Optimal Poker Strategies Under Collusion

Simulating Fictitious Play in Simplified Poker Variants

Robert J. Ullman

Advised by Professor Andrew Appel

Princeton University
Department of Computer Science
Date of Submission: April 30th, 2015

Abstract

We create modular software to construct and solve general hold'em poker games. Because most poker variants, like Texas Hold'em, are prohibitively complicated to analyze directly, we instead use the flexibility of our software to look at a variety of stripped-down hold'em-like games. In these simplified games, we answer questions about how game rules, player position, and betting history affect a player's strategy to give insight into optimal strategies for more complicated poker variants. We simulate these variants with collusion and find that collusion leads to an overall increase in aggression by colluding players. We find that although strong collusion leads to a significant advantage for colluding players, weak collusion does not.

1 Introduction

Unlike most casino games, poker is a game of skill. Talented players gain an advantage by approximating the likelihood of having the winning hand. Sometimes, players can be certain that they will win; other times, this approximation requires careful consideration of the entire betting history. Approximating win probability helps a player decide if she should bet, and if so how much, or fold. Yet, even assuming a player can perfectly compute the probability she will win a given hand — a calculation that can be done exactly with enough time — her optimal betting strategy remains unclear because her expected winnings depend on the strategy of her opponents. Given the uncertainty of opposing strategies, a Nash equilibrium strategy is favorable because it hedges one's bets against all possible opposing strategies.

Another way to gain an advantage is to cheat and gain private information as to what cards are left in the deck or what cards the other players have. In many instances, cheating only helps a single player (the cheater) at the expense of all other players. As such, cheating is disincentivized by all players (players are careful to hide their cards, join only reputable and fair games, etc.). Yet, cheating can be incentivized if some group of players can do better — either in aggregate or individually — by working together. Players who **strongly collude** share private information, like the cards they were dealt, for mutual gain. Strong collusion can be policed in in-person poker games by monitoring players to prevent sharing information, but such measures are ineffective in online poker games where sharing info outside the game is easy. As such, online poker communities rely primarily on reactive measures that use statistical analysis on past hands to detect collusion. This often works by looking for abnormal patterns in playing history but can also be done with more nuanced approaches that do not assume any particular behavior pattern [10]. Another form of collusion is **weak collusion** where players decide to work together without sharing information outside the rules of the game. Weak collusion strategies rely primarily on beneficial coordinated betting. Also, weak collusion is easy to implement and can be particularly effective against poor opponents. Strong collusion produces larger expected winnings than weak collusion, but weak collusion is harder to detect, even in in-person games, as it relies only on public information exchange.

In this paper, we characterize the Nash equilibrium strategy in two simple hold'em variants. In particular, we quantify the effects of game rules, number of players, player positions, and betting history on the equilibrium strategy. We also consider these variants with both strong and weak collusion to understand the effects of collusion in hold'em-like games. We explain the poker abstractions and software implementation that we created to facilitate studying hold'em-like poker variants.

2 Finding a Nash Equilibrium

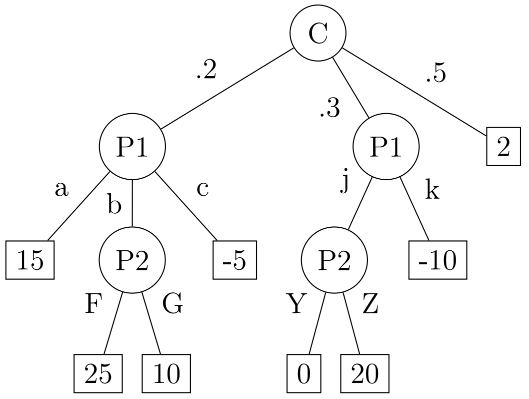
Game theory, a field pioneered by John Von Neumannn in the 1940s, studies strategic competitive decision making. A game is specified by offering a player a list of potential strategies and indicating the payoff for each strategy conditional on all the other player's strategies. Specifying a game in this way, called a game's **normal form**, lends itself easily to a matrix representation. For example, consider a two-player game where the first player has n strategies and the second player has m strategies. We can define the first player's payoffs by an $n \times m$ matrix A, where A_{ij} is her payoff when she plays strategy i and the second player plays the strategy j. Similarly, we can define an $m \times n$ matrix B to be the payoffs for player two. A game is **zero-sum** if the payoffs for all players sum to zero. In our example, the game would be zero sum if $A = -B^T$.

A game in normal form can be thought of as being played in a single turn where all players simultaneously choose their strategies and then consult the payoff matrix for their wins or losses. Some games are more intuitively expressed in a turn-based form in which players alternate making moves. This is known as the **extensive form** of a game and is often expressed as a tree. In such a tree, nonleaf nodes represent a player's turn, edges represent possible moves on a given turn, and leaves represent game payoffs. Many games (e.g. most card games) have situational variation caused by neither player (e.g. the cards you are dealt), which can be conveniently represented by assigning some nodes to a nature player with outgoing edges representing the different game variations that can happen at that point. One can always convert between the matrix normal form and tree extensive forms of a game, but the matrix form is less concise because the number of strategies is exponential in the number of tree nodes for each player (figure 1).

Players can choose to play any distribution over their given strategies to form a **mixed strategy**. The special case where a player chooses to play only one such strategy is called a **pure strategy**. Von Neumann proved that in two-player zero-sum games there exists a value V and mixed strategies

	(F,Y)	(F,Z)	(G,Y)	(G,Z)
(a,j)	5	11	5	11
(a,k)	1	1	1	1
(b,j)	6	12	3	9
(b,k)	3	3	0	0
(c,j)	0	6	0	6
(c,k)	-3	-3	-3	-3

(a) Normal Form



(b) Extensive Form

Figure 1: An example two-player game (P1 versus P2) in both normal form and extensive form. The node labeled C is played by the nature (chance) player with the edges leaving it labeled by the probability of each game variation. The payoffs in the normal form are the expected values of playing each pure strategy against each pure strategy of the opponent.

for both players such that V is the best payoff the first player can achieve if the second player fixes his strategy and $-V$ is the best payoff the second player can achieve if the first player fixes her strategy. In other words, there exist strategies for each player such that no player has any incentive to change their strategy. Von Neumann's result, called the **minimax theorem**, was later generalized by John Nash who proved that in any game there exists mixed strategies for all players such that no player has any incentive to change her strategy individually. A set of such mixed strategies is called a **Nash equilibirum**.

2.1 Linear Programming

Finding a Nash equilibrium in games is often done with linear programs (LPs). For example, take a two-player game with payoff matrix A. Given strategy distribution vectors x and y for players 1 and 2 respectively then the payoff for player 1 is $x^T Ay$. As such, for a fixed strategy y the best response strategy x is the solution to the linear program

$$\mathbf{max}_x \quad x^T Ay \qquad \mathbf{subject\ to} \quad \vec{1} \cdot x_i = 1, \quad x \geq 0.$$

The analogous LP with payoff matrix $B = -A^T$ produces the second player's best response strategy. Combining the constraints of both LPs yields an LP that finds Nash equilibrium strategies. This can be generalized to a linear complementary problem (LCP) for nonzero-sum games [9]. Although linear programs can be solved in polynomial time in the size of the matrix, the size of the payoff matrix grows quickly with game complexity. The number of entries in the payoff matrix for the game is the product of the number of pure strategies for each player. The number of pure strategies for each player is the product of the number of actions they have at each node in the tree of the game's extensive form — a number exponential in the number of nodes for each player. This exponential blowup in matrix size is in some part due to information redundancy introduced in representing a game in normal form. Koller et al. showed that this redundancy can be eliminated in two-player games with perfect recall [9]. At a high level, this is accomplished by representing strategies not as a distribution over pure strategies, but instead as a distribution over game tree states. Representing strategies in this way reduces the payoff matrix to a sparse matrix with dimensions the number of decisions for each player (the same space as the game tree itself).

Linear programs are favored in tractable use cases because the LP produces the exact equilibrium. Yet, the construction of the LP is tricky to generalize from two-player to multi-player games. In particular, how to eliminate the exponential space blowup of a game's normal form is unknown in the general case. Furthermore, in most applied use cases an approximate Nash equilibrium is sufficient. As such, iterative algorithms, like **gradient descent** and **fictitious play** [3], allow for solutions with adjustable precision that run much faster than the LP for the same result. We have chosen to implement the simplest of these iterative algorithms, fictitious play.

2.2 Fictitious Play

Smoothed fictitious play is an iterative algorithm to approximate a Nash equilibrium. Let s_i^t be the strategy for player i at time t and let s_{-i}^t be the strategies for all players besides i at time t. Then, smoothed fictitious play uses the update rule

$$s_i^t = \left(1 - \frac{1}{t}\right) s_i^{t-1} + \frac{1}{t} r_i^{t-1},$$

where r_i^t is the best response to s_{-i}^t. The fictitious play algorithm terminates when the maximum regret of any player at time t is below some specified threshold. The regret of a player i at time t is the difference between the expected payoff of playing s_i^t and of playing r_i^t, fixing all other strategies at that time. In other words, this difference quantifies the regret of player i of stopping fictitious play at time t and not playing her actual best response to everyone else's strategies. If all players have regret less than some value ϵ, then the strategy profile is called an ϵ-equilibrium. Fictitious play does not always converge — in particular, algorithm 1 does not always terminate for small ϵ — but when it does it does so to a Nash equilibrium [4]. The intuition here is that if there exists some t such that s_p^t equals r_p^t for all p, then no players will ever be motivated to change their strategy.

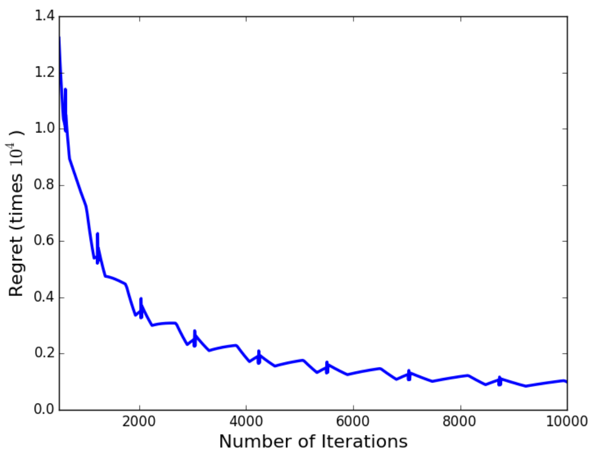
Algorithm 1 FictitiousPlay(ϵ)

```
regret, $t \leftarrow 0$
$s^0 \leftarrow$ initStrategy()
**do**
    $t \leftarrow t + 1$
    **for each** player $p$ **do**
        $r \leftarrow$ bestResponse($s_{-p}^{t-1}$)
        $s_p^t \leftarrow \left(1 - \frac{1}{t}\right) s_p^{t-1} + \frac{1}{t} r$
        $regret \leftarrow \max\left(regret, \text{payoff}(r) - \text{payoff}(s_p^t)\right)$
    **end for**
**while** $regret \geq \epsilon$
```

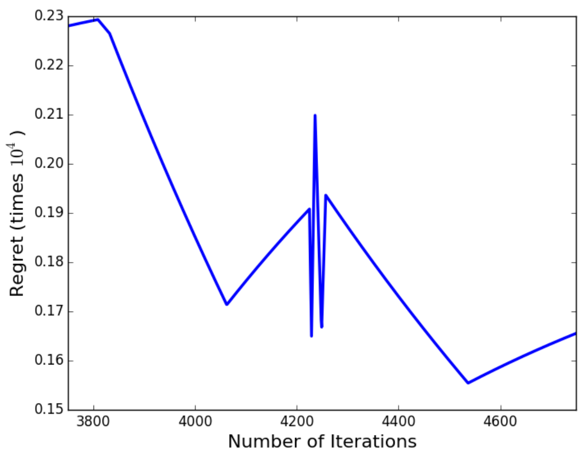
Because smoothed fictitious play takes a running average, it is resistant to change as the number of iterations increases. As such, even when it eventually converges, fictitious play tends to cycle around an equilibrium strategy (figure 2). Intuitively this is because a change in some dimension of the best response is dampened in its effect on the current equilibrium approximation. When fictitious play converges, the rate of convergence can be as slow as exponential in the order of magnitude of the regret bound.

2.3 Best Response Algorithm

In most papers that make use of a best response function, the function is used as an oracle because the particular implementation is unimportant. As such, the existing literature often falls short in giving a detailed explanation of any particular best response function. Although we find straightforward LP and LCP formulations of best response functions in the literature, these constructions are only applicable to two-player games [9]. We describe a simple procedure to compute a best response in any extensive form game with perfect recall. Our contribution here is not to present a particularly clever or optimized best response function, but instead to describe the straightforward



(a) Big Picture



(b) Single Cycle (Zoomed In)

Figure 2: The player regret in Mercer Hold'em at each iteration of fictitious play. Notice the dampened cyclic behavior.

and widely-applicable approach we implement for this research.

For a given player, the best response to a set of fixed strategies for all other players is always to select the action that maximizes that player's expected payoff. For this reason, the best response strategy is always a pure strategy. We define an information set I for a player p' to be a set of nodes that are indistinguishable from the perspective of p'.[1] For notational convenience, we define an information set function $I(n)$ to be the information set of a tree node n. Take any information set I for player p' and consider the set of available actions A for all $n \in I$. For action $a \in A$, we define $\mathbf{P}(a)$ to be the probability that the current player chooses a. For node $n \in I$, we define $\mathbf{F}(n)$ to be the frequency with which n occurs in the game. Finally, we define a child function $\mathbf{C}(n, a)$ to be the node in the game tree found by starting at n and taking action a. For an information set I, action a available at I, and payoff function V_p, we define the weighted value of taking action a:

$$\mathbf{W_I}(a) \equiv \sum_{n' \in I} \mathbf{F}(\mathbf{C}(n', a)) \times \mathbf{V}_p(\mathbf{C}(n', a)).$$

[1]Nodes are indistinguishable when they have the same history except for private information (e.g. hold cards for other players).

We recursively define the payoff function for player p at a node $n \in I$:

$$\mathbf{V}_p(n) = \begin{cases} \sum\limits_{a \in A} \mathbf{P}(a)\mathbf{W}_I(a) & : p \neq p' \\ \mathbf{max}_{a \in A}\, \mathbf{W}_I(a) & : p = p' \end{cases}$$

The best response strategy s for player p defined on any of her information sets I is the pure strategy

$$s(I) \equiv \mathbf{maxarg}_{a \in A}\, \mathbf{W}_I(a).$$

The best response strategy and the payoffs at each node can be calculated together, where the maximizing action used in calculating the payoff for player p is also the action selected as player p's pure strategy.

The frequency and payoff functions $\mathbf{F}$ and $\mathbf{V}$ respectively are fundamentally different in nature. The frequency function for a node n is computed by taking the product of the probabilities of selecting each action along the path from the root to n, while the payoff function for a node n is computed by working backwards from all the leaves in the subtree rooted at n. An effective way to visualize (and implement) finding the best response is to propagate frequency information from the root towards the leaves and propagate payoff information backwards from the leaves towards the root until all nodes are explored. Information sets pose a potential problem because computing the payoff at a given node involves jumping around the tree and consulting the payoff of all nodes that share the same information set. As such, how to propagate frequency/payoff information along the game tree and how to deal with the potential of infinite loops when computing node payoffs is often unclear. To avoid the confusion with propagating frequencies and payoffs, these values are not calculated incrementally but instead are calculated on the fly with caching (**dynamic programming**). The problem with infinite loops is avoided entirely because poker (or at least all variants we study) is a game of **perfect recall**: for two tree nodes n, n', where n is an ancestor of n', we have that no node $v' \in I(n')$ is an ancestor of any $v \in I(n)$. In other words, every player can recall all past events in the game when making a decision.

3 Game Representation

3.1 Poker

Poker serves as a classic example of a zero-sum game. There are many variants, but all follow the same basic premise: each player has some hole (private) cards and participates in one or more betting rounds as shared cards and/or more hole cards are dealt. In the end, the player with the best hand — as determined by the rules of the poker variant — wins the accumulated bets. In cash poker, people bet money (or chip equivalents) and thus aim to maximize expected winnings. In tournament poker, a variant that has become popular in televised poker, players bet chips that have no cash equivalent. Instead, people are awarded cash prizes based on when in the tournament they lose all their chips. The key difference between these two types of poker is that cash poker has a local win condition to win as many hands as possible, while tournament poker has a global win condition to end up with all the chips. As such, playing tournament poker requires taking into account all players' chip stacks in addition to the traditional concerns of cash poker (hole cards, shared cards, betting history, etc.).

In tournament poker, because the win condition involves bankrupting opposing players, aggressive bets that force players **all in** (to call with all their chips) are more common. In fact, the **fold/shove model**, a simplified model for tournament poker where players decide to fold or go all in as their first bet, is actually a reasonably good approximation of a Nash equilibrium. Although this model greatly reduces the complexity of the game, how to value chip stacks — the likelihood a player will win given all players' chip stacks — is still unclear. The poker community often uses the Independent Chip Model (ICM) heuristic to approximate chip stack cash value. Research has been done to improve upon ICM using a double iterative approach where Nash equilibrium strategies computed using ICM are used to update chip stack valuations [5].

In essence, tournament poker is a more complex version of cash poker where the expected value of the game comes not only from the strategy in any single hand, but also from the chip stacks of all players. This complexity does not necessarily result in more complex strategies, but instead frequently leads to a simpler strategy as seen with the fold/shove model. We restrict our study to cash hold'em variants.

3.2 The Abstractions

We represent poker games as a game tree in extensive form. For this purpose, we developed a set of abstractions for specifying poker games that hide the tedious underlying tree construction and wrote a program implementing these abstractions that can be configured to a wide variety of poker-like games. The abstractions have three main pieces: a deck, a list of game rounds, and a payoff evaluator. The deck abstraction specifies the ranks and suits available in the game. Each game round has three phases: anteing, dealing, and betting. The anteing phase requires paying the ante, a forced bet for each player. Antes provide incentive for players not to fold during the upcoming round because doing so would forfeit their ante. Although such an incentive is usually only required for the first round because in later rounds remaining players have already had (unforced) bets, our particular abstraction allows antes at every round. In some variants of poker, **blinds** — forced bets for some subset of the players that contribute towards the betting phase — are used as alternatives to antes. We chose to study games without blinds, so they are not currently implemented in the abstraction but could be easily added. Each dealing phase gives each player zero or more additional hole cards and gives all players zero or more additional shared cards. Each betting phase is specified by a list of allowed bets and maximum number of bets per person. Players take turns selecting one of the allowed bets or folding until either all but one player has folded or everyone has called/checked (bet no more than the previous person). The list of allowed bets is not a perfect model for no-limit games where players can conceivably bet anything, but in this case it can serve as an approximation.

The payoff evaluator specifies how to distribute the pot to each player at terminal nodes. This incorporates the winning condition of the poker variant and usually involves determining the best possible hand(s) out of the shared and hole cards and rewarding those players (who have not already folded) with the best hand(s). In Texas Hold'em, there are 6 types of hands[2], and players make the best five-card hand of five shared cards and two hole cards. Most poker variants involve making five-card hands, so implementing one five-card evaluator is usually sufficient. For our research, we make use of the flexibility of our abstraction to play poker-like games with a four-card evaluator. In addition, one can use the evaluator abstraction to explore game variants with untraditional win conditions. For example, our evaluator abstraction expresses weak collusion by sharing colluding

[2]The Texas Hold'em hand types are high card, one pair, two pair, three of a kind, straight, flush, full house, four of a kind, and straight flush. Ties are broken within each hand type by card rank.

players' payoffs.

3.3 The Construction

The game tree is built with three different types of nodes: action nodes whose outgoing edges represent different player betting choices, deal nodes whose outgoing edges represent different assignments of hole or shared cards, and payoff nodes that are leaves and store the payoff results for each player. This specification leads to a straightforward algorithm to build a game tree where for each round one adds a series of nodes that correspond to all possible deals with all possible combinations of player bets. At the end of the game — either caused by all but one player folding or finishing all rounds of betting — the payoff evaluator is used to generate the payoff nodes.

From any node in the game tree, one can follow the path back to the tree root to determine the entire game state. This game state includes all public information, like shared cards and bets, and private information, like hole cards. As such, only an omniscient player knows her location in the game tree in a game with private information. Players of the game must play the same strategy (the same distribution over their possible actions) at each action node where the game history is indistinguishable to them. To enforce this constraint, each action node is part of a set, called its **information set**, that contains all other nodes indistinguishable for this node from the given player's perspective.

In general, generating the game tree from our abstractions does not tie us to any particular underlying implementation and allows us to perform many general optimizations to increase fictitious play performance. Precomputing the game tree before iterating fictitious play allows the tree and information sets to be cached, which simplifies and speeds up calculating the best response. In addition, terminal payoffs, which do not change between iterations of fictitious play, are cached, which leads to big performance gains because the payoff evaluator need not be run more than once for each game. Furthermore, the game tree can often be simplified after initial construction to save space and time. Simple, but effective, optimizations involve skipping deal and action nodes for folded players and truncating the game tree at nodes where all but one player has folded. Because the dealing and betting phases of each round consist of many deal and action nodes each, a more involved optimization involves compressing this information into fewer nodes. These types of

Example Poker Game Tree Construction

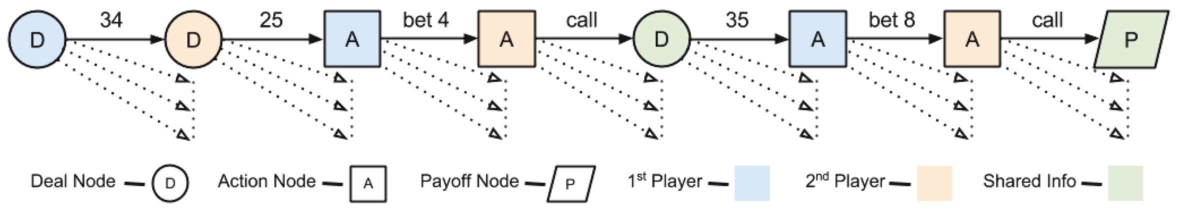


Figure 3: A visualization of a game tree our program can create. Because the trees branch rapidly, only one full path is shown. In the path shown, the following events happen in order: the first player is dealt a 3 and 4, the second player is dealt a 2 and 5, the first player bets 4 chips pre-flop, the second player calls, there are a 3 and 5 on the flop, the first player bets 8 chips, the second player calls, and then the game ends. In this case, the second player would win the accumulated pot (24 chips plus antes) because she has the best 4-card hand.

compression optimizations offer the most potential gains because, if applied early in the tree, they significantly reduce the tree size.

3.4 The Program

We wrote a Java program for constructing and solving hold'em-like poker games (`https://github.com/rjullman/poker-nasheq`). The `PokerGame` abstraction defines the hold'em variant and is defined by a `Deck`, an `Evaluator`, and a specified number of players. The `Deck` defines the set of cards used in the game, and the `Evaluator` determines the share of the pot for each player given her hole cards, shared cards, and betting history. For our research, we implemented a `FourCardEvaluator` and `CollusionEvaluator` that serve as an example of how to implement a custom `Evaluator`. The `PokerGame.buildGameTree()` method constructs the `GameTree` object used by the `Strategies.nes(GameTree gt, double tolerance)` utility method to compute a Nash equilibrium strategy using fictitious play. The result of a fictitious play calculation is a `Strategy` object indexed by `InfoSets`, the information set abstraction. Each `InfoSet` is programmatically generated by `buildGameTree()` and assigned a human-readable label indicating the current game history of the information set it represents.

The resulting game tree can be quite large and, in the current implementation, must fit in memory. Three-player Multiround Mercer Hold'em, the stripped-down poker variant that we consider in this paper, has around 5 million nodes and took over 4 gigabytes of memory to store. In general, the game trees generated (figure 3) grow exponentially with the complexity of the game. For example, adding a check bet to Multiround Mercer Hold'em made the resulting game tree overflow the (self-imposed) 32-gigabyte memory cap on our shared-resource super computer.[3] On that same super computer, finding a 0.01-equilibrium in Multiround Mercer Hold'em took roughly 10 hours and 3600 iterations of fictitious play. Under collusion, the equilibrium strategy took longer to converge, requiring up to roughly 24 hours of computation (and 8500 iterations of fictitious play) to reach a 0.01-equilibrium.

4 Hold'em Variants

4.1 Texas Hold'em (TH)

Texas Hold'em is a particular variant of poker played at many casinos. Representing two-player Texas Hold'em in its extensive form requires $O(10^{18})$ nodes or $O(10^{17})$ nodes leveraging suit and rank symmetries in the deck [2]. Since computing a Nash equilibrium for a game of this size is intractable, the general technique for solving Texas Hold'em (and other complex poker variants) is to find the Nash equilibrium for a simplified poker abstraction and then scale up the abstracted solution to formulate a strategy for the original game.

The primary source of game complexity comes from the combinatorial blowup of the number of possible hands in Texas Hold'em. To reduce this complexity, people often try to group hands of relatively equal situational value and handle them together. Gilpin and Sandholm designed an algorithm, *GameShrink*, to simplify extensive form games that they then applied to Texas Hold'em [6,7]. Whelan used a supervised machine learning approach to cluster hands of similar value [12]. Game complexity can be furthered reduced by discretizing the traditional betting model or limiting the number of rounds of betting [2, 8].

In 2015, Bowling et al. weakly solved two-player (unsimplified) Texas Hold'em by approximating the Nash equilibrium with an iterative algorithm [11]. They improved upon a counterfactual regret minimization (CFR) algorithm [13] to mitigate intractability problems but still resorted to brute force to reach a solution. Their program ran on 4800 processor cores and required over 68 days and nearly 11 TB of disk space to complete.

[3]Computations were run on a Fujitsu RX200 S8 server with dual, eight-core 2.8GHz Intel Xeon E5 2680 v2 processors with 256GB RAM running the Springdale distribution of Linux.

Bet Percentages by Holding

	25	**34**	**35**
1st Player	66	65	100
2nd Player	0	0	97

Table 1: The betting and folding percentages for the nontrivial hands in two-player SMH.

4.2 Standard and Multiround Mercer Hold'em

We will study variants of a hold'em poker game called Mercer Hold'em [1]. The game is played with a 20 card deck with 4 identical copies (no suits) of each of 5 cards (2 through 6). Mercer Hold'em has two rounds. In the first round, the **private round**, there is a 1 chip ante, two hole cards, and betting. In the second round, the **public round**, two shared cards are revealed (**the flop**) and there is (optionally) more betting. Afterwards, the best four-card hand wins.[4] The variants of Mercer Hold'em are determined by the constraints on the betting in both rounds. The simplest variant, **Standard Mercer Hold'em (SMH)**, allows each player only to bet 4 chips or to fold in the private round with no additional betting in the public round.

We choose to study Mercer Hold'em because it has nontrivial Nash equilibrium strategies while also having a low computational complexity.[5] Making use of the expressiveness of our game tree abstraction and the relative simplicity of Mercer Hold'em, we are able to study Nash equilibrium strategies for a large variety of game variants. Noting how slight variations in game rules affect Mercer Hold'em strategies provides insight into how poker variants affect game play even in more complicated games like Texas Hold'em.

In two-player SMH (2P-SMH), most hands can easily be classified as either good hands — players always bet with 22, 26, 33, 36, 44, 45, 46, 55, 56, or 66 — or bad hands — players never bet with 23 or 24 — regardless of the betting history. Concisely, we can describe the good hands as the pairs and high cards and the bad hands as the low unpaired cards. The remaining three possible hands $25, 34$, and 35 have a nontrivial playing strategy at equilibrium (table 1). Notice that for these hands the first player bets aggressively (by bluffing), while the second player plays defensively. This makes sense because only the first player sends signals to the second player, so

[4]In increasing value order, the hands in Mercer Hold'em are high card, one pair, two pair, three of a kind, and four of a kind.

[5]The simplest variant two-player SMH has approximately 10^4 nodes in the game tree, but even the more complex variants considered have at most 10^9 nodes.

First Bet Percentages by Holding

	22	**23**	**24**	**25**	**26**	**34**	**35**	**36**	**45**
1st Player	98.1	0.2	0.2	12.4	27.1	0.3	66.5	100	100
2nd Player	46.2	0.2	0.3	0.5	1.0	0.5	0.9	34.6	46.2

Table 2: The equilibrium strategy for the first bet in 2P-MMH. Only the hole cards with nontrivial equilibrium strategies are given (the other hands mimic 2P-SMH).

the second player cannot bluff. Furthermore, we find that the game has an expected value of 0.1 to the second player (and -0.1 to the first player), which makes sense because the second player acts with strictly more information than the first player.

We also examine a slightly more complicated variant of Mercer Hold'em we call **Multiround Mercer Hold'em (MMH)**. Just as in SMH, MMH allows two actions from the players: fold or bet 4 chips. In both the private and public rounds, there are 2 rounds of betting. For example, in the two-player game there can be at most 8 bets for a total pot size of 34 chips (there are 2 ante chips) at the end of the game. Notice that a player in position i gets a chance to raise only if all the players before her fold (otherwise her bets are calling the previous player). The multiple bets per round and the multiple rounds of betting in MMH simulate the features of a more complex poker game.

Two-player MMH (2P-MMH) has an equilibrium strategy approximated by 2P-SMH. This makes sense because intuitively the information gained from the betting history is secondary to the information gained by a player's hole cards. The restricted betting in SMH makes SMH a good first order approximation to 2P-MMH and in general all Mercer Hold'em variants. For this reason, the places where 2P-MMH differs from 2P-SMH are of particular interest as these are places where the flop is significant. The first player's first bet, which is made with no betting history (in both SMH and MMH) has a different equilibrium strategy in MMH than in SMH (tables 1 and 2). In particular, the first player no longer always folds bad hands — namely 23 and 24 — but instead bluffs a small percentage of the time. This makes sense because these weak hands can become strong given particular flops (e.g. 22) and, because there is a round of betting after the flop, players are able to capitalize on initially weak hands by betting post-flop. For much the same reason, it is no longer favorable for the first player to bluff with hole cards 34; the continued betting would force them to over-commit to the bluff. If players are bluffing, driving opposing players out of the game

First Post-Flop Bet Percentages Given 45 Hole Cards

	22	**23**	**26**	**33**	**36**	**46**	**56**	**66**
1st Player	20.5	93.9	62.9	47.7	24.7	100	100	53.6
2nd Player	30.0	22.7	34.9	7.6	46.1	60.9	98.1	33.8

Table 3: The equilibrium strategy for the first bet in the public round in 2P-MMH given that the player has 45 hole cards. Only the flops with nontrivial equilibrium strategies are given. The other flops are always bet on at equilibrium.

as soon as possible (i.e. by excessive betting) is in their best interest, but in this case the betting limit to 4 chips prevents such strategies — the opponent is not dissuaded from calling.

Another distinguishing part of MMH is how the first player bets initially in the public round. For example, if the first player is holding 45, a hand that is always bet on initially, the hand is bet on much less aggressively post-flop (table 3). In particular, the flops that are bet on less aggressively when holding 45 are similar in that they do not produce a pair (or better) with 45. As such, these flops potentially provide victory to initially weaker opposing hands (flop cards that make initially weak hands strong are called **outs**). As expected, the exact opposite is true for initially weak hands like 34. These initially weak hands are sometimes bluffed on in the private round, but are bet on in the public round only if the flop produces almost certain victory. As a general rule — applied to all holdings — the first player should bet a large percentage of the time on flops that produce good hands with their holdings, while also frequently bluffing on hands that do not. In the latter case, the player continues to bet despite only having a mediocre hand because it is not worth forfeiting the accumulated bets so far.

Although the second player bets more defensively than the first player in general, as in 2P-SMH, the second player bets much less defensively in the public round of 2P-MMH (table 3). Additionally, the overall defensive play of the second player leads to some quirks in the equilibrium strategy. In particular, assuming a 45 holding and a 23 flop, the first player bets over 90% of the time. This is bizarre given that a 23 flop produces no good hands with 45 and that a 22 flop, a flop with seemingly fewer outs than a 23 flop, is bet on only around 20% of the time. Yet, we can justify this equilibrium strategy in retrospect. The second player is unlikely to continue playing any initially poor hand that would combine well with a 23 flop. The difference between the 23 and 22 flop is more subtle but hinges on the fact that the second player can be convinced to fold when holding a

2 only in the former case because triple 2's is a nearly unbeatable hand.

We find that there is an increased positional advantage in 2P-MMH compared to 2P-SMH. The expected value of the game is 0.4 to the second player (-0.4 to the first player), four times the expected payoff in 2P-SMH. Unlike in 2P-SMH, why the second player is expected to win over the first player is unclear. Although the second player gets to act with more information initially, the first player has an opportunity to respond to these actions with even more information. In MMH, it turns out that the exact balance of the game (e.g. ratio of ante to bets, number of players, betting options, etc.) makes it much better to play second, but there is no reason for this to be true in general or in other Mercer Hold'em variants.

4.3 Three-Player Variants

The added complexity of **three-player SMH (3P-SMH)** changes the structure of the equilibrium strategies. Overall, the inclusion of a third player causes people to bet more defensively because with everything else equal they are more likely to lose than in the two-player case. As in 2P-SMH, one still always folds on unpaired low cards, always bets on pairs, and nearly always bets on unpaired high cards. The difference in betting on unpaired high cards is that at equilibrium the third player folds these hands when the first two players bet. Basically, the increased competition is enough to drive out the third player. The equilibrium strategy profile is also decisive. In every situation players either bet or fold at least 90% of the time. That said, there are more situations — 15 in 3P-SMH, versus 3 in 2P-SMH — where the equilibrium strategy has some, albeit small, amount of bluffing. This increase in decisive play and small-probability bluffing is characteristic in games with more public information because players, in particular later positioned players, can more accurately predict each other's cards. The expected value of this game is $-0.29, 0.008, 0.282$ for the first, second, and third players respectively. Once again, the expected value of the game increases by position because later players still have strictly more information.

The game of **three-player MMH (3P-MMH)** admits strategies that are much harder to characterize. Still, $33, 44, 55, 56$, and 66 are strong hands and are bet on almost always. The rest of the traditionally good hands, namely $22, 26, 36, 45$, and 46, as well as most of the weaker hands, have more nuanced play strategies that depend significantly on betting history, player position, and

Second Player's First Bet Percentage

	22	**23**	**24**	**25**	**26**	**34**	**35**	**36**	**45**	**46**
1st Player Folds	84.7	1.9	2.4	4.0	100	4.0	62.6	100	100	100
1st Player Bets	37.4	1.3	1.9	3.0	5.6	3.0	5.1	12.6	12.1	53.0

Table 4: The equilibrium strategy for the second player's first bet in 3P-MMH. The hands not shown ($33, 44, 55, 56$, and 66) are always bet on.

betting round. We focus primarily on characterizing these three factors.

Consider how players bet when no one has folded before their turn. On their first bet, earlier players bet more aggressively on mid-value hands, while weaker hands are played equally if not less aggressively (table 5). The general decrease in aggression by position is expected because later players must have overall better hands to compete with betting players. For this reason, the increase in aggression for hands 25 and 36 are unexpected. This behavior can be partially explained by the fact that there is still one more betting opportunity for each player. As such, there is some bluffing incentive that increases with pot size. The holding 22 shows the most mysterious behavior: not only are the betting strategies for 22 nonmonotone by position, but the differences in strategies are significant. Because this hand has high variance — post-flop it is either almost certainly the best or almost certainly the worst — perhaps the opposing effects of competition and bluffing are magnified and result in the observed equilibrium strategies. Yet, this trend among the strategies could also indicate that there is some significant factor for which we have still not accounted.

The second player's play style is dramatically more aggressive if the first player folds than if the first player bets (table 4). This directly follows the intuition that with fewer players remaining in the game, weaker hands will tend to win the pot. When the first player folds, the game is approximated by 2P-MMH. The differences in strategies for the remaining two players comes from the fact that the hole cards for the first player, which probably form a weak hand, are no longer in the deck. This has the effect of making bluffing weaker hands (like $23, 24, 25$, and 34) and playing mid-value hands more defensively (like 22 and 35) more advantageous.

The third player playing slightly differently when the first player folds than when the second player folds is somewhat surprising. Yet, because the third player knows the equilibrium strategies for the other players, she can use their bets along with their betting positions to tailor a response. In the case of 3P-MMH, we have already discovered that the second player will bet on weaker hands

First Bet Percentage

	22	**23**	**24**	**25**	**26**	**34**	**35**	**36**	**45**	**46**
1st Player	76.1	1.3	1.9	3.0	5.1	3.0	4.6	36.3	33.6	100
2nd Player	37.4	1.3	1.9	3.0	5.6	3.0	5.1	12.6	12.1	53.0
3rd Player	100	1.3	1.9	3.5	6.2	3.0	6.2	11.6	11.6	24.5

Table 5: The equilibrium strategy for the first bet in 3P-MMH assuming no one has yet folded. The hands not shown $(33, 44, 55, 56,$ and $66)$ are always bet on.

if the first player has already folded. Knowing this, the third player is comfortable playing more aggressively when the first player folds than when the second player folds (table 6). In practice, when playing more complicated poker variants like Texas Hold'em, positional information is often ignored because it adds considerable complexity to a player's strategy but has only a small influence on the equilibrium strategy (in most cases).

The public round equilibrium betting in 3P-MMH is easy to characterize because post-flop players know the value of their hands and know exactly what potential holdings of their opponents to which they would lose. As a general rule, a player finds herself in one of three situations: she has only a high card, in which case she bets (bluffs) just a small percent of the time; she has the best (or nearly best) hand, in which case she bets (nearly) always; or neither of these situations apply, and she bluffs that she has the best hand between 10 and 20 percent of the time. There are only a few scattered situations where the equilibrium strategy does not follow this rule.

5 Collusion

Players collude to increase their total winnings. As such, we motivate collusion in our existing poker variants by placing some sharing constraint on the payoffs for each colluding player to align their payoff goals. We aim to observe how this affects the equilibrium strategies and payoffs for

Third Player's First Bet Percentage

	22	**23**	**24**	**25**	**26**	**34**	**35**	**36**	**45**	**46**
1st Player Folds	70.7	2.4	3.0	5.1	12.6	4.6	9.4	100	80.9	100
2nd Player Folds	20.7	2.4	3.0	4.6	9.9	4.6	8.3	16.9	16.9	76.6

Table 6: The equilibrium strategy for the third player's first bet in 3P-MMH. The hands not shown $(33, 44, 55, 56,$ and $66)$ are always bet on.

all players. To be clear, colluding players are not altruistic towards each other but simply want to maximize their individual winnings under this sharing constraint.

In our research we initially considered the **pot sharing constraint** where colluding players split their sum total share of the pot evenly. Applying this constraint to 3P-MMH, we find that colluding players (regardless of playing position) do much worse than if they had not colluded. In particular, if the second and third players (who, without collusion, both make positive expected value on the game) collude, they both make negative expected value on the game. At first glance this seems bizarre, but this is the result of pot sharing being a poor motivating constraint. Under the pot sharing constraint, it makes sense that the colluding players bet and fold together. Otherwise, the betting player contributes more to the pot, effectively donating his money to the other colluding player if he wins, while also accepting more of the risk of losing. Yet, the players are also not motivated to bet together because even if they can force the other player to fold, they only stand to win half as much as before. The colluding players can potentially earn winnings in some situations, but the expected value of the game is an overall loss for them. In particular, this is seen in 3P-SMH where if the second and third players are colluding then they have no chance to push the noncolluding player out. As such, when players in the second and third position collude, they have no hands worth playing and always choose to fold (forfeiting their ante). Although the problem is not as extreme when other players collude, the central problem with pot sharing is the same: colluding players are overly motivated to fold. A better collusion constraint — which we use when simulating games with collusion – is the **win sharing constraint** where colluding players split the pot evenly after reclaiming their contribution to the pot. This allows colluding players to bet separately because they accept no more betting risk than they do without collusion.

Strong Collusion

In strong collusion, the colluding players look at each others' hands. We first consider strong collusion in 3P-SMH. This game broadly follows the strategy of standard (no collusion) 3P-SMH except, as a general rule, the colluding players do not bet when their partner has a better hand than they do. This makes sense because both colluding players betting doubles their betting risk.

Although normally colluding players do not both bet on a given hand, there are a few notable exceptions to this characterization that highlight the effects of strong collusion. In 3P-SMH, the only time both colluding players bet on a hand is when they both have good hands with different win conditions on the flop. The betting from the colluding side serves to bully the noncolluding player into folding. Additionally, both colluding players remaining in the game serves to swing the odds of winning in their favor, especially when both have good hands. As such, in this case, putting more money in the pot for the same potential winnings is worth the risk for the colluding players because it steeply increases the chance they will win. Perhaps the best example of this is when the colluding players have hands that provide maximal coverage of the out cards — for example, hole cards 36 and 45. Although neither 36 nor 45 is the best possible holding, the flop will almost certainly make one or both of their hands good.

There are also places where neither colluding player bets despite one or both of them having hands they would have bet on without collusion. This happens when the colluding players are holding each other's out cards. As such, the flop is less likely to have the cards they need to win. This happens most often when both colluding players have the same exact holding — for example 34. Although 34 is a relatively good hand on its own, both colluding players fold knowing that getting a 3 or 4 on the flop is incredibly unlikely. Additionally, even if the flop has a 3 or 4, it may also produce a stronger hand for the noncolluding player (say if the second flop card is a 5 or 6). This outcome is minimized when the colluding players both have the higher ranked hands — namely $36, 44, 45, 46, 55, 56$, or 66. In these cases, the colluding players revert to having only one player bet.

Bluffing can be made more effective under strong collusion. In many cases where both colluding players have weak hands, one of the colluding players will bet anyway. Whereas such a bet would not be feasible without collusion, under collusion the betting player is able to overrepresented her hand knowing her partner will fold. Take for example the hand 24, one of the weakest hands in SMH. This holding is nearly always folded by the first colluding player except when the second colluding player holds 23 or 35. Of these two cases, the 23 case is a bluff (the 35 case, as we have already seen, ensures flop coverage). Although the other colluding player folds immediately, she is critically important to the bluff because she makes the game appear move competitive to the noncolluding player.

Strong collusion in 3P-MMH has many of the same general strategies as in 3P-SMH. The added rounds of betting in MMH allow colluding players to exploit their advantage more fully. For example, flop coverage betting is more widespread: the first colluding player will now frequently bet with a holding of 24 if her colluding partner is holding $35, 36$, or 56. Because MMH has higher stakes than SMH, betting on these types of hands is worthwhile because colluding players are nearly certain to win the final pot. In general, collusion strategies that take advantage of aligning hands become more effective in higher stakes scenarios.

Bluffing, on the other hand, works differently in MMH than in SMH. Although many of the same hands are bluffed, they are bluffed less frequently. For example, while colluding players holding 24 and 23 bluff around 80% of the time in SMH, they only bluff around 20% of the time in MMH. Because there are more rounds of betting in MMH, the noncolluding player is more willing to call earlier bets to gain more information from her opponents' betting. This works in the noncolluding player's favor because the colluding players are forced to commit mores chips to the bluff. The noncolluding player is in general more willing to stay in the game longer, so bluffing with bad cards does not pay off for the colluding players. The bluffing strategies in MMH are much more like bluffing strategies in more complicated poker variants than those in SMH — poor hands are bluffed, but with low frequency.

As one would expect, strong collusion increases the expected value of the game for the colluding players. This increase is tiny[6] in 3P-SMH because the game is too short for colluding players to have a big advantage, but in 3P-MMH the gains are significant. We find that the collusion and noncollusion Nash equilibria are different for all players. In particular, the noncolluding player has to select between her noncollusion strategy and collusion strategy often without knowing which is optimal in her current situation. Given players A, B, and C, there are three types of collusion: **Public**, where A and B are colluding and C responds optimally; **Private**, where A and B are colluding and C responds with the noncollusion equilibrium strategy; and **Suspicious**, where A and B are actually not colluding and C responds with the collusion equilibrium strategy. We find that even under public collusion the noncolluding player loses over 17.5% of an ante each game (table 7). The losses are even greater for the noncolluding player under private collusion.

[6]We found expected gains on the order of 10^{-4}.

Collusion Payoffs in 3P-MMH

	Strong Collusion			**Weak Collusion** ($\times 10^{-3}$)		
	Public	Private	Suspicious	Public	Private	Suspicious
Player A	-0.178	-0.195	-0.0316	-0.996	-0.748	-0.448
Player B	0.0889	0.0977	0.0127	0.498	0.374	0.325
Player C	0.0889	0.0977	0.0189	0.498	0.374	0.122

Table 7: The expected payoffs of players in 3P-MMH given that players A and B are potentially colluding. All payoffs have an error of 0.00577, which is notably orders of magnitude more than the expected payoffs under weak collusion. We assume that the players rotate dealer each round (from A to B to C, then back to A and so on).

Weak Collusion

In weak collusion, the colluding players share no private information. Due to the simplicity of 3P-SMH, there are only a few differences from the noncollusion equilibrium strategy. The only significant change in strategy for the first bet is that the second colluding player does not bet on 34 if the first colluding player also bets. In 3P-MMH, the first colluding player bets on $22, 26, 35, 36$, and 45 more frequently. On the other hand, the second colluding player bets on 22 more frequently regardless of the first colluding player's action and bets on 45 and 46 less frequently but only if the first colluding player bets. As such, we find that weak collusion in MMH produces all around more aggressive play by the colluding players except that the second colluding player plays less aggressively on mid-value hands.

Under weak collusion, colluding players attempt a strategy that allows them to work together without explicitly sharing any private information. We observe the **extremes principle**: colluding players bluff frequently on low-value and high-value hands but bet less frequently on mid-value hands. Generally, for colluding players, betting helps to drive out the noncolluding player, while folding reduces the shared risk by decreasing the colluding players' aggregate contribution to the pot. SMH and MMH innately limit betting risk by having only one betting option, so betting in these variants is overall more aggressive relative to variants with open betting.

To better understand the intuition behind the extremes principle, first consider generally how to bet with each type of hand outside the scope of SMH and MMH. Given a weak hand, players either want to fold and accept the loss of their ante or to try to reclaim the pot by bluffing and betting high. Given a good hand, players should either bet high to drive players out (if they think the hand will depreciate in value) or bet low to encourage players to call (if they think the hand will appreciate in value). Betting with mid-value hands is more complicated but intuitively involves a balance bet aimed to goad low-value hands to call but mitigate losses against high-value hands. Under weak collusion, mid-value hands are bet on less frequently because it is hard for colluding players to assess the joint value of their holdings. In particular, as already seen in strong collusion, two mid-value hands may depreciate both hands' value by eliminating possible out cards in the flop. That said, Mercer Hold'em is played with a small deck and only four-card hands, so most games are won with a one or two pair. As such, MH tends to polarize hands and leave only a few hands to be considered potentially mid-value. If any hands were to be considered mid-value, they would be $26, 34, 35$, and 36 — those being the statistically weakest of the strong hands — but these hands are still highly favored. In the end, whether these weaker high-value hands are bet on more or less frequently because of collusion is influenced by other factors (e.g. MH variant and player position). Finally, because our variants of MH polarize the betting — players either bet 4 chips or fold — generally high-value hands are bet on frequently, while low-value hands are bluffed on somewhat infrequently. Collusion has the effect of making bluffing better — a colluding player bluffs with less risk — so it makes sense that colluding players want to bluff more than if they had not colluded.

Because we are computing equilibrium strategies, we have assumed public collusion[7]. As such, we can observe how the equilibrium strategy for the noncolluding player changes under collusion. In 3P-SMH we find no significant changes[8] to the noncolluding player's strategy. This makes sense because the equilibrium in SMH is polarizing — players fold absolutely or bet absolutely in most cases — and as such it seems reasonable that collusion is unable to shift the equilibrium. On the other hand, in 3P-MMH we expect, and observe, changes in strategy with collusion (table 8). We observe a slight, but significant, increase in aggression from the noncolluding player. This is probably a result of the colluding player trying to adjust her strategy against the increase in bluffing from the colluding players.[9]

Although weak collusion is at least as good as not colluding, whether it is actually an improvement is unclear. We found no significance difference in collusion payoffs because the payoffs are

[7]In this case of our fictitious play simulations, the noncolluding player determines collusion by observing her opponents' strategies over time and responding to them.

[8]There are changes, but they are within the margin of error of our fictitious play calculation.

[9]Reminiscent of the mantra "the best defense is a good offense."

Third Player's First Bet Percentages by Holding

		22	23	24	25	26	34	35	36	45	46
No Col	No Folding	100	1.3	1.9	3.5	6.2	3.0	6.2	11.6	11.6	24.5
	1st Player Folds	70.7	2.4	3.0	5.1	12.6	4.6	9.4	100	80.9	100
	2nd Player Folds	20.7	2.4	3.0	4.6	9.9	4.6	8.3	16.9	16.9	76.6
Col	No Folding	100	2.0	2.0	3.7	6.5	3.1	6.5	12.7	12.7	36.4
	1st Player Folds	30.2	2.5	3.1	5.9	13.8	5.4	9.9	100	68.6	100
	2nd Player Folds	46.6	2.5	3.1	5.4	12.1	4.8	9.3	28.5	41.5	100

Table 8: The equilibrium strategy for the third player's first bet in 3P-MMH with and without weak collusion by the first two players. Notice that the third player tends to bet more aggressively under collusion except for on the hands 22 and 45. Also note that the hole card holdings not shown in the table are always bet on.

within the margin of error of our fictitious play calculation (table 7). These results are not that surprising because poker is, for lack of a better word, *strongly* zero-sum: players that decide to collude are still competing with each other over the pot. As such, under public weak collusion, the colluding players gain no private information — not even the fact that they are colluding — and as such it is reasonable to predict that they would be unable to curry an advantage. Yet, because we observed that under public collusion all players modify their strategy, namely there exists a different Nash equilibrium than in a noncollusion situation, the colluding players can do slightly better under private collusion because the noncolluding player is probably not playing her best response strategy.

6 Conclusion and Future Work

Although SMH and MMH are toy Hold'em variants, their trends in Nash equilibrium provide heuristics that extend to the equilibrium strategies in more complicated poker games. Our results from SMH and MMH characterize how game rules, player position, and betting history affect a player's optimal betting scheme. We find that poker variants with fewer betting opportunities polarize the betting strategies towards a fold/shove model. Adding more rounds of betting makes the equilibrium strategy more complicated such that all hands, even low-value hands, may be bet on. The potential of future betting changes how players bet in earlier rounds even when they have the same private and public information. We also find, regardless of variant, that players who bet earlier tend to bluff more aggressively because their bet will affect more players' choices. Along these lines, the effectiveness of bluffing in general decreases later in the game, especially in games where limited betting forces later bets to be a diminishing part of the total pot size. Poker games with more players demand a more defensive betting strategy because bluffs are more likely to be called by opponents with genuinely good cards.

Collusion shifts the equilibrium strategies. In particular, colluding players aim to take an advantage by coordinating their bets. Under strong collusion, colluding players bet more aggressively on weak hands. Furthermore, strongly colluding players bet more on complementary hands that have different win conditions, while they bet less frequently on hands that have the same win conditions. Even when some of the colluding players fold, the remaining colluding players still gain a significant advantage by knowing additional cards that will not appear later. In 3P-MMH, strongly colluding players win over 17.5% of an ante (in expectation) each game even if their opponent knows they are colluding. Under weak collusion, players coordinate their betting by playing extreme hands. In other words, colluding players bet more aggressively on low-value and high-value hands, while they bet more defensively on mid-value hands. As expected, we find that weakly colluding players have no significant advantage when they collude publicly but can gain a slight advantage if they collude in secret.

To study a variety of poker variants we built an abstraction to facilitate constructing poker game trees and using fictitious play to find a Nash equilibrium. The implementation of this abstraction was sufficient for considering SMH and MMH but suffers from complexity issues for bigger games. In particular, we initially experimented with poker variants more complex than MMH, but many of these exceeded the memory limitations needed to construct the game tree. Furthermore, some game variants that did fit in memory took a prohibitively long time to simulate. Modifying the best response algorithm to work in parallel is a stepping stone to make the implementation scale to larger simulations.

There are many places to go from here. To start, it would be helpful to study poker variants with more betting options — a facet of full-scale poker that we ignored. Perhaps our simplified betting scheme explains why we observed player position correlating with expected payoff. Namely, it intuitively makes sense that discretized betting would adversely affect earlier players. In addition, it is worth exploring poker variants with more players. Not only does this more closely model large table online games, but it may also identify relevant collusion trends. In particular, we could empirically determine the changing effectiveness of collusion as the number of colluding players increases. Other ideas include considering variants that use full fifty-two-card decks, have five-card hands, or have more shared cards. Looking at any or all of these variants would help to further characterize strategy trends in poker.

References

[1] Andrew W. Appel. Game theory analysis of hold ’em poker. Unpublished, July 2010.

[2] D. Billings, N. Burch, A. Davidson, R. Holte, J. Schaeffer, T. Schauenberg, and D. Szafron. Approximating game-theoretic optimal strategies for full-scale poker. In *In International Joint Conference on Artificial Intelligence*, pages 661–668, 2003.

[3] G. W. Brown. Iterative solutions of games by fictitious play. In *Activity Analysis of Production and Allocation*, pages 374–376. Wiley, 1951.

[4] Drew Fudenberg. *The theory of learning in games*, volume 2. MIT press, 1998.

[5] Sam Ganzfried and Tuomas W. Sandholm. Computing equilibria in multiplayer stochastic games of imperfect information. In *In IJCAI*, 2009.

[6] Andrew Gilpin and Tuomas Sandholm. A texas hold’em poker player based on automated abstraction and real-time equilibrium computation. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS ’06, pages 1453–1454, New York, NY, USA, 2006. ACM.

[7] Andrew Gilpin and Tuomas Sandholm. Lossless abstraction of imperfect information games. *J. ACM*, 54(5), October 2007.

[8] Andrew Gilpin, Tuomas Sandholm, and Troels Bjerre Sørensen. A heads-up no-limit texas hold’em poker player: Discretized betting models and automatically generated equilibrium-finding programs. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS ’08, pages 911–918, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[9] Daphne Koller, Nimrod Megiddo, and Bernhard von Stengel. Fast algorithms for finding randomized strategies in game trees. 1994.

[10] Parisa Mazrooei, Christopher Archibald, and Michael Bowling. Automating collusion detection in sequential games. In Marie desJardins and Michael L. Littman, editors, *AAAI*. AAAI Press, 2013.

[11] Michael Johanson Oskari Tammelin Michael Bowling, Neil Burch. Heads-up limit hold’em poker is solved. pages 145–149, 2015.

[12] Mark Whelan. Supervised machine learning for strategy-based abstraction with applications to poker. 2014.

[13] Martin Zinkevich, Michael Bowling, Michael Johanson, and Carmelo Piccione. Regret minimization in games with incomplete information. Technical report, 2007.