A Statistical Exploitation Module for Texas Hold'em

And Its Benefits When Used With an Approximate Nash Equilibrium Strategy

Kevin Norris
Department of Computer Science
University of Auckland
Auckland, New Zealand
knor031@aucklanduni.ac.nz

Ian Watson
Department of Computer Science
University of Auckland
Auckland, New Zealand
ian@cs.auckland.ac.nz

***Abstract*— An approximate Nash equilibrium strategy is difficult for opponents of all skill levels to exploit, but it is not able to exploit opponents. Opponent modeling strategies on the other hand provide the ability to exploit weak players, but have the disadvantage of being exploitable to strong players. We examine the effects of combining an approximate Nash equilibrium strategy with an opponent based strategy. We present a statistical exploitation module that is capable of adding opponent based exploitation to any base strategy for playing No Limit Texas Hold'em. This module is built around the idea of recognizing statistical anomalies in the opponent's play and capitalize on them through the use of expert designed statistical exploitations. The use of expert designed statistical exploitations ensures that the addition of the module does not increase the exploitability of the base strategy. In this paper we merge an approximate Nash equilibrium strategy with the statistical exploitation module. This approach has shown promising results in our initial experiments against a range of static opponents with varying exploitability's. It could lead to a champion level player once the module is improved to deal with dynamic opponents.**

Keywords*— *artificial intelligence, game AI, opponent modeling, nash equilibrium, exploitation, poker, Texas Hold'em

I. Terminology

The term "Opponent Modeling" has been ambiguously used in the literature to mean playing an opponent based strategy. This is not intuitive; intuitively one would think that the opponent model would be the collection of information about the opponent that represents the way in which the opponent plays. From this definition opponent modeling would just be the creation of this model, but in the literature it has been used to also incorporate the way in which the models are used to play an opponent based strategy. We believe that for ease of understanding, the ambiguous usage of the term "Opponent Modeling" should be broken into three terms:

- **Opponent Model:** the set of information that represents the opponent's play style.
- **Opponent based actions:** the way in which you use the given Opponent Model to determine actions that are suited towards the opponent you are facing.
- **Opponent based strategy:** The combination of creating an opponent model for your opponent and using this model to determine opponent based actions. There are two types of opponent based strategies that can be employed:
  - **Fully opponent based strategy:** The agent uses the opponent model to play an opponent based action for every action.
  - **Partially opponent based strategy:** The agent plays opponent based actions only some of the time.

II. Introduction

Many online poker players use Heads up Displays (HUD's). A HUD shows the player the recorded frequency statistics for each of his/her opponents and him/her self. Many novice online players play a relatively static strategy as they learn the game, which they alter only if they see anomalies in the statistics of their opponents. When they see these anomalies they devise statistical exploits: ways to alter their strategy to take advantage of the statistical anomaly such that they increase their win rate against the opponent. The statistical exploitation module presented in this paper is built around the idea of recognizing statistical anomalies and allowing the bot to capitalize on them through the use of expert designed statistical exploitations, in the domain of heads up no-limit Texas Hold'em poker. An overview of the modules design and its parts is given in section VI.

The module is not a complete strategy in and of itself. It is a module that is to be added to a base strategy. The resulting strategy of the combination is an enhanced version of the base strategy which is now able to capitalize on statistical anomalies, exploiting opponents. We decided on using an approximate Nash equilibrium strategy as our base strategy for testing the module. A Nash equilibrium strategy is described as "a strategy for each player of the game, with the property that no single player can do better by changing to a different strategy" in [1]. Johanson [1] reasons that while playing such a strategy one can do no worse than tie the game, since the opponent cannot do better by playing a strategy other than the equilibrium. Johanson [1] recognizes that "using such a strategy allows us to defend against any opponent, or allows us to learn an opponent's tendencies safely for several hands before attempting to exploit them". Finding a Nash

Unknown
Deleted: Reference

Unknown
Deleted: Reference

equilibrium in a complex game is very difficult, so instead they are approximated, providing a suboptimal strategy, a strategy which is exploitable. However the closer the approximation, the closer the strategy is to being un-exploitable, therefore, approximate Nash equilibrium strategies are very difficult to exploit. We felt that such a defensive strategy would be the ideal base strategy. We thus tried to preserve the desirable aspect of the base strategy, being difficult to exploit; and added the ability to exploit opponents, producing a strong bot that can do well against both weak and strong opponents.

The idea of combining a Nash equilibrium strategy and exploitation abilities has been done before in the poker bot Polaris created in [1]. We will give an overview of Polaris in section IV. The tradeoff between *difficulty to exploit* and *exploitability of opponents* is the main challenge faced when trying to add exploitation abilities to a Nash equilibrium strategy. In section VIII we will compare our technique of combining exploitation abilities with a Nash equilibrium strategy to the technique used by Polaris, and discuss why the statistical exploitation module had a much easier time with this challenge than the Polaris bot.

Section III provides a brief overview of the game of No Limit Texas Hold'em, the game that our module has been designed for. In section IV we discuss related work, particularly in the area of combining a Nash equilibrium strategy and opponent exploitation abilities. Section V discusses the role of frequency statistics in opponent modeling, how they are used by many opponent based strategies, and how our usage of them differs to enhance our strategy. Section VI provides an overview of the modules design along with a deeper look into the various parts of the module. Thereafter we illustrate our experimental methodology and present our results in Section VII, for the strategy resulting from an approximate Nash equilibrium base strategy and the addition of the statistical exploitation module. Section VIII provides conclusions and a comparison between our technique for adding exploitation to a Nash equilibrium strategy versus the technique described in [1]. Section IX discusses avenues for future work.

III. No Limit Texas Hold'em

We describe briefly the game of Texas Hold'em focusing on two-player no limit Hold'em as our module has been specialized for this domain. If a game consists of only two players, it is described as being a heads-up match. The game of heads-up no limit Texas Hold'em consists of four stages: pre-flop, flop, turn and river. During the pre-flop stage each player is dealt two hole cards, which only they can see. Two forced bets are contributed to the pot, these being the small blind (SB) and the big blind (BB) before any betting takes place. The big blind is usually double the value of the small blind. In the game of heads-up Texas Hold'em the dealer contributes the small blind and the non-dealer contributes the big blind. The dealer signifies the player who is first to act during the pre-flop stage of the game and last to act for each of the other stages of the game. The betting actions, which are common to all variations of poker, are described as follows:

- **Fold:** When a player abandons their hand, no longer committing any chips to the pot and giving up any right to contest the chips that make up the pot.
- **Check/Call:** When a player commits the minimum amount of chips with which he/she is able to continue to contest the pot. A check requires zero chips to be committed, and a call requires an amount greater than zero to be committed.
- **Bet/Raise:** When a player commits a larger number of chips than the amount necessary to continue to contest the pot, this is known as a bet. If a player is in the position where he/she must call a bet to continue, but then decides to invest more than the call amount in the pot, this is known as a raise.

In a no limit game a player may bet any amount they desire up to the total value of chips they possess. Once the betting in one stage of the game is complete and as long as no players have folded, play continues on to the next stage. Each further stage after the pre-flop stage involves the drawing of community cards from the shuffled deck of cards as follows:

- **Flop:** 3 community cards
- **Turn:** 1 community card
- **River:** 1 community card

In a standard heads-up no-limit poker game the chip stacks of each player would fluctuate between hands depending on who won the previous hand. To reduce the variance of this structure a variation known as Doyle's Game is played during our experiments where the starting stacks of both players are reset to a specified amount at the beginning of every hand.

IV. Related Work

A. Polaris

Polaris is a collection of techniques for creating poker bots, which are described in [1]. Johanson [1] outlines a new approach to calculating approximate Nash equilibrium strategies that require linear memory in the number of information sets instead of in the number of game states as was previously the case. This technique is called Counterfactual Regret Minimization. A new technique for calculating abstract game best responses is also presented, called Frequentist Best Response. A technique called Restricted Nash Response is illustrated, that creates a bot which is a compromise between the two previously mentioned techniques. The last technique presented in [1] is creating a meta-agent which is made up of agents created from the

SIT 12/3/13 12:14 PM
Formatted: Normal, Indent: Left: 0.63 cm, No bullets or numbering

Unknown
Deleted: Reference

previous techniques, along with a "coach" agent which decides which agent to use for every hand. We will delve deeper into the last two techniques as they are the ones which deal with combining an approximate Nash equilibrium strategy with exploitation abilities.

1) Restricted Nash Response (RNR)

Counterfactual Regret Minimization creates an agent that is difficult to defeat, but is unable to exploit its opposition so does not win very much. Frequentist Best Response creates an agent that can exploit specific opponents, but is easily defeated by opponents it is not designed to defeat. The Restricted Nash Response technique uses Regret Minimization to find a compromise between these two extremes, creating agents that exploit particular opponents or classes of opponents and still provide a bound on their exploitability. This strategy is constructed by finding a Nash equilibrium in a restricted game, where the opponent must play according to a fixed strategy with probability p. p is chosen when creating the strategy and determines the proportion of time the opponent must use the fixed strategy. p ranges between zero and one: a p of zero means that the opponent never plays the fixed strategy so a Nash equilibrium is computed; and a p of one means that the opponent only uses the fixed strategy so a best response is computed. All values for p between zero and one represent a tradeoff between exploitation and exploitability. Instead of constructing the usual two agents who play and adapt to one another for millions of hands to approach a Nash equilibrium, three agents are used for computing the Restricted Nash Response strategy: the RNR agent that learns the Restricted Nash Response and two agents for the opponent: a learning agent and a static agent. During the millions of games the RNR agent tries to minimize its regret against both the learning and the static components of the opponent. p is used to determine the amount of weight placed on each part of the regret.

2) Meta-Agent

In competitions the opponents will be unknown. So which of the previously discussed agents should be used against each opponent? Each of the previous agents have their pros and cons. In order to obtain the benefits of each, [1] created a team of agents comprising several agents of various types, thereby creating a meta-agent. The problem faced by the meta-agent of: which of the team of agents to choose when selecting an action is solved by expert algorithms. In this case the algorithm UCB1 was chosen, it is designed to trade off exploration and exploitation when choosing the agent. Since there are various types of agents in the team, the UCB1 algorithm uses different costs of exploration for different types. For example the cost of exploring the use of a Frequency Best Response agent is very high, whereas the cost of exploring a Restricted Nash Response is lower and the cost of using the Nash equilibrium agent is the lowest. Johanson [1] found that using a team of agents provided better results than using only one of its parts.

3) Discussion

A single RNR performs worse against arbitrary opponents overall in comparison to an equilibrium strategy. RNR agents are only able to exploit opponents that play strategies similar to the strategy they were trained against and are easier for opponents to exploit. This means that not only will the majority of opponents the RNR agent plays not be exploited by it, they will also be able to exploit it better than they would an equilibrium strategy. In the experimental results shown in [1] a team of RNR, and a team of Frequentist Best Response (FBR) agents played 4 opponents that they had a counter-strategy against and 2 opponents that were unknown. The FBR team performed worse than an equilibrium agent against the opponents and the RNR team did better. There is a cost associated with exploring the various agents in the team. The FBR team performed worse because the cost of choosing the wrong FBR agent is high, since FBR agents are easily exploited by opponents they are not designed against. The RNR team performed better because the cost of exploring suboptimal RNR agents is low, since the agents play strategies that are close to an approximate equilibrium strategy. In these experiments the RNR team performed better, not only on the known opponents, but also on the unknown opponents. Although [1] states that they have no prior reason to believe that the counter agents employed by their team of agents should work against the unknown opponents, it stands to reason that the team would only do better against the opponents if at least one of the agents in the team was able to exploit each of the unknown opponents.

To create a meta-agent using this technique that does better than an equilibrium strategy on average over arbitrary opponents should be relatively straightforward. A number of RNR agents along with a Nash equilibrium agent should do better than just a Nash equilibrium agent overall. The counter strategies the RNR agents employ will allow this meta-agent to exploit certain play styles; and if the meta-agent comes across a play style it does not have a counter-strategy for it should just play the Nash equilibrium strategy. The RNR agent's strategies are similar to the strategy employed by a Nash equilibrium agent and therefor also similar to one another's strategies. Due to this it may take the "coach" agent a long time to determine which agent is best against an opponent. The more RNR agents in the meta-agent the longer the exploration phase would take. Although the cost of exploring an RNR agent is low it does add up, and if the exploration phase is long enough, may cause the meta-agent to perform worse than a single Nash equilibrium agent.

Creating an optimal meta-agent using this technique would be very difficult. It would have to be determined what mix of Nash equilibrium and best response would be best for the meta agent, i.e. what p value the RNR agents should be created with. This is a difficult question because it could go either way. With a lower p value the RNR agents are closer to Nash equilibrium agents. This would make the cost of exploring them lower, but would also make it take longer to distinguish

SIT 12/3/13 12:16 PM
Formatted: Font:10 pt, Italic

SIT 12/3/13 12:16 PM
Formatted: Font:10 pt, Italic

SIT 12/3/13 12:16 PM
Formatted: Font:10 pt, Italic

SIT 12/3/13 12:16 PM
Formatted: Font:10 pt, Italic

SIT 12/3/13 12:16 PM
Formatted: Font:10 pt, Italic

SIT 12/3/13 12:16 PM
Formatted: Font:10 pt, Italic

SIT 12/3/13 12:17 PM
Formatted: Font:10 pt, Italic

SIT 12/3/13 12:17 PM
Formatted: Font:10 pt, Italic

SIT 12/3/13 12:17 PM
Formatted: Font:10 pt, Italic

Unknown
Deleted: Reference

the best among the team for the opponent, increasing exploration time. The RNR agents would also be unable to exploit opponents as well as those created with a higher *p*. A higher *p* value on the other hand would create RNR agents that are closer to Frequentist Best Response agents. These agents would have a higher exploration cost, but would be easier to distinguish which is best, decreasing exploration time. They would also be able to exploit opponents better. It would require a lot of testing to determine the *p* to use for creating the RNR agents and the number of RNR agents to use to create a meta-agent which minimizes its exploitability and maximizes its exploitation of opponents.

V. Frequency Statistics

Poker is an incomplete information game as well as a stochastic game, making it very difficult for a player to know where they stand in a hand. How can a player make good decisions without knowing what the opponent has, or what they will do? If a player wishes to make decisions based on the opponent they must have knowledge of the opponent. This knowledge comes in the form of history, the history of actions that the opponent carried out in their previous hands. Statistical opponent models record and use past play by keeping statistics of the opponent's action frequencies for use in later similar situations to improve the quality of decision making and to maximize profit. Keeping statistics on the opponent's action frequencies gives the player a good indication of how the opponent will act and react in given situations. Such indication follows from the probability distributions for the possible actions an opponent can make in each situation.

Schauenberg [2] describes two problems faced when using frequency statistics for opponent modeling. The first problem is that without enough observations of a situation the action distribution statistics can be largely inaccurate. There are two possible solutions to this problem: using priors, or not using the statistics to impact the action decisions until the game states they correlate to have been observed enough times to make their probability distributions statistically significant. The second problem is that opponents often alter their strategy throughout the game, invalidating the statistics that have been observed. If the opponent's play does not match the frequencies the player is using to make their decisions the opponent will be exploiting the player's incorrect information, causing them to lose money. This problem can be solved through decaying history, a technique through which the statistics are only affected by the latest *x* hands. *x* needs to be set to a small enough number, so that, if the opponent changes his/her strategy, the statistic will depict the alteration quickly.

The statistics used in opponent modeling research are usually limited in number and simplistic in nature, each generalizing greatly over a large set of game states. Many researchers are only using simple statistics such as: raise, call, fold percentage per street, VPIP (voluntary put money in pot) how often one calls or raises pre-flop, PFR(pre-flop raise) how often one raises pre-flop and aggression factor: the ratio of the number of times the player is aggressive vs. the number of times he is passive. Opponent model research that has utilized only such simple statistics includes [2], [3], [4], [5], and [6]. Adding in more statistics, while possibly making opponent models more expressive and better at capturing the nuances of an opponent play style, will also increase the computation and complexity of the program. Such complexity is needed to extract relevant data from the statistics to make betting decisions and leads to the tradeoff between program complexity and required computation verses opponent model expressivity.

Commercial heads up displays or HUD's, currently prevalent in online poker, provide a full observation model. They provide the player with any thinkable statistic because they save every hand. Online players have been using HUD's for some time now and have discovered many statistics, not generally used in opponent modeling research, that are very helpful for determining an opponent's play style. We believe opponent models can be greatly improved by adding some of the statistics that have become popular in online play through the HUD's. For example **3bet**, i.e. how often the opponent re raises a pre-flop raise, and fold to 3bet; **cbet**, i.e. how often an opponent bets the flop after raising pre-flop and fold to cbet and various other statistics are now a staple for online player's HUD's to allow them to gain a better idea of opponent's play styles.

The thought is that if statistics for a situation are specific for that situation they will be better than generic statistics in predicting the opponent's action. For example, in the case where the opponent was last to raise preflop and is first to act on the flop a statistic for cbet percentage (continuation bet percentage: how often the opponent bets the flop after being the last to raise preflop) will be more accurate in determining how the opponent will act than the generic fold, check, bet flop percentages. Online players have understood this and populated their HUDs with statistics for situations which occur frequently and are frequently played incorrectly. Much of the research on opponent modeling, however, uses generic frequency statistics instead of situation specific ones. The reason for this is that much of the opponent modeling research has been conducted on fully opponent based strategies, which must use the statistics when making every action decision. Researchers usually keep frequency statistics general, thereby reducing the complexity of determining which statistics to use in order to impact the action decision, and the weight that each statistic will be given. If a fully opponent-based strategy were to use situation specific frequency statistics, the logic to determine which statistics to use and their impact on the action decisions would have to be largely altered. One would have to consider which of the specific frequency statistics to use and how to weight their impacts on the action decision for every possible game state. Although the use of specific frequency

SIT 12/3/13 12:17 PM
Formatted: Font:10 pt, Italic

SIT 12/3/13 12:17 PM
Formatted: Font:10 pt, Italic

SIT 12/3/13 12:17 PM
Formatted: Font:10 pt, Italic

SIT 12/3/13 12:19 PM
Formatted: Font:10 pt, Bold

SIT 12/3/13 12:19 PM
Formatted: Font:10 pt, Bold

Unknown
Deleted: Reference

SIT 12/3/13 12:18 PM
Formatted: Font:10 pt, Italic

SIT 12/3/13 12:19 PM
Formatted: Font:10 pt, Italic

statistics could increase the accuracy of the opponent models used by fully opponent based strategies, it would make the agents much more complex, and would be difficult and time consuming to implement.

Consequently, our statistical exploitation module is based around the use of specific frequency statistics. Each exploitation uses a situation specific statistic along with some generic statistics to determine if the opponent is playing that situation incorrectly and if it can profitably exploit this. The module provides a partially opponent based strategy, using frequency statistics to determine an action only if an exploit is available for the situation. Unlike a fully opponent-based strategy it does not have to use frequency statistics for every situation; rather, it plays the base strategy in situations where an exploit does not exist or an exploit does not apply. The workings of the statistical exploitation module will be presented in more detail in the next section.

VI. Module System Design

We have created a module that we have added to SartreNL, presented in [7], that exploits players through the use of statistical exploits. This produces a bot which has a significantly increased win rate, without increasing its exploitability. This was accomplished by creating a statistical model that records detailed frequency statistics of an opponent in many contexts, and a number of exploits that provide highly profitable actions in the situation they apply to. The module also includes an opponent exploiter that tracks the exploits that apply to a given opponent model at any given time and provides the underlying bot with actions from the exploits when a situation arises in which one of the exploits applies. We have used SartreNL as the underlying agent. However, this module could be added to any underlying agent through minor changes made to the opponent exploiter and the underlying agent chosen.

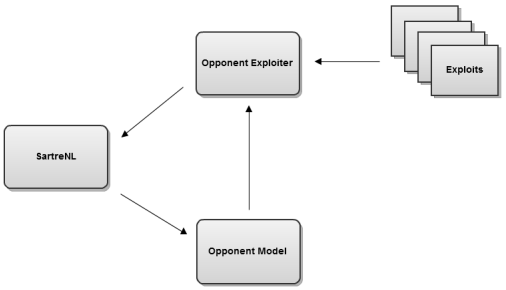

Fig. 1. Model of exploitation system design

Fig. 1 depicts the manner in which the individual parts work together to provide the addition of a partially opponent based strategy to SartreNL. Partially opponent based because the actions determined by the bot/module combination are not always dependent upon the module's opponent model; actions are only dependent on the module's opponent model when an exploit applies to both the current game state and the modules opponent model. Our opponent model is based on frequency statistics and thus faces the two problems discussed in section V. The first problem is easily overcome since our bot is playing a partially opponent based strategy. If the frequency statistics are not significant the agent merely plays the base strategy. The second problem, however is not yet overcome. In section IX we discuss a way in which we aim to solve the problem, but currently the bot can only guarantee safe exploitation against opponents who play static strategies.

SartreNL is used as the underlying poker agent because it plays an approximate Nash equilibrium strategy, providing the trait of being difficult to exploit, and is the only poker agent that was readily available to us. SartreNL uses the opponent imitator described in [8] to play the style of the player whom its case-base was trained on. In the case of SartreNL the case-base was trained on the Hyperborian bot's hand histories from the 2011 AAAI Heads-up No-Limit Texas Hold'em competition. This bot plays an approximate Nash equilibrium strategy, so the approximate Nash equilibrium strategy should emerge in SartreNL's play.

A. Opponent model

The opponent model is a collection of counters and variables representing the frequency statistics. After each hand is played out the counters are updated and then the statistics are recalculated based on the updated counters. In the poker framework used in the Annual Computer Poker Competitions (ACPC), messages are passed to and from the bots. The messages sent to the bots include all of the contextual information from the game; and the messages sent from the bots contain the actions they wish to take. The bulk of the code for the opponent model consists of methods that update the counters from the information found in the messages sent from the server. The statistics that are used in the model are often context based and in order to update the counters one has to check numerous conditions for each.

B. Exploit specifications

Exploits are basically rule modules that adhere to the generic exploit template. The generic exploit template includes two methods: applysToStats and getAction. applysToStats is the same for every exploit, it takes a stats model object and returns whether the given exploit applies to the stats model. An exploit can only apply if the prerequisite statistics have been observed enough times to be considered statistically significant, the threshold for statistical significance is expert defined for each statistic in each exploit. The getAction method is given the game context and returns an action if the exploit applies to the specified context. There are two versions of this method: one for pre-flop exploits and one for post-flop exploits. They differ in that the pre-flop exploits are given the two card hand ranking and the post-flop exploits are given a hand ranking calculated by SartreNL which takes into account

the community cards. For getAction to be called applysToStats must have been called previously and must have returned true.

The profitability of each exploit is determined by calculating the exploit's expected value to ensure the exploit is profitable. In order to determine this we must first calculate the equity. Equity is the chance that a hand or range of hands (the possible hand the opponent could have) will win the pot. We used Poker Stove [9] to calculate the equities that were used in our expected value calculations. Exploits have hand rank thresholds to determine which action to take. The value of these thresholds is dependent of the opponent's statistics. The threshold determines the range of hands with which the bot will take the action. The opponent's range is estimated, using the statistics from the opponent model. The estimation always takes from the top of the opponent's range so that, as long as the opponent model is correct, the opponent's range will never be stronger than the estimated one. However, it may be weaker but this works in our favor. SartreNLs hand range, based on the thresholds, and the opponent's hand range are used to calculate our equities when calculating the exploitation's expected value. Expected Value (EV) is the long term expected outcome of a given hand or situation. Expected value is calculated using the following equation:

*EV = [Our Equity] * [What we win] - [Opponent's equity] * [What we lose]*

Through the use of this calculation we can ensure that our exploits are profitable. This ensures that as long as the opponent model is accurate, our use of an exploit should never be exploitable by the opponent.

C. Opponent exploiter

The opponent exploiter is the module that provides the interaction between SartreNL, the statistical model, and the exploits. The opponent exploiter has a statistical model associated with it and has four lists of exploits: pre-flop, flop, turn and river. Each list represents all the exploits that apply to the associated statistical model for each of the four betting rounds. These lists are populated through the findApplicableExploits method which goes through each exploit, calling its applysToStats method, and supplying the statistical model the opponent exploiter is associated with as the parameter. If an exploit returns true, it is added to the list of exploits for the betting round it applies to. The opponent exploiter also has a method for each betting round that takes the game state information and calls the getAction method for each exploit in the list for the given betting round. If an exploit returns a non-null value this is passed along to the bot, that then uses the action.

VII. Experimental Results

A. Methodology

We required several opponents to challenge in order to evaluate the results of using our statistical exploitation module. Optimally we would evaluate statistical exploitation against a variety of competencies, ranging from easily exploitable to un-exploitable. The participants in the Annual Computer Poker Competition (ACPC) represent a good variety of computer players. While it is not possible to challenge the agents submitted to the competition directly, due to them not being publicly available, the hand history information is available for each agent that participated. Expert imitator case-bases were created for several of the no limit Texas Hold'em participants from the 2011 ACPC that imitate and generalize the opponent's style of play from their hand histories. These case-bases were created to be used by the expert imitation based framework described in [8], training each expert imitator on the decisions made in the competition by each of the chosen agent. Agents imitating an opponent through the use of the expert imitator always play a static strategy since the case-base used to determine their actions do not change. For these experiments this is fortunate because the statistical exploitation module is currently only able to handle static opponents. To create a strong poker bot its necessary to be able to deal with non-static strategies, and we will be addressing this in section VIII. The agents chosen cover a variety of exploitability, ranging from *highly exploitable* to *difficult to exploit*. Table 1 shows the author's views on the exploitability of the various agents.

SIT 12/3/13 12:21 PM
Formatted: Font:8 pt

Table 1. Exploitability of the opposition

Player	**Exploitability**
POMPEIA	highly exploitable
Kappa	highly exploitable
Hugh	highly-moderately exploitable
Lucky7	moderately exploitable
Hyperborean-iro	difficult to exploit
Hyperborean-tbr	difficult to exploit

Six opponents were challenged against SartreNL without exploits, and SartreNL with the addition of exploits which will henceforth be denoted as SartreNLExp. Each of the six bots played two seeded duplicate matches against both SartreNL and SartreNLExp. A duplicate match consists of 20,000 hands in total. 10,000 hands are initially played, the players then switch seats and the same 10,000 hands are played again. This way each of the players receives the cards that their opponents received before. The duplicate match style was used to decrease the variance that is normally involved in poker. To decrease the overall variance further, the same seed value was used for each of the duplicate matches played between each of the variants of SartreNL and the various opponents.

Table 2. Results

Bot Name	POMPEIA		Kappa		Hugh	
	Run 1	Run 2	Run 1	Run 2	Run 1	Run 2
Number of Hands	2090	2090	4160	4160	4703	5122
SartreNL	1,055	1,102	14,713	14,713	-629	-592
SartreNLExp	24,450	24,450	23,600	23,600	814	584
Difference	23,394	23,347	8,888	8,888	1,443	1,176
Bot Name	Lucky7		Hyperborean-iro		Hyperborean-tbr	
	Run 1	Run 2	Run 1	Run 2	Run 1	Run 2
Number of Hands	6392	6353	1265	1265	1357	1357
SartreNL	112	110	90	29	80	-18
SartreNLExp	504	252	486	448	427	442
Difference	392	142	396	418	348	461

Overall 24 duplicate matches were played; SartreNL played two duplicate matches against each of the opponents and SartreNLExp also played two duplicate matches against each opponent. To determine the effectiveness of the addition of the partially opponent based strategy based on statistical exploits on SartreNL the duplicate matches were split into two subsets: Run 1 and Run 2. Run 1 consisted of the first duplicate match SartreNL played against each of the opponents and the first duplicate match SartreNLExp played against each opponent. Run 2 consisted of the second duplicate match that SartreNL and SartreNLExp ran against each of the opponents. The bots played each run without any knowledge of the opponent they were facing.

In each run the match SartreNL played against a particular opponent is used as the base-line. The difference in performance between SartreNL and SartreNLExp can then be taken as the effect of the exploits. Some of the bots have some randomness associated with their strategies, so although the same hands and community cards came up in all matches this does not mean the bots chose the same action each time. Due to this it is likely that the scores fluctuated between matches. Furthermore an exploit was not applied to every hand and SartreNLExp chooses the actions as SartreNL normally would for hands where exploits are not applied to. Therefore, the situation could have occurred where exploitations were used and the overall score for SarterNLExp was lower than SarterNL's score. This would not have been due to the exploits losing money, since exploits always have a large positive expected value, it would have been caused by the fact that there is randomness in SarterNL's action selection process. This means that although the exploits had positive effects, SarterNLExp chose less profitable actions in the hands in which exploits were not used, causing SarterNLExp's overall score to be lower than SarterNL's.

To combat these problems we only compared the resulting scores of the hands in which exploits were used. There exists a duplicate match between SartreNLExp and each opponent and a corresponding duplicate match between SartreNL and each opponent in each run. The scores for the exploited hands were found for both the SartreNL and SartreNLExp match by going through the log files and tallying up the result for each of the hands in which exploits were used. The difference between these two scores shows the impact of the exploits much more accurately than the difference between the overall scores.

B. Results

Table 2 presents the results against the set of chosen opponents from the 2011 ACPC competition. The opponent in the match is given as the column heading, which is further split into the two Runs. The table is split into two sections due to spatial limitations. The *SartreNL* row depicts the outcome of SartreNL's matches. The *SartreNLExp* row depicts the outcome of SartreNLExp's matches. The match outcomes are depicted in milli-big blinds per hand, and only take into account hands in which exploits were used by SartreNLExp. Milli-big blinds records the average number of big blinds won per hand, multiplied by 1000. The *number of hands* row shows the number of hands in the match in which exploits were used by SartreNLExp. The *difference* row depicts the difference in win rate in milli-big blinds per hand between SartreNLExp and SartreNL, clearly indicating the effect the use of exploits had on the hands.

All of the results shown in Table 2 indicate statistically significant improvement in performance when statistical exploits are used. The results suggest that the statistical exploits module is able to appropriately determine and exploit statistical anomalies found in the play of opponents. The fact that significant increases are seen not only against the *highly exploitable* opponents, but also against the *difficult to exploit* opponents suggests that using an approximate Nash equilibrium strategy as the base strategy is having the desired effect. That being, SartreNLExp is able to remain difficult to exploit while concurrently exploiting its opposition.

SIT 12/3/13 12:22 PM
Deleted: t

VIII. Conclusion

In conclusion, we have presented an approach for exploiting statistical anomalies in the game of No Limit Texas Hold'em. Rather than create a fully opponent based agent we have created a module which can be added to a base strategy to create a partially opponent based agent. This allowed us to easily overcome one of the major problems faced by frequency statistic based opponent models, allowing us to play the base strategy as we wait for our frequency statistics to become statistically significant. We added the statistical exploitation module to an approximate Nash equilibrium base strategy to create an agent which was both difficult to exploit and able to exploit opponents. The module was able to safely exploit static opponents, by which we mean it was able to exploit opponents without making itself any more exploitable than the base strategy. Our experimental results show that the use of statistical exploitations not only did not make the agent more exploitable, but significantly increased the win rate of the agent in the hands in which exploits were used.

The statistical exploitation module's tradeoff comes not in the form of increasing the agent's exploitability to allow for the ability to exploit opponents, as done in Polaris [1]. Instead we limit the type of opponent exploitation to only statistical exploitation, allowing the module to exploit opponents without increasing the exploitability of the base strategy. This limitation means the statistical exploitation module is not able to fully exploit an opponent like a best response would. However we feel that ensuring the agent is as difficult for opponents to exploit as possible is more important than allowing it the ability to maximally exploit some opponents.

IX. Future Work

There are several improvements that could be made to increase the performance of the current system and increase the scalability. The most obvious improvement is the addition of more exploits. The more exploits that are in the system, the better it is at exploiting opponents. So far the system does not have a large number of exploits. Currently the frequency statistics model does not use any form of decaying history and is unable to remember opponents. Decaying history is an important upgrade as the system is currently vulnerable to any non-static strategies as it cannot react quickly to strategy alterations after it has built up a view of its opponent.

The module would be improved greatly by the re-implement of the statistical model to store hand histories. This would improve the performance and scalability of the statistical model by making it easy to include and compute further frequency statistics and allowing the ability to recognize previously played opponents again. Decaying history would also be easier to implement for a system that stored hand histories than for the current system that has only counters. If only counters are available, it is difficult to determine how to alter the counter to implement decaying history. While, if hand histories are stored, it would be simple to determine what the last M hands were and how they affected each counter. Stored hand histories would also allow the module to have statistic specific histories. In this way, statistics could be based on the last k occurrences of the situation the statistic is associated with, instead of just having the last M hands played.

References

[1] M. Johanson, "Robust Strategies and Counter-Strategies: Building a Champion Level Computer Poker Player," *M. Sc. Thesis*, 2007.

[2] T. Schauenberg, "Opponent Modelling and Search in Poker," *M. Sc. Thesis*, 2006.

[3] D. Billings, D. Papp, J. Schaeffer, D. Szafron "Opponent Modeling in Poker," Proceedings of the Fifteenth National Conference of the American Association for Artificial Intelligence (AAAI), 1998.

[4] D. Billings, L. Pena, J. Schaeffer, D. Szafron, "Using Probabilistic Knowledge and Simulation to play Poker," *Proceedings of the Sixteenth National Conference of the American Association for Artificial Intelligence (AAAI)*, 1999.

[5] Davidson, "Opponent Modeling in Poker: Learning and Acting in a Hostile and Uncertain Environment," *M. Sc. Thesis*, 2002.

[6] J. Rubin, I. Watson, 1st Initial. , "Opponent Type Adaptation for Case-Based Strategies in Adversarial Games," *ICCBR 2012*, Vol. , no. , 357-368, 2012.

[7] J. Rubin, I. Watson, 1st Initial. , "Successful Performance via Decision Generalisation in No Limit Texas Hold\'em," *ICCBR 2011*, Vol. , no. , 467-481, 2011.

[8] J. Rubin, I. Watson, 1st Initial. , "Similarity-based retrieval and solution re-use policies in the game of texas hold'em," *18th International Conference on Case-Based Reasoning, ICCBR 2010*, Vol. , no. , 465-479, 2010.

[9] (2012), PokerStove, . Available from: , : http://www.pokerstove.com/blog/ [Accessed: Oct 12, 2012].