S.CR	0.0236602
38	W5.CA	S.DF	0.0200617
39	W5.CB	S.DF	0.0191034
40	S.PR	W5.CA	0.0177429
41	S.CB	W5.NC	0.0172996
42	S.PR	W5.CB	0.0171413
43	S.CA	W5.NC	0.00822049
44	S.CL	W5.DF	0.00709707
45	S.CB	W5.PR	0.00508397
46	S.CL	W5.NC	0.00439548
47	S.CA	W5.PR	0.00377856
48	S.CR	W5.PR	0.00339619
49	S.CL	W5.CA	0.00282149
50	S.CL	W5.CB	0.001137
51	W5.CA	S.CB	-0.0034543
52	S.CA	W5.CB	-0.00741888
53	S.CB	W5.CB	-0.00808813
54	W5.CA	S.CA	-0.00957701
55	W5.CL	S.CL	-0.0119513
56	S.CB	W5.DF	-0.0128572
57	S.DF	S.PR	-0.0149641
58	S.CL	S.DF	-0.0158269
59	S.CA	W5.DF	-0.0169635
60	S.CL	S.NC	-0.0176101
61	S.NC	S.PR	-0.0194311
62	W5.CL	W5.DF	-0.0196964
63	W5.CL	S.NC	-0.0201323
64	W5.CL	S.PR	-0.0205257
65	W5.CL	S.DF	-0.0222914
66	W5.CL	W5.NC	-0.0250387
67	S.CL	W5.CR	-0.0261453
68	S.CL	W5.PR	-0.0401749
69	W5.CL	W5.PR	-0.0404366
70	W5.CL	W5.CR	-0.0448513
71	W5.CL	W5.CB	-0.0448892
72	S.DF	S.NC	-0.0451959
73	W5.CL	W5.CA	-0.046016
74	S.CR	S.DF	-0.0547415
75	W5.CL	S.CB	-0.0613365
76	S.CR	S.NC	-0.0653532
77	W5.CL	S.CA	-0.0655395
78	S.CL	S.PR	-0.0677596
79	S.CL	S.CA	-0.0700813
80	S.CA	S.CR	-0.0738579
81	S.CL	S.CB	-0.0740398
82	S.CA	S.NC	-0.0778774
83	S.CB	S.NC	-0.0785555
84	S.CB	S.CR	-0.0869855
85	S.CA	S.PR	-0.0877551
86	S.CR	S.PR	-0.0885355
87	S.CB	S.PR	-0.0949973
88	S.CA	S.DF	-0.118132
89	S.CB	S.DF	-0.118176
90	S.CL	S.CR	-0.129872
91	W5.CL	S.CR	-0.168905

Table A.6: E1-Agent pairs ranked according to the Mutual Impact Score).

A.2 Experiment 2: Strong v.s. 2s Weak Agents

Rank	Agent i	Agent j	Money
1	S.CA	S.CB	119.9245833
2	S.DF	S.NC	83.48125
3	S.CA	S.DF	83.07791667
4	S.CB	S.DF	82.66416667
5	S.CA	S.NC	81.79208333
6	S.CB	S.NC	81.51625
7	S.DF	S.PR	75.52083333
8	S.NC	S.PR	75.03625
9	S.CB	S.PR	74.87333333
10	S.CA	S.PR	73.79083333
11	S.CL	S.NC	60.94833333
12	S.CL	S.DF	60.8625
13	S.CA	S.CL	57.37875
14	S.CB	S.CL	56.86333333
15	S.CL	S.PR	54.49875
16	S.CL	S.CR	50.11583333
17	S.CR	S.NC	42.21458333
18	S.CR	S.DF	41.6465
19	S.CR	S.PR	39.91570833
20	S.CA	S.CR	38.36758333
21	S.CB	S.CR	35.87870833
22	W2.PR	S.NC	9.155291667
23	S.CA	W2.NC	8.943416667
24	S.DF	W2.NC	8.631583333
25	W2.CB	S.NC	8.456083333
26	W2.CB	S.CA	7.609125
27	W2.DF	S.DF	7.2585
28	W2.CB	S.DF	7.155833333
29	W2.CA	S.CB	7.132833333
30	W2.CA	S.NC	7.121666667
31	W2.DF	S.NC	7.035
32	W2.NC	S.NC	7.026333333
33	W2.PR	S.CB	6.744375
34	S.CB	W2.NC	6.742791667
35	W2.PR	S.DF	6.700166667
36	W2.PR	S.CA	6.65075
37	W2.CA	S.DF	6.646916667
38	W2.CB	S.CB	6.491041667
39	S.CA	W2.DF	6.405583333
40	W2.CA	S.CA	6.301958333
41	S.CB	W2.DF	5.923583333
42	W2.PR	S.PR	4.667166667
43	W2.NC	S.PR	2.124666667
44	W2.CA	S.PR	1.240325
45	W2.DF	S.PR	1.142995833
46	W2.CB	S.PR	0.341670833
47	S.CA	W2.CL	-3.7270375
48	W2.CL	S.NC	-4.264833333
49	W2.CL	S.DF	-4.864166667
50	S.CB	W2.CL	-5.085583333
51	W2.CL	S.PR	-7.203
52	S.CL	W2.NC	-12.161375
53	W2.PR	S.CL	-12.368
54	S.CL	W2.DF	-12.43183333
55	W2.CB	S.CL	-13.6285
56	W2.CA	S.CL	-13.89041667
57	S.CR	W2.NC	-16.98483333
58	S.CR	W2.DF	-17.47508333
59	W2.PR	S.CR	-18.52083333
60	W2.CB	W2.CA	-19.03895833
61	W2.CB	S.CR	-21.689625
62	W2.CL	S.CL	-21.742375
63	W2.CA	S.CR	-22.625375
64	W2.PR	W2.CL	-24.96266667
65	W2.CR	S.PR	-30.24016667
66	W2.PR	W2.CA	-32.38191667
67	W2.PR	W2.NC	-32.39558333
68	W2.CR	S.DF	-32.888625
69	W2.PR	W2.DF	-32.90866667
70	W2.CR	S.NC	-33.10620833
71	W2.PR	W2.CB	-33.20579167
72	S.CA	W2.CR	-34.10908333
73	S.CB	W2.CR	-35.47104167
74	W2.CL	S.CR	-38.69241667
75	W2.DF	W2.NC	-39.5385
76	W2.CB	W2.DF	-40.419125
77	W2.CB	W2.NC	-40.57116667
78	W2.CA	W2.DF	-41.35133333
79	W2.CA	W2.NC	-41.8875
80	S.CL	W2.CR	-45.83
81	W2.CL	W2.NC	-47.715
82	W2.CL	W2.DF	-48.18916667
83	W2.CA	W2.CL	-48.41
84	W2.CB	W2.CL	-49.06916667
85	W2.CR	S.CR	-63.22791667
86	W2.CR	W2.DF	-68.47
87	W2.CR	W2.NC	-69.78083333
88	W2.CB	W2.CR	-69.97666667
89	W2.CL	W2.CR	-70.15625
90	W2.CA	W2.CR	-71.36291667
91	W2.PR	W2.CR	-79.99791667

Table A.7: E2-Agent pairs ranked according to the total money gained (mbb/g).

Rank	Agent i	Agent j	TI
1	S.CA	S.CB	1.13789
2	S.CL	S.CR	0.415116
3	W2.CR	S.CR	0.335429
4	W2.PR	W2.CL	0.17274
5	W2.CB	S.CR	0.116838
6	W2.CA	S.CR	0.11506
7	S.CR	W2.NC	0.0615105
8	S.CR	W2.DF	0.0489052
9	S.CR	S.PR	0.0101103
10	S.CL	S.PR	-0.00386081
11	S.CL	S.NC	-0.00430348
12	W2.PR	S.CR	-0.00497141
13	S.CA	S.CL	-0.00574327
14	S.CB	S.PR	-0.0079995
15	S.CL	S.DF	-0.00860491
16	S.CA	S.PR	-0.0115699
17	S.CB	S.CL	-0.0122188
18	S.CB	S.DF	-0.025552
19	S.DF	S.PR	-0.0258311
20	S.NC	S.PR	-0.026532
21	S.CA	S.DF	-0.0271918
22	S.CB	S.NC	-0.0364653
23	W2.PR	W2.NC	-0.03724
24	S.DF	S.NC	-0.0415185
25	S.CA	S.NC	-0.0427431
26	W2.PR	W2.DF	-0.0443041
27	W2.CL	S.CR	-0.06092
28	W2.PR	W2.CB	-0.0709103
29	W2.PR	W2.CA	-0.0785074
30	S.CR	S.NC	-0.0824857
31	S.CR	S.DF	-0.0831059
32	W2.CL	W2.DF	-0.0916694
33	W2.CL	W2.NC	-0.106833
34	S.CA	S.CR	-0.109627
35	S.CB	S.CR	-0.111831
36	W2.DF	W2.NC	-0.120481
37	W2.CB	W2.NC	-0.137771
38	W2.CA	W2.NC	-0.138803
39	W2.CA	W2.CL	-0.13971
40	W2.PR	S.CB	-0.14219
41	W2.PR	S.CA	-0.146719
42	W2.CB	W2.CL	-0.151129
43	W2.CA	W2.DF	-0.162551
44	W2.PR	S.NC	-0.162805
45	W2.CB	W2.DF	-0.168358
46	W2.PR	S.CL	-0.174309
47	W2.PR	S.PR	-0.174549
48	S.CL	W2.DF	-0.181368
49	W2.PR	S.DF	-0.18193
50	W2.CR	W2.NC	-0.183147
51	W2.CA	W2.CR	-0.183813
52	W2.CR	W2.DF	-0.189457
53	W2.CB	W2.CR	-0.190511
54	S.CA	W2.NC	-0.191532
55	S.CL	W2.NC	-0.191885
56	W2.CB	S.CB	-0.194981
57	W2.CB	S.CA	-0.196234
58	S.CB	W2.NC	-0.19749
59	W2.CA	S.CL	-0.198489
60	W2.CA	S.CA	-0.200965
61	W2.CA	S.CB	-0.202018
62	S.CA	W2.DF	-0.202088
63	W2.CL	S.CL	-0.204396
64	W2.CB	S.CL	-0.209616
65	W2.CB	W2.CA	-0.209993
66	S.CA	W2.CL	-0.214823
67	S.CB	W2.DF	-0.216394
68	S.CB	W2.CL	-0.217498
69	W2.NC	S.PR	-0.225511
70	S.DF	W2.NC	-0.22914
71	W2.DF	S.PR	-0.232071
72	W2.NC	S.NC	-0.232615
73	W2.CA	S.PR	-0.235737
74	W2.CB	S.DF	-0.236154
75	W2.CB	S.PR	-0.237098
76	W2.DF	S.NC	-0.237308
77	W2.DF	S.DF	-0.238279
78	W2.CB	S.NC	-0.242364
79	W2.CA	S.DF	-0.244706
80	W2.CA	S.NC	-0.245536
81	W2.CL	S.PR	-0.247188
82	W2.CL	S.NC	-0.252549
83	W2.CL	S.DF	-0.259712
84	W2.CL	W2.CR	-0.291146
85	W2.PR	W2.CR	-0.296062
86	S.CL	W2.CR	-0.33153
87	W2.CR	S.PR	-0.432061
88	S.CA	W2.CR	-0.46662
89	S.CB	W2.CR	-0.468154
90	W2.CR	S.NC	-0.497572
91	W2.CR	S.DF	-0.499057

Table A.8: E2-Agent pairs ranked according to the Total Impact Score.

Rank	Agent i	Agent j	MaI
1	W2.CR	S.CR	1.44774
2	S.CA	S.CB	1.16105
3	S.CL	S.CR	0.294364
4	W2.CA	S.CR	0.237159
5	W2.CB	S.CR	0.234338
6	S.CR	W2.NC	0.186118
7	W2.PR	S.NC	0.177526
8	W2.CL	S.DF	0.17574
9	S.CR	W2.DF	0.173203
10	W2.CL	S.NC	0.17293
11	W2.CR	S.NC	0.157832
12	W2.CR	S.DF	0.157477
13	W2.PR	S.DF	0.157206
14	W2.CB	S.NC	0.143451
15	W2.CB	S.DF	0.142371
16	W2.DF	S.NC	0.140003
17	W2.CA	S.DF	0.138915
18	W2.NC	S.NC	0.138124
19	W2.CA	S.NC	0.137902
20	S.DF	W2.NC	0.137902
21	S.CL	W2.CR	0.128606
22	W2.DF	S.DF	0.128345
23	S.CA	W2.CL	0.128026
24	W2.PR	S.CA	0.122675
25	W2.PR	S.CB	0.119151
26	S.CB	W2.CL	0.11602
27	W2.CL	S.PR	0.113557
28	S.CA	W2.CR	0.106443
29	W2.CB	S.CB	0.103871
30	S.CA	W2.DF	0.103496
31	W2.CB	S.CA	0.100685
32	W2.CA	S.CA	0.0981262
33	W2.PR	S.PR	0.0974976
34	W2.CA	S.CB	0.0951835
35	W2.CR	S.PR	0.093795
36	S.CB	W2.CR	0.0905163
37	W2.CL	S.CL	0.090342
38	S.CA	W2.NC	0.0875944
39	S.CB	W2.NC	0.0864466
40	W2.CB	S.CL	0.0804442
41	W2.DF	S.PR	0.0798659
42	S.CB	W2.DF	0.079394
43	W2.NC	S.PR	0.0789342
44	W2.CA	S.CL	0.0757377
45	W2.CB	S.PR	0.0743132
46	W2.CA	S.PR	0.0732166
47	S.CL	W2.DF	0.0646136
48	S.CL	W2.NC	0.0560313
49	W2.PR	S.CL	0.0111282
50	W2.CL	S.CR	-9.42E-05
51	S.CL	S.PR	-0.0162439
52	S.CL	S.DF	-0.0217318
53	S.CL	S.NC	-0.0297141
54	S.DF	S.PR	-0.040421
55	S.NC	S.PR	-0.0433342
56	S.CA	S.CL	-0.0766577
57	S.CB	S.CL	-0.0864668
58	S.DF	S.NC	-0.0917564
59	S.CB	S.PR	-0.0941364
60	S.CA	S.PR	-0.100404
61	W2.PR	S.CR	-0.109923
62	W2.CR	W2.DF	-0.111181
63	S.CB	S.NC	-0.118187
64	W2.CR	W2.NC	-0.118656
65	W2.CA	W2.CR	-0.128242
66	S.CA	S.NC	-0.129336
67	S.CR	S.PR	-0.134255
68	W2.CB	W2.CR	-0.143401
69	S.CA	S.DF	-0.14797
70	S.CB	S.DF	-0.149175
71	S.CR	S.NC	-0.174606
72	S.CR	S.DF	-0.175664
73	W2.PR	W2.CL	-0.186825
74	W2.CL	W2.DF	-0.222964
75	S.CB	S.CR	-0.232651
76	S.CA	S.CR	-0.234223
77	W2.CL	W2.NC	-0.252944
78	W2.CA	W2.CL	-0.260948
79	W2.CB	W2.CL	-0.268198
80	W2.DF	W2.NC	-0.295943
81	W2.CA	W2.NC	-0.323691
82	W2.CB	W2.DF	-0.329986
83	W2.CA	W2.DF	-0.332668
84	W2.CB	W2.NC	-0.337983
85	W2.CL	W2.CR	-0.35832
86	W2.PR	W2.CB	-0.399832
87	W2.PR	W2.CR	-0.413052
88	W2.PR	W2.NC	-0.416846
89	W2.PR	W2.CA	-0.421944
90	W2.CB	W2.CA	-0.430003
91	W2.PR	W2.DF	-0.434818

Table A.9: E2-Agent pairs ranked according to the Marginal Impact Score.

Rank	Agent i	Agent j	MiI
1	S.CA	S.CB	0.457624
2	S.CL	S.CR	0.11473
3	S.DF	S.NC	-0.0282926
4	W2.PR	W2.CL	-0.0345984
5	S.CB	S.DF	-0.0350939
6	S.DF	S.PR	-0.0355195
7	S.NC	S.PR	-0.0366603
8	S.CA	S.DF	-0.0389293
9	S.CB	S.NC	-0.0392255
10	S.CA	S.NC	-0.0394283
11	S.CA	S.PR	-0.044461
12	S.CL	S.PR	-0.0447222
13	S.CB	S.PR	-0.0453031
14	S.CL	S.DF	-0.0483226
15	S.CL	S.NC	-0.0501218
16	S.CA	S.CL	-0.0639879
17	S.CB	S.CL	-0.0649373
18	W2.PR	W2.NC	-0.0855723
19	W2.PR	W2.DF	-0.0920056
20	W2.PR	W2.CB	-0.0963644
21	S.CR	S.PR	-0.102053
22	W2.PR	W2.CA	-0.102564
23	W2.CR	S.CR	-0.112098
24	S.CR	S.NC	-0.114776
25	S.CR	S.DF	-0.11834
26	S.CR	W2.NC	-0.142262
27	S.CR	W2.DF	-0.14503
28	W2.CL	W2.DF	-0.160158
29	W2.CL	W2.NC	-0.160373
30	W2.CB	S.CR	-0.162595
31	W2.CA	S.CR	-0.164664
32	W2.CA	W2.CL	-0.176971
33	W2.CB	W2.NC	-0.192993
34	W2.DF	W2.NC	-0.195046
35	W2.CB	W2.CL	-0.197297
36	S.CB	S.CR	-0.198949
37	S.CA	S.CR	-0.199039
38	W2.CA	W2.DF	-0.199565
39	W2.CB	W2.DF	-0.20283
40	W2.CA	W2.NC	-0.203297
41	W2.PR	S.CB	-0.205945
42	W2.PR	S.CA	-0.213517
43	W2.PR	S.PR	-0.215025
44	W2.PR	S.CL	-0.216585
45	W2.PR	S.CR	-0.219509
46	W2.PR	S.NC	-0.223101
47	W2.CL	S.CR	-0.233194
48	W2.PR	S.DF	-0.236853
49	W2.CB	S.CA	-0.242854
50	W2.CB	S.CB	-0.244613
51	W2.CA	S.CA	-0.246436
52	W2.CA	S.CB	-0.247948
53	S.CA	W2.NC	-0.248076
54	S.CB	W2.NC	-0.250251
55	S.CL	W2.DF	-0.251583
56	S.CA	W2.DF	-0.254087
57	W2.NC	S.PR	-0.254115
58	W2.CA	S.CL	-0.254146
59	S.CL	W2.NC	-0.257257
60	S.CB	W2.CL	-0.257341
61	S.CA	W2.CL	-0.25949
62	W2.CB	S.CL	-0.262026
63	W2.DF	S.PR	-0.263105
64	W2.CL	S.CL	-0.26402
65	S.CB	W2.DF	-0.265018
66	W2.CA	S.PR	-0.265206
67	W2.CB	S.PR	-0.271078
68	W2.CA	S.NC	-0.274076
69	S.DF	W2.NC	-0.274946
70	W2.CB	S.NC	-0.276407
71	W2.DF	S.DF	-0.276419
72	W2.CB	S.DF	-0.276648
73	W2.NC	S.NC	-0.278451
74	W2.DF	S.NC	-0.27997
75	W2.CA	S.DF	-0.281203
76	W2.CL	S.NC	-0.286641
77	W2.CL	S.DF	-0.29082
78	W2.CL	S.PR	-0.308512
79	W2.CL	W2.CR	-0.361477
80	W2.CA	W2.CR	-0.362905
81	W2.CR	W2.NC	-0.369855
82	W2.CR	W2.DF	-0.370178
83	W2.CB	W2.CR	-0.372931
84	W2.CB	W2.CA	-0.402951
85	W2.PR	W2.CR	-0.452538
86	W2.CR	S.NC	-0.50848
87	W2.CR	S.DF	-0.511762
88	S.CA	W2.CR	-0.531765
89	S.CB	W2.CR	-0.532186
90	W2.CR	S.PR	-0.539223
91	S.CL	W2.CR	-0.580931

Table A.10: E2-Agent pairs ranked according to the Minimum Impact Score.

Rank	Agent i	Agent j	DI
1	S.CA	S.CB	0.747346
2	W2.CR	S.CR	0.239798
3	S.CL	S.CR	0.170694
4	S.CL	S.PR	0.138459
5	S.NC	S.PR	0.129413
6	S.DF	S.PR	0.127802
7	S.CL	S.DF	0.114495
8	S.CL	S.NC	0.114352
9	S.DF	S.NC	0.0835895
10	S.CA	S.PR	0.0581982
11	S.CB	S.PR	0.0505685
12	S.CA	S.CL	0.0364487
13	S.CB	S.CL	0.023174
14	S.CB	S.NC	0.0152373
15	S.CA	S.NC	0.0079695
16	S.CB	S.DF	0.003627
17	S.CA	S.DF	-0.000930967
18	S.CR	S.PR	-0.0409137
19	S.CR	S.DF	-0.064396
20	S.CR	S.NC	-0.064556
21	W2.CB	S.CR	-0.122357
22	S.CR	W2.NC	-0.124974
23	S.CR	W2.DF	-0.130104
24	W2.CA	S.CR	-0.131269
25	W2.PR	S.NC	-0.14029
26	S.DF	W2.NC	-0.145776
27	W2.DF	S.NC	-0.149715
28	W2.CB	S.DF	-0.150351
29	W2.CA	S.DF	-0.150617
30	W2.NC	S.NC	-0.151966
31	W2.CB	S.NC	-0.152511
32	W2.CA	S.NC	-0.155275
33	W2.PR	S.DF	-0.157452
34	W2.DF	S.DF	-0.15859
35	S.CA	S.CR	-0.181991
36	S.CL	W2.DF	-0.183389
37	S.CB	S.CR	-0.186806
38	W2.CA	S.CL	-0.19162
39	W2.NC	S.PR	-0.191956
40	W2.CB	S.CL	-0.193063
41	S.CL	W2.NC	-0.197001
42	W2.DF	S.PR	-0.198809
43	W2.CB	S.PR	-0.204557
44	W2.CA	S.PR	-0.205894
45	W2.CB	S.CB	-0.205898
46	W2.CL	S.NC	-0.212431
47	W2.CL	S.DF	-0.215066
48	W2.CB	S.CA	-0.217712
49	W2.CA	S.CA	-0.218714
50	W2.PR	S.PR	-0.218773
51	S.CB	W2.NC	-0.219723
52	W2.PR	S.CB	-0.220821
53	W2.CA	S.CB	-0.221247
54	S.CA	W2.NC	-0.2216
55	W2.PR	S.CA	-0.2251
56	S.CA	W2.DF	-0.225427
57	S.CB	W2.DF	-0.226274
58	W2.PR	S.CL	-0.231406
59	W2.CL	S.CL	-0.251652
60	W2.CL	S.PR	-0.263224
61	W2.CL	S.CR	-0.274622
62	S.CB	W2.CL	-0.28673
63	S.CA	W2.CL	-0.287367
64	W2.PR	S.CR	-0.32541
65	W2.PR	W2.CL	-0.352166
66	W2.DF	W2.NC	-0.386564
67	W2.CA	W2.NC	-0.410786
68	W2.CB	W2.DF	-0.416542
69	W2.CA	W2.DF	-0.416961
70	W2.PR	W2.CB	-0.417398
71	W2.CR	S.PR	-0.419944
72	W2.CB	W2.CA	-0.420391
73	W2.CL	W2.DF	-0.424924
74	W2.PR	W2.CA	-0.42502
75	W2.CB	W2.NC	-0.427349
76	W2.PR	W2.NC	-0.433636
77	W2.CL	W2.NC	-0.435917
78	W2.PR	W2.DF	-0.445402
79	W2.CA	W2.CL	-0.447642
80	W2.CB	W2.CL	-0.457611
81	W2.CR	S.NC	-0.460484
82	W2.CR	S.DF	-0.470277
83	S.CL	W2.CR	-0.48808
84	W2.CR	W2.NC	-0.497397
85	W2.CR	W2.DF	-0.502807
86	W2.CA	W2.CR	-0.53755
87	W2.CB	W2.CR	-0.541369
88	S.CA	W2.CR	-0.54367
89	S.CB	W2.CR	-0.55371
90	W2.CL	W2.CR	-0.610066
91	W2.PR	W2.CR	-0.796904

Table A.11: E2-Agent pairs ranked according to the Differential Total Impact Score.

Rank	Agent i	Agent j	MuI
1	W2.CR	S.CR	1.1123
2	W2.CR	S.DF	0.656533
3	W2.CR	S.NC	0.655403
4	S.CA	W2.CR	0.57306
5	S.CB	W2.CR	0.558672
6	W2.CR	S.PR	0.525854
7	S.CL	W2.CR	0.460135
8	W2.CL	S.DF	0.435451
9	W2.CL	S.NC	0.42548
10	W2.CB	S.NC	0.385815
11	W2.CA	S.DF	0.38362
12	W2.CA	S.NC	0.383439
13	W2.CB	S.DF	0.378524
14	W2.DF	S.NC	0.377312
15	W2.NC	S.NC	0.370739
16	S.DF	W2.NC	0.367043
17	W2.DF	S.DF	0.366624
18	W2.CL	S.PR	0.360743
19	S.CA	W2.CL	0.34285
20	W2.PR	S.NC	0.340332
21	W2.PR	S.DF	0.339136
22	S.CB	W2.CL	0.333516
23	W2.DF	S.PR	0.311935
24	W2.CB	S.PR	0.31141
25	W2.CA	S.PR	0.308956
26	S.CA	W2.DF	0.305581
27	W2.NC	S.PR	0.304444
28	W2.CA	S.CA	0.299093
29	W2.CB	S.CB	0.29885
30	W2.CA	S.CB	0.297202
31	W2.CB	S.CA	0.29692

Rank	Agent i	Agent jj	MuI
32	S.CB	W2.DF	0.295788
33	W2.CL	S.CL	0.294738
34	W2.CB	S.CL	0.290061
35	S.CB	W2.NC	0.283939
36	S.CA	W2.NC	0.279127
37	W2.CA	S.CL	0.274228
38	W2.PR	S.PR	0.272048
39	W2.PR	S.CA	0.269395
40	W2.PR	S.CB	0.261341
41	S.CL	W2.NC	0.247917
42	S.CL	W2.DF	0.245984
43	W2.PR	S.CL	0.185438
44	S.CR	W2.NC	0.12461
45	S.CR	W2.DF	0.124298
46	W2.CA	S.CR	0.122099
47	W2.CB	S.CR	0.117499
48	W2.CR	W2.DF	0.0782753
49	W2.CR	W2.NC	0.0644937
50	W2.CL	S.CR	0.0608265
51	W2.CA	W2.CR	0.0555691
52	W2.CB	W2.CR	0.0471094
53	S.CA	S.CB	0.0231502
54	S.CL	S.PR	-0.0123831
55	S.CL	S.DF	-0.0131267
56	S.DF	S.PR	-0.0145901
57	S.NC	S.PR	-0.0168023
58	S.CL	S.NC	-0.0254109
59	S.DF	S.NC	-0.0502381
60	W2.CL	W2.CR	-0.0671722
61	S.CA	S.CL	-0.0709139
62	S.CB	S.CL	-0.0742483

Rank	Agent i	Agent j	MuI
63	S.CB	S.NC	-0.0817202
64	S.CB	S.PR	-0.0861367
65	S.CA	S.NC	-0.0865928
66	S.CA	S.PR	-0.0888346
67	S.CR	S.NC	-0.0921198
68	S.CR	S.DF	-0.092558
69	W2.PR	S.CR	-0.104952
70	W2.PR	W2.CR	-0.116991
71	W2.CB	W2.CL	-0.11707
72	S.CL	S.CR	-0.120751
73	S.CA	S.DF	-0.120779
74	S.CB	S.CR	-0.120818
75	W2.CA	W2.CL	-0.121237
76	S.CB	S.DF	-0.123622
77	S.CA	S.CR	-0.124597
78	W2.CL	W2.DF	-0.131293
79	S.CR	S.PR	-0.144366
80	W2.CL	W2.NC	-0.14611
81	W2.CB	W2.DF	-0.161628
82	W2.CA	W2.DF	-0.170118
83	W2.DF	W2.NC	-0.175462
84	W2.CA	W2.NC	-0.184888
85	W2.CB	W2.NC	-0.200212
86	W2.CB	W2.CA	-0.220009
87	W2.PR	W2.CB	-0.328921
88	W2.PR	W2.CA	-0.343437
89	W2.PR	W2.CL	-0.359565
90	W2.PR	W2.NC	-0.379607
91	W2.PR	W2.DF	-0.390514

Table A.12: Agent pairs ranked according to the Mutual Impact Score.