WAVEFORM CLUSTERING AS AN EXTENSIVE FORM ABSTRACTION TECHNIQUE FOR POKER

BY

Kevin Kauth

THESIS

Submitted in partial fulfillment of the requirements
for the degree of MASTER OF SCIENCE
awarded by DigiPen Institute of Technology
Redmond, Washington
United States of America

December
2014

Thesis Advisor: Pushpak Karnick

DIGIPEN INSTITUTE OF TECHNOLOGY

GRADUATE STUDIES PROGRAM

DEFENSE OF THESIS

THE UNDERSIGNED VERIFY THAT THE FINAL ORAL DEFENSE OF THE MASTER OF SCIENCE THESIS TITLED

Waveform Clustering as an Extensive Form Abstraction Technique for Poker

BY

Kevin Kauth

HAS BEEN SUCCESSFULLY COMPLETED ON November 10, 2014.

MAJOR FIELD OF STUDY: COMPUTER SCIENCE.

APPROVED:

name date	name date
Graduate Program Director	Dean of Faculty
name date	name date
Department Chair, Computer Science	President

DIGIPEN INSTITUTE OF TECHNOLOGY

GRADUATE STUDIES PROGRAM

THESIS APPROVAL

DATE: November 10, 2014

BASED ON THE CANDIDATE'S SUCCESSFUL ORAL DEFENSE, IT IS RECOMMENDED THAT THE THESIS PREPARED BY

Kevin Kauth

ENTITLED

Waveform Clustering as an Extensive Form Abstraction Technique for Poker

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE AT DIGIPEN INSTITUTE OF TECHNOLOGY.

Pushpak Karnick date	Dmitri Volper date
Thesis Committee Chair	Thesis Committee Member
Xin Li date	Barnabus Bede date
Thesis Committee Member	Thesis Committee Member

ABSTRACT

Game tree abstraction is a hard computational problem due to the exponential storage bounds on the size of the tree. Hence, storing a complete game tree is not a viable option but for the simplest of two-person games. A more practical approach is to store a heuristic-based, trimmed representation of the entire tree that maintains as much information from the perfect tree as possible. Information related to the navigation of the tree is currently stored as a value for every node in the tree. However, this approximation is not guaranteed to be fully accurate in strategically representing multiple game states that may be viable and evaluate to the same "score". An alternative is to store a collection of values (waveform) at every state in the game tree.

This thesis investigates the use of waveform clustering as an abstraction technique applied to the extensive form of two player, limit, Texas Hold'em Poker. The goal of this research activity is to perform an initial evaluation of the validity of waveform clustering as a viable technique for problem abstraction. We present a detailed analysis of the problem domain, related approaches, our approach and the results of our research.

To my mom. Without your support none of this would have been possible.

TABLE OF CONTENTS

Page

LIST OF FIGURES ix

CHAPTER 1 Introduction 1

1. Problem Description 1

2. Goals 3

3. Problem Justification 4

CHAPTER 2 Related Work 6

1. Solving Games 6

2. Nash Equilibrium Points 7

3. Before Counter Factual Regret Minimization 7

3.1. Late Extensive Form Abstraction (2005 - 2007) 8

4. Counter Factual Regret Minimization 9

4.1. Default CFR Implementation 9

4.2. Local Maxima Problems 11

5. Reducing CFR Training Times 12

6. Building the Tree, Reducing Extensive Form Complexity of Poker . . 14

6.1. Suit Isomorphism 15

6.2. Clustering Dealt Card "Situations" 16

6.3. Alternating Dealt Card and Action Layers of CFR Tree 17

Page

CHAPTER 3 Methodology 20

1. Introduction 20

2. Design 22

3. Testing Strategy and Parameters 24

4. Implementation 25

CHAPTER 4 Results 29

1. Conclusions 44

CHAPTER 5 Future Work 45

REFERENCES 46

LIST OF FIGURES

Figure		Page
1.	Simple CFR Example	10
2.	Poker Hands Ranked	14
3.	Starting Hand Strategy Chart	16
4.	Deal Action Block Chain	18
5.	Action Node Tree	19
6.	Waveform Clustering Pseudocode	21
7.	Waveform Example	23
8.	Experiment Methodology	25
9.	Testing Framework Screenshot	27
10.	Testing Framework Class Diagram	28
11.	Agent Memory Directory Structure	30
12.	Action Node XML Example	31
13.	Chip Counts During Training	32
14.	Saturation During Training	33
15.	5K Saturation	34
16.	5K Chip Count	35
17.	10K Saturation	36
18.	10K Chip Count	37
19.	20K Competition Saturation	37
20.	20K Competition Chip Count	38

Figure Page

21. 50K Competition Saturation 39

22. 50K Competition Chip Count 40

23. 100K Competition Saturation 40

24. 100K Competition Chip Count 41

25. 250K Competition Saturation 42

26. 250K Competition Chip Count 43

27. Chip Count Trend Across Training Slices 44

CHAPTER 1

Introduction

1. Problem Description

Inside a computer's representation of a game, the set of game outcomes are represented as "nodes" or "game states", each node stored by the computer has the information that describes the game at that moment in time (i.e. what cards have been dealt, what actions are possible for a player to take). When a player takes an action, or the dealer deals a card, the state of the game changes and the current node transitions from the previous state to the now current state. Thus the nodes are attached to each other in a "parent to child" relationship, where parent nodes transition to child nodes as play progresses.

Each node in the tree, other than a starting "root" node, has exactly one parent and can have any number of children as determined by the rules of the game. A node with no children is called a "leaf" node, and represents the end of play. The root node represents the beginning of play. The entire structure of these nodes, from the root node to all the possible leaf nodes, is referred to as the "game tree" or the "Extensive Form" of the game.

Games with "hidden" information (i.e. the cards dealt to an opponent) are also known as games of "imperfect information". Each node in the tree for imperfect information games is considered as a collection of game states. These collections are made up of all the possible ways the hidden information could be distributed at that moment (i.e. all the possible cards your opponent could have). These are referred to as "Information Sets". You can consider "perfect information" games with no hidden information to still be made of Information Sets, where each Information Set contains a single State (i.e. Chess).

Using the Extensive Form of a game to develop AI playing strategies is one of the most popular approaches to solving games. The most significant aspect of games like Poker, in terms of algorithmic design, is that the set of every possible way the game can turn out is too numerous to store in a computer's memory. This forces computers attempting to play poker to group patterns of game outcomes together to bring the number of outcomes in an abstraction of the game down to a manageable size. The smallest Extensive Form for Texas Hold'em Poker is the" Limit" rule variant with two player - and even in its smallest possible form, the Extensive Form of Poker has complexity $O(10^{18})$ (very big).

In Texas Hold'em Poker, most of the complexity comes from possible ways of the cards being dealt and many successful algorithms today include a step that classifies common game "situations" from dealt cards and builds the Extensive Form abstraction on that set of classifications. This paper focuses on that classification.

Counter Factual Regret Minimization (CFR) is the algorithm that is most

commonly used in today's computer Poker competitions to navigate a game tree. CFR is an unsupervised learning algorithm similar to Monte Carlo tree search. Each iteration the game tree is navigated from the root to a leaf and an outcome is evaluated (i.e. win or loss by how much). Just like in Monte Carlo tree search, there is a back propagation step that updates a value at each node in the tree navigated during that iteration.

However, unlike Monte Carlo tree search, the utility values are stored on disk rather than kept in working memory. This allows a larger tree to be maintained, giving the algorithm more power (tree size correlates to strategic power). Utility values from previous iterations then influence the path taken through the tree in future iterations, and over many iterations the utility values congeal to represent a strategy.

At run time, the algorithm is very fast compared to algorithms that would perform a forward search of the game tree, such as a derivative of Alpha Beta pruning. CFR simply looks up the utility values for each child node and makes a decision. For Poker this is a great fit as the rules of the game demands fast decisions and the maintenance of a large tree is needed to form good strategies.

2. Goals

This research seeks to answer the following:

1. Is clustering game states using a waveform metric of strength aggregated across dealing events suited to building an Extensive Form abstraction for Texas

Hold'em Poker?

2. How does "Waveform Clustering" compare to a the most common implementation of game state classification used in competition today, all other details of agent implementation being identical?

3. Determine what more would need to be done to potentially use "Waveform Clustering" in a competition grade Poker agent.

3. Problem Justification

Computers do not naturally make guesses. However, real-world games include random events and hidden information that increase the size of the Extensive Form representation of the game [24, 21] such that a complete representation to the ideal solution cannot be feasibly stored. Hence, the algorithm targeting the game tree attempts to solve the game using some smart heuristics. In a fashion similar to other games like Chess [17, 25] or Jeopardy [11, 10], the goal of such algorithms is to consistently defeat human experts. This has not yet been accomplished for Poker [16].

At the highest levels of expert human play, there are many difficult elements in play that elude computers. The recognition of the Nash Equilibrium for the game, rather than being a single solvable answer, is an attempt to hit an ever moving target [28]. Furthermore, expert human players will engage in a "meta-game" where they will deviate from the current Nash Equilibrium point purposefully (with short term loss of utility) in order to encourage or force opponents to misjudge the current Nash

Equilibrium point going forward. Computers have trouble adapting in such situations as most algorithms are engineered to hone in on the current Nash Equilibrium point assuming that all opponents are rationally attempting to do the same.

To assist in the solution of Poker is to help confront a great frontier of Artificial Intelligence in general.

CHAPTER 2

Related Work

1. Solving Games

Von Neumann and Morgenstern in "The Theory Of Games and Economic Behavior" [27], published 1944, established the foundation of game theory for 1, 2, and 3 person zero sum games. A zero sum game is one in which the sum of all wins and losses in the game across all players is equal to zero (i.e. "If I win you lose"). Poker is an example of a zero sum game.

"The Theory of Games and Economic Behavior" [27] discussed a formal framework for describing a game as a set of decision nodes, at which strategies (a pattern of decisions across all decision nodes) can be assigned a quantifiable utility. It also was a fundamental work regarding the incorporation of random events into the formal description of a game, and in the description of formal strategies that incorporate random events (i.e. a player making a decision at random, or a card being dealt at random).

2. Nash Equilibrium Points

Shortly thereafter in 1950, John Nash presented the theory of "Equilibrium Points" in competitive games[26]. The fundamental concept in a "Nash Equilibrium" being that each player in the game is using the best strategy they can, given what they know about the strategies or all the other players in the game. More simply put, when a player reaches "Nash Equilibrium" they are playing the perfect strategy for the situation.

Applying the concept of Nash Equilibrium to machine learning algorithms, it has been shown that a complete solution for a game requires that no information is lost when the tree representing the game is described. Given such a tree along with infinite time and processing power, a strategy at a Nash Equilibrium for the game can be determined [1]. However, more often than not, time and processing power are not in infinite supply, and the "perfect" tree representation of a game is too large for modern computers to have a chance of ever solving. Poker was shown in 1990 to be one such game [21].

3. Before Counter Factual Regret Minimization

3.0.1. *Early Extensive Form Abstraction (1998 - 2004).* Around the turn of the 21st century, processing power had increased to the point that the Extensive Form complexity of Poker determined 10 years earlier was less intimidating. Analysis of methods to abstract complex Extensive Form games was beginning to mature [29]. General analysis into the characteristics of Nash Equilibrium points in Poker [30]

continued. Exhaustive statistical analysis [2, 3], Bayesian networks [22], statistical analysis plus neural networks [5], and other less successful approaches were attempted as ways to approximate the Nash Equilibrium of Poker in ways that deviated from "perfect solution" techniques that required navigation of the entire Extensive Form of the game. Approximating Nash Equilibrium with graph (i.e. tree) representations of complex games [20] would eventually lead to the present algorithmic favorite, Counter Factual Regret Minimization(CFR).

3.1. Late Extensive Form Abstraction (2005 - 2007). Later in the first decade of the 21st century, techniques for abstracting Extensive Forms of sequential games with a formal proof of maintaining the same Nash Equilibrium in the smaller Extensive Form abstractions (within a delta corresponding to the magnitude of the shrink) were developed that gave confidence to abstraction based techniques in general [13].

Attempts were then made to use efficiently abstracted Extensive Forms of Poker in various ways. Real time Nash Equilibrium approximation was considered [14] but was found to have performance issues on Extensive Form abstractions that were large enough to produce human competitive strategies. Ultimately incorporation of real time equilibrium calculations would depend heavily on off line calculations for half or more of the Extensive Form navigation during play.

Meanwhile, the use of Regret in machine learning extended into the realm of Extensive Form strategy formation. In 2006 Regret was applied as a proof of concept to several On line Convex Programs, including single card Poker [15]. Also,

gradient based solutions in general were also shown to be a solution to memory limitations previously encountered at run time by algorithms attempting to find Nash Equilibrium in large Extensive Form games [12].

4. Counter Factual Regret Minimization

A revolution in agents navigating abstracted Extensive Forms of games occurred with the development of Counter Factual Regret Minimization in 2007. This allows for the establishment of utilities at each node in the tree over time while an agent is being trained. These utilities then converge on values that if followed result in a strategy that bounds the Nash Equilibrium for that abstraction. A formal description with proof of Nash Equilibrium convergence for CFR can be found in [32].

This algorithm has become the overwhelming favorite starting point for agents attempting to play in the benchmark competition for artificially intelligent Poker players, the AAAI Computer Poker Competition [31].

4.1. Default CFR Implementation. The general CFR algorithm is an iterative adaptation of Monte Carlo Tree Search [4]. Starting at the root node of the tree at the start of an iteration, the tree is navigated by following a utility value that measures the Regret associated with each transition between a parent node and its set of children. When a terminal node in the tree is reached, the iteration ends and a Regret value is calculated for the iteration as a whole. An average Regret is then updated at each node traversed in the iteration. In this way, nodes higher in the tree in general accumulate more values into their running average over time than do the

terminal nodes of the tree. When applied to Poker, Regret information is maintained at nodes representing player actions. After many iterations, each player action node in the CFR tree contains a value representing the expected Regret associated with taking that action (i.e. transitioning into that node). Nodes in the CFR tree for Poker that do not represent player actions can have Regret values assigned to them, but in practice that is not useful as players have no control over deal events and cannot use the Regret information at those nodes to make any strategic decisions.

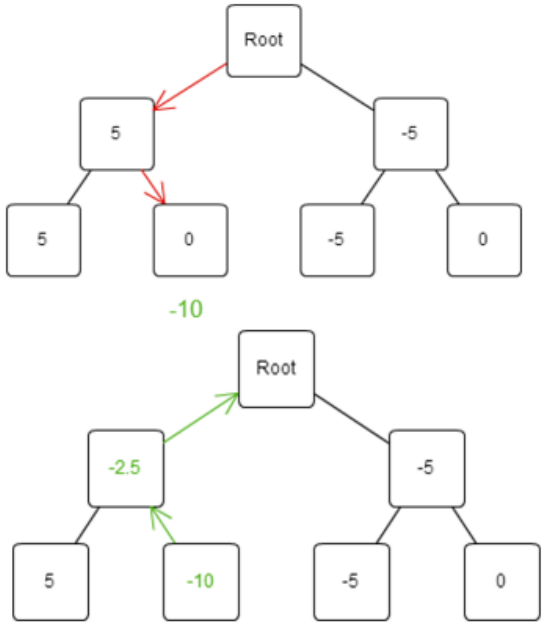


Figure 1. The above shows a simple example of CFR a single iteration. In Poker, this would represent the complete play of a single hand. The top image shows navigation down through the tree to discover a utility value. In the bottom image this utility value is then back propagated and averaged into the nodes that were traversed. Unlike Monte Carlo based algorithms, these utility values are stored on disk to be used for the next iteration.

4.2. Local Maxima Problems. When applying the general CFR implementation to Poker, some adaptations are needed to avoid getting trapped in local Maximae as the algorithm hill climbs over time to approach a Nash Equilibrium point.

Firstly, during initial training it is important to intelligently break ties between transitions that have the same Regret value. When training first starts, for example, all Regret values are set to 0. In Poker, the Fold action is immediately a terminal node that will evaluate to a non positive Regret value, the magnitude of which depends on the amount of betting that took place earlier in the hand. In a previously unexplored or briefly explored branch of the tree, if Fold is selected it can result in the entire sub tree being labeled with a negative Regret value, never to be explored again. The CFR algorithm, therefore, needs to have some built in parameter for how many attempts are needed before a Regret value can be trusted and in general should avoid breaking ties in Regret values with a Fold action.

Secondly, as CFR training progresses in large trees, Regret value averages will accumulate in the root nodes of the tree, leading them to become very stable as the tree converges on the Nash Equilibrium for the game abstraction the tree represents. In some problems spaces this is desirable, but not in Poker. At high skill levels, part of the strategy in Poker is to move the Nash Equilibrium (through short term utility loss) during play. Also, different players will adopt different strategies for various Nash Equilibrium points in the course of play, even if they make no attempts to deviously shift it. Therefore, a successful CFR implementation when applied to Poker must

maintain some level of instability in the Regret values of the high traffic root nodes to accommodate Nash Equilibrium shifts. Stable enough to form a coherent strategy, but unstable enough to adapt.

Recently, to address the elements of advanced human play where players will strategically take short term hits to their utility to confuse their opponent's understanding of the location of the current Nash Equilibrium for the game, there have been investigations into replacing traditional "Perfect Recall" algorithms with "Imperfect Recall" algorithms. This can also be described as "strategic forgetting" which keeps the utility values at the nodes in the tree used during CFR more current and allows for adaptation to attempted exploitation of the Nash Equilibrium's location. [19]

5. Reducing CFR Training Times

There is a direct correlation between the size of the Extensive Form abstraction used and the strength of the resulting algorithm. However, when applying CFR, increased nodes mean increased training time. Training time eventually becomes so expensive that it is not practical.

Conveniently, it has been formally proven [32] that an agent can play a copy of itself while establishing these utility values. The amount of time that it takes to establish the values for a player depends on the size of the tree, and the technique used to record utilities. The more nodes in the tree, the greater the number of hands the agent using that tree must play to establish statistically sound utility values at

each node in the tree. This can become burdensome when trying to converge upon a solution for very large trees. To counteract that specific problem, much attention has been given to sampling techniques during the training process to speed up convergence in very large game trees [18, 23]. The agents used in testing the hypothesis here do not use a large enough game tree to warrant aggressive training optimizations, and can converge within several days of training. Competition grade agents can take several months or years to train given the size of their tree abstractions relative to modern processing power.

Another large contribution to the reduction of training time was the streamlining of the fundamental card analysis algorithm that converts a set of 5, 6, or 7 cards to a 5 card Poker hand classification. For example, classifying a set of cards as "Pair" or "Straight". A truly impressive contribution in allowing for larger node trees to be processed was the development of the "2+2" hand evaluation algorithm for determining which 5 card poker hand (i.e Pair, Two-Pair, ect.) is represented from a set of 5, 6, or 7 cards [9]. Such an algorithm is foundational to any computational analysis of Poker and the size of the tree that can be solved depends greatly on how fast hand evaluation can be performed. Consisting of 7 array lookups, 6 addition operations, and two bit shifts; the "2+2" algorithm is an impressive feat of optimization.

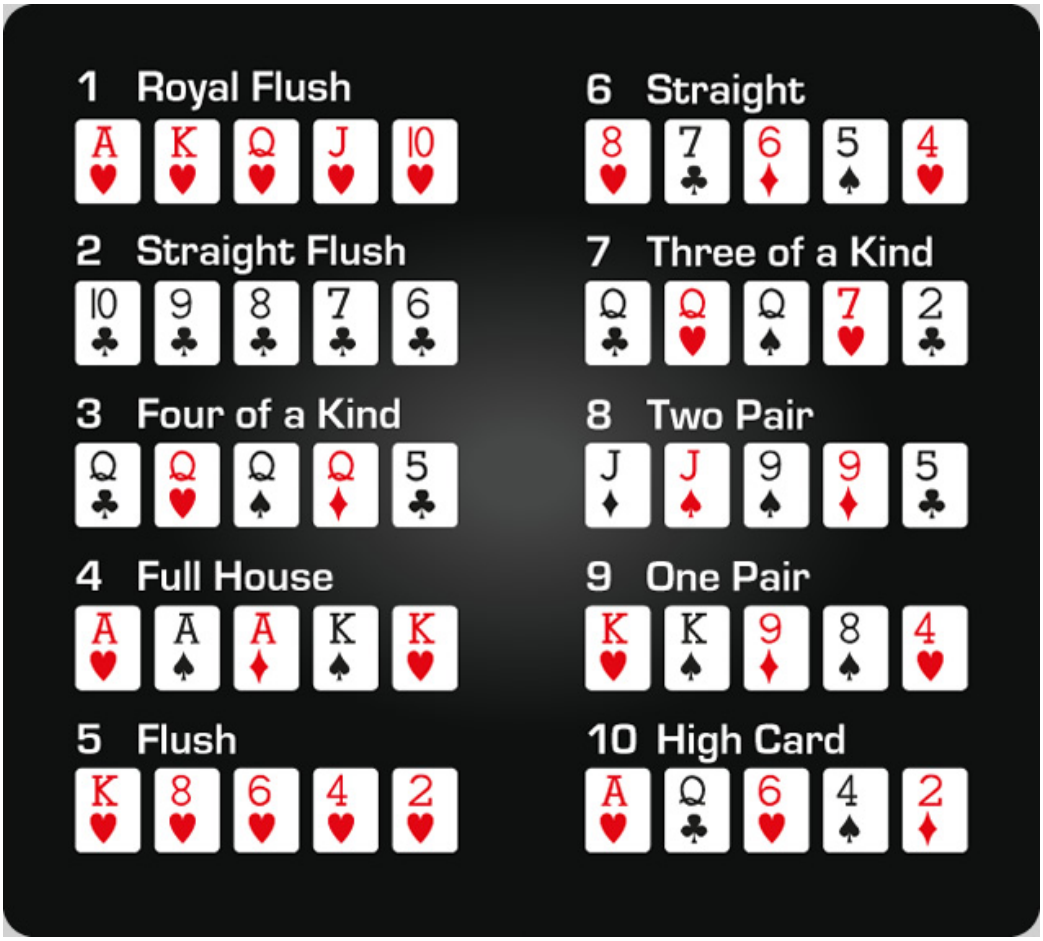


Figure 2. The above shows the different classifications sets of cards must be sorted into. This classification is one of the most fundamental operations in any poker related algorithm and has been extremely optimized via the "2+2" hand evaluation algorithm [9].

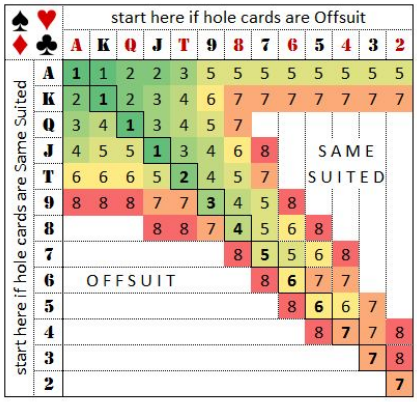
6. Building the Tree, Reducing Extensive Form Complexity of Poker

Since Poker was proven to have a prohibitively large complete tree, attention has been given to techniques that approximate the Nash Equilibrium with manageable processing requirements.

6.1. Suit Isomorphism. An effective technique for reducing the size of the complete tree while minimizing information loss is suit isomorphism. Simply put, it doesn't matter what suit a flush (five cards of the same suit) is in, the strategy for play with a flush remains the same. A great example of the power of this technique is in the enumeration of starting hands. The complete description of the starting hands of Texas Hold'em is every possible combination of two cards dealt from the 52 card deck. This yields 2652 (52 * 51) possible starting hands. However, applying suit isomorphism we consider hands that have identical face values as hands that result in the same strategy pattern, incorporating suit information only to consider if both cards are the same suit or different suits. This yields 169 (13 * 13) possible starting hands.

The concept of suit isomorphism can also be applied to later stages of the game with the same pattern in mind. "Which" suit only matters if a flush is possible. Once a flush becomes impossible for a given suit, all permutations of suits in that classification with the same card values yield identical strengths and map to identical strength situations in a CFR Extensive Form abstraction.

Suit Isomorphism is one of the most effective forms of Extensive Form abstraction because the size of the tree is reduced without loss of information or power. This is opposed to techniques, such as strength bucketing, which collapse rough categories of cards into common "Bucket" nodes in the Extensive Form abstraction, that inevitably do result in lost information.



Rank	Hole Cards	Playable Positions
1	AA, KK, QQ, JJ; AKs	Early, Middle, Late
2	TT; AQs, AJs, KQs; AKo	Early, Middle, Late
3	99; ATs, KJs, QJs, JTs ; AQo	Early, Middle, Late
4	88; KTs, QTs, J9s, T9s, 98s; AJo, KQo	Early, Middle, Late
5	77; *A9s thru A2s* , Q9s, T8s, 97s, 87s, 76s; KJo, QJo, JTo	Early*, Middle, Late
6	66, 55; K9s, J8s, 86s, 75s, 54s; ATo, KTo, QTo	Middle**, Late
7	44, 33, 22; *K8s thru K2s,* Q8s, T7s, 64s, 53s, 43s; J9o, T9o, 98o	Late***
8	J7s, 96s, 85s, 74s, 42s, 32s; A9o, K9o, Q9o, J8o, T8o, 87o, 76o, 65o, 54o	

* Playable in early position if game is loose/passive.
** Playable in middle position if game is loose/passive.
***Playable in late position if you are the first to bet.

Figure 3. The above shows Suit Isomorphism being used to simplify the possible starting hands of Texas Hold'em Poker along with an accompanying strategy classification for each entry in the chart. Because all chances for a Flush are the same at the start of the game it doesn't matter which suits the starting cards are. The only strategically important information at the start of the game is whether the two cards dealt are the same suit and the numeric values of the two cards.

6.2. Clustering Dealt Card "Situations". Because of the large number of ways cards can be dealt (order matters), mapping cards to playing situation classifications is a must if the Extensive Form complexity of Texas Hold'em Poker is to be reduced to a manageable point.

Clustering based on some evaluation of hand strength is widely used by competitive agents today [1, 7, 8, 32]. By branching through a node in the CFR tree that represents the strength of the player's hand, strategies are then separately formed in the tree for when the player has a strong hand or a weak hand.

Commonly, in evaluating what "situation" a set of cards maps to, algorithms will attempt to measure the strength of the hand that the set of cards represents. Typically for this measurement two assumptions are made. First, we assume that no one in the hand will fold. Second, we assume that all unknown cards are random.

With these assumptions we can simulate every possible way the remaining cards can be dealt and every possible hand our opponent can have. We can then count the number of times, given all the possible outcomes, that we win. Divide win by the total number of possible outcomes and we get the strength metric used by most competitive algorithms today.

At the point of a deal event, many algorithms use this metric differently. However, most algorithms in competition today will use this metric to assign the set of cards in play to a "Bucket" that all situations within a specified range of strength values fall into. For example, there could be a 10 bucket abstraction where strengths 0 - 0.1 go in bucket 0; 0.1-0.2 in bucket 1; etc.

6.3. Alternating Dealt Card and Action Layers of CFR Tree. Most Extensive Form abstractions used to apply CFR to Poker wind up with a structure that alternates between sections of player action and deal events. The deal events are clustered into "Buckets" that represent the set of "card situation" classifications that can result from how the cards are dealt (i.e. Lots of betting and I have a strong hand or No betting and I have a weak hand); whereas the Action sections represent the actions taken by the players themselves where decisions are made based upon the Regret values maintained at each node.

Most abstractions will maintain the pure Extensive Form of Poker for the Action sections of the tree and dial the number of Buckets up and down in the Deal sections of the tree to manipulate the size of the overall tree. Each algorithm in competition today has its own "special" mapping function from cards to Buckets.

If their mapping function results in less lost information from the pure Extensive Form of the game compared to an abstraction of similar size but different mapping technique then they will have an advantage. See Figure 4 and Figure 5 for visual examples of the tree structure common to most CFR implementations in competition today.

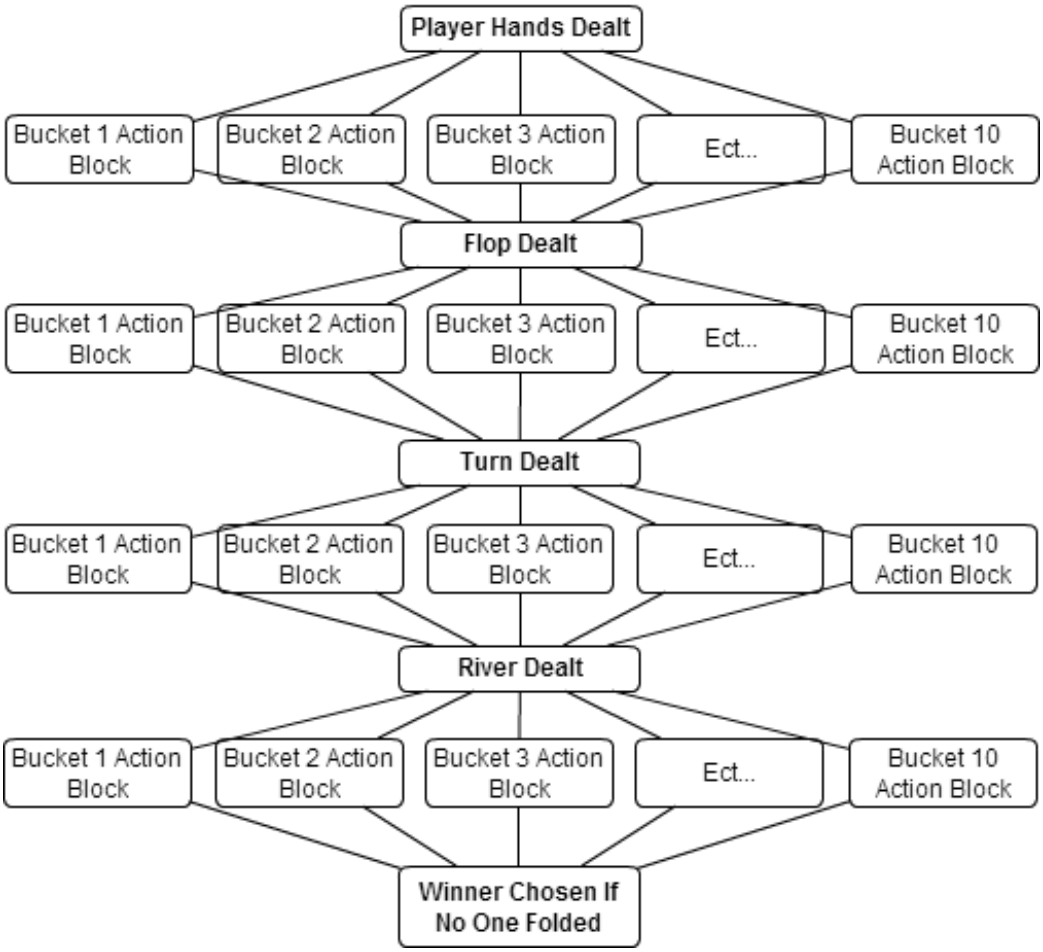


Figure 4. The above shows the typical high level structure of a CFR tree for Poker. Notice that Deal and Action sections of the tree alternate as betting rounds are punctuated by random Deal events that are outside the control of the players. Action node sections contain Regret values at each node that governs play, while a mapping function decided by the algorithm creator determines how cards are mapped to Buckets (aka "situations") that correspond to a different strategy in the Action section of the sub tree that follows.

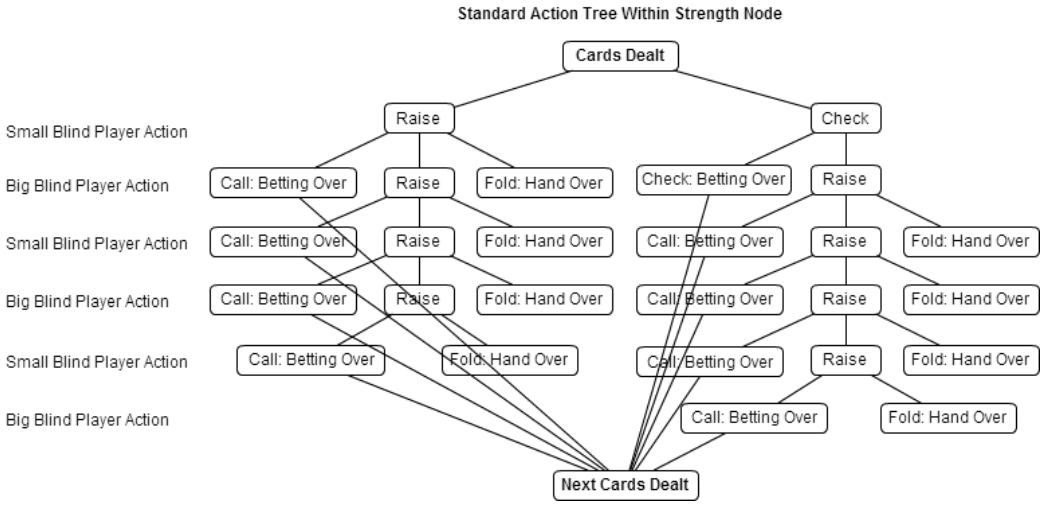


Figure 5. The above shows the structure of an Action section of a CFR tree for Poker. Each transition between nodes here has a Regret value that is maintained over time by the algorithm that guides playing decisions.

CHAPTER 3

Methodology

1. Introduction

A typical strength metric used to evaluate hand strength proceeds by (a) enumerating all possible outcomes for dealt cards and opponent cards, (b) dividing the number of wins by the total number of outcomes, thereby (c) obtaining a scalar representation of the game state's 'value.' The proposed method evaluates the metric at each possible way ALL cards can be dealt in the future of a given hand and then maintaining a waveform of those possible outcomes at nodes where NOT ALL cards have been dealt yet, rather than collapsing them all into an average. I refer to this as "Waveform Clustering". See Figure 6 for a pseudo code comparison of the typical strength evaluation metric and the Waveform Clustering metric I am proposing.

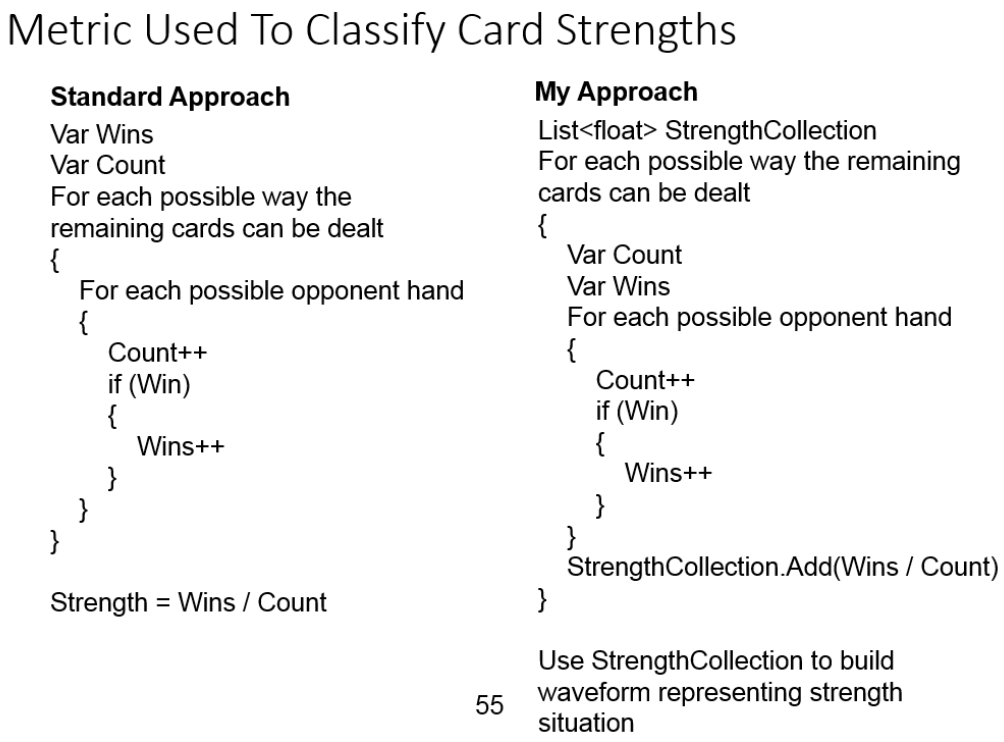


Figure 6. The above is a side by side comparison of the typical strength metric used as the Control (left) and our suggested approach, the Experimental (right). These calculations of the strength metrics are used to classify card sets into strength situations or "Buckets" for use in the standard Counter Factual Regret Minimization algorithm described in [32]

2. Design

Waveform Clustering allows for disparate situations, recognized commonly by expert human players as strategically different, to be accounted for. Figure 7 shows an example of the strength waveform metric as it would be calculated throughout a hand.

On the left is a player with a flush draw and on the right is a player with a small pair. Notice that if their waveforms were to be averaged into a single value (the typical approach [7, 8, 32]) that the value would be similar for both. However, notice that the player on the left with the flush draw has value spikes at the extreme ranges of strength. This reflects that with somewhat equal likelihood the player will end up with either a hand worth nothing or a monster hand that will beat everything. Compare that to the player on the right with the small pair. This player has a spike of concentrated values in the center of the waveform reflecting that no matter which cards are dealt, the player will end up with a hand that will win half the time. The difference between these two players isn't the chance they have to win, they will both win approximately half the time against a random opponent.

The difference between these two players is when they know the strength of their hand. Human experts would assert that the player with the draw on the left wants to avoid large bets until more cards have been dealt whereas the player on the right should favor aggressive play earlier in the hand to force other players without known hands to fold early [6]. The right hand player already knows approximately how strong they will be at the end of the hand and should seek strategies that

capitalize on that while others are still relatively uncertain about how their hands will turn out.

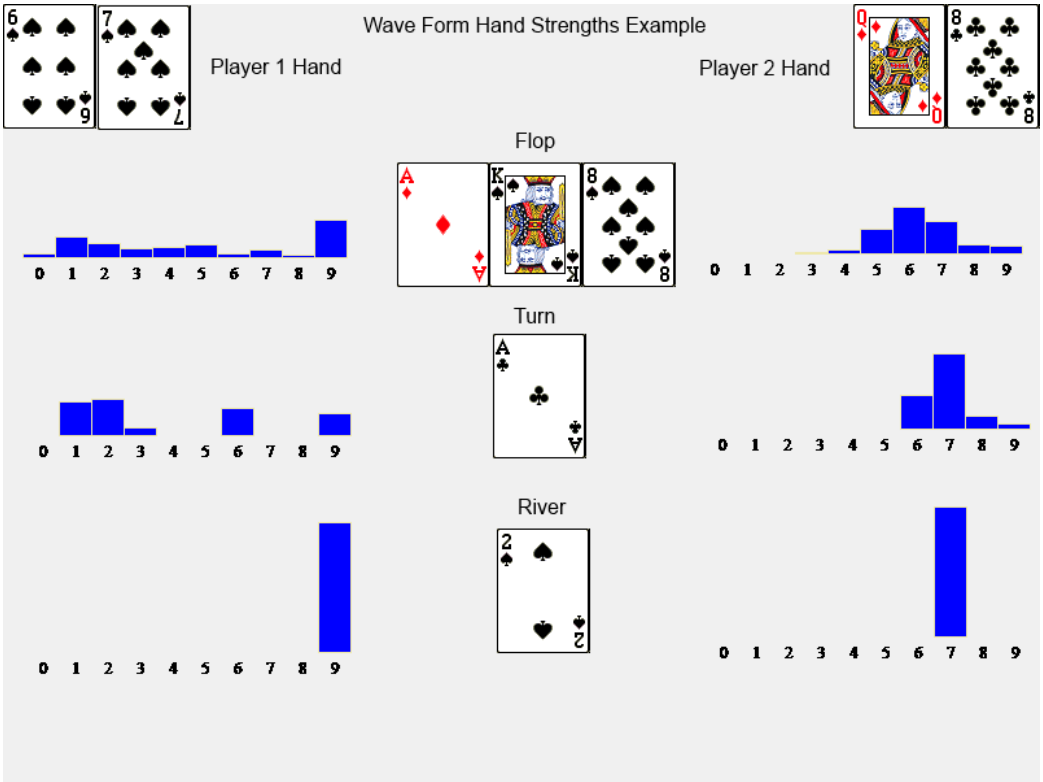


Figure 7. The above is an example of the waveform metric calculated throughout a hand after the deal events for that hand. Notice the probabilities converge after deal events until a single strength bucket remains after the last deal event in the hand.

Waveform Clustering is applied to a CFR implementation by maintaining an average waveform at nodes in the Extensive Form that result from deal events. This is similar to how utility information is recorded at nodes that result from player actions (that then guide agents to avoid Regret), except that it is a waveform rather than a single scalar value. When traversing the tree during play and deciding which Bucket to classify a card situation in, the node selected is the node with the closest average waveform (recorded over all hands) compared to the waveform produced by

evaluating the current set of cards.

3. Testing Strategy and Parameters

To measure the efficacy of using the aforementioned Waveform Clustering metric to cluster card situations in the Extensive Form abstraction for Texas Hold'em, we conducted an experiment comparing the typical implementation with an identical implementation that uses the Waveform Clustering technique.

Two metrics are recorded by the testing framework: the comparative chip counts of the players after each hand and a Saturation metric representing how familiar the players are with the section of the Extensive Form tree they traversed in a given hand.

More specifically, Saturation is calculated as the number of nodes visited by the player that have been visited 50 times or more divided by the total number of nodes visited by the player during a the hand. Saturation therefore can tell us two important things. First, when Saturation values are consistently 1.0, the player has finished training. Second, during play, if the Saturation values take a sudden downturn we know the player was forced into a section of their tree that they had not visited before because the path through the tree that they usually take has unexpectedly become unprofitable and they were forced to adapt.

We trained each player, Control and Experimental, with equal iterations. Memory slices were sampled at 10K, 20K, 50K, 100K, and 250K iterations. A 5 bucket Extensive Form abstraction was used for the shared CFR implementation.

By playing similar training slices from each player against each other we can then evaluate comparative strengths throughout training and draw conclusions based on the patterns that emerge. With a 5 bucket CFR implementation, the Extensive Form the players train on takes approximately 10 days of continuous automated play to converge on 1.0 Saturation.

Comparative play between the Control and Experimental implementations are rounds of 10K hands, each round resetting the memory banks of the players to their original self trained memory slice. Multiple rounds are played at each training slice between the Control and Experimental players until statistically valid data is gathered (fewer rounds for consistent blowouts, more rounds for closer results).

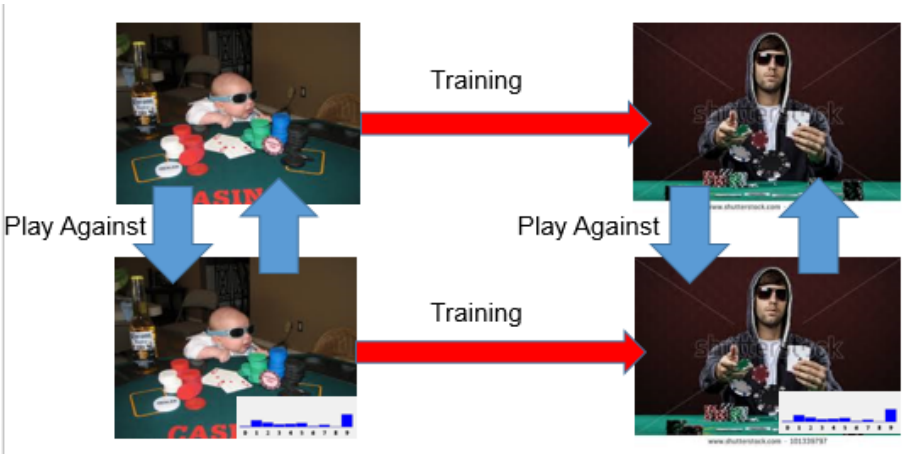


Figure 8. The above is a simple visualization of the experimental methodology used to collect data.

4. Implementation

We developed a framework capable of playing two players against themselves (for training) or against each other (for comparative evaluation). The framework

switches the classification algorithm used to map cards to strategies for each algorithm under consideration, keeping all other aspects of the CFR implementation used by the players the same. Most algorithms in competitive play today share a core CFR implementation and their differences lie in choices for their Extensive Form abstractions and parameters chosen for the typical CFR algorithm (i.e. how fast utility values should converge). Those parametric details will be chosen arbitrarily from the set of possible valid choices. As long as they are identical between the Control and Experimental agents, we assume they are irrelevant to this research.

Figure 10 shows the class diagram for the framework. A Form class displays the user interface (UI) and serves as the "dealer" governing the dealing of cards and direction of play. It maintains two Player objects that have Hands and TreeNavigation utilities. The TreeNavigation utilities, in turn, utilize a GameTreeDirectoryStructure class that handles disk IO for that player as needed, and a ActionBlock class that has a HandStrength class derivative as a "parent" node in the overall tree. The HandStrength classes are where the implementations between the two player types, Control and Experimental, differ. ActionNode objects are sub-trees within an ActionBlock and manage the utility values for each action the player can take, dynamically read and written on disk via the GameTreeDirectoryStructure utility class.

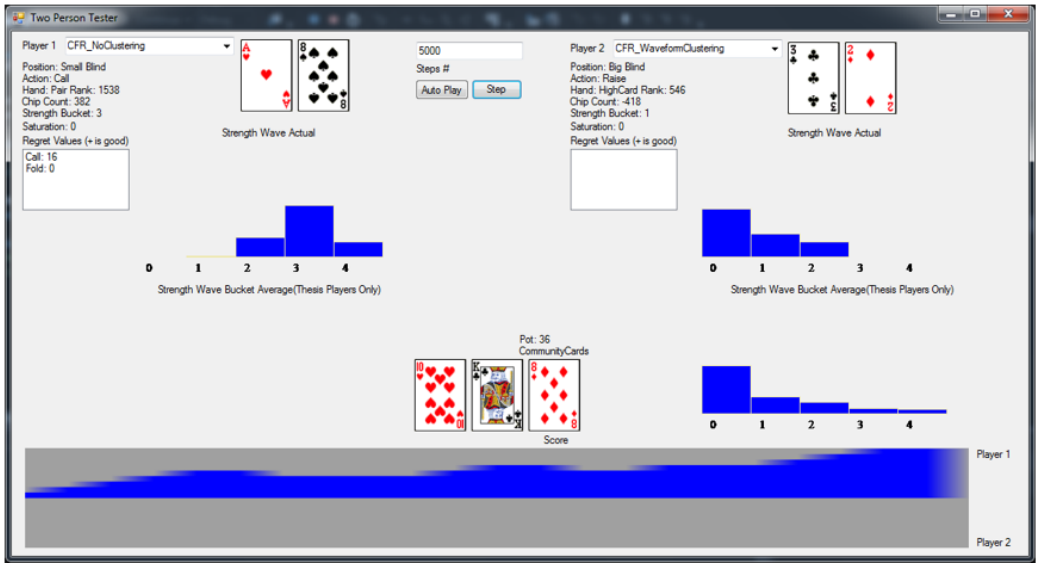


Figure 9. The above is a screenshot of the testing framework as it plays the Control and Experimental CFR implementations against each other. All recorded metrics as well as hand classifications are shown here, as well as output to disk after each hand. The framework allows for a set number of iterations to be played automatically.

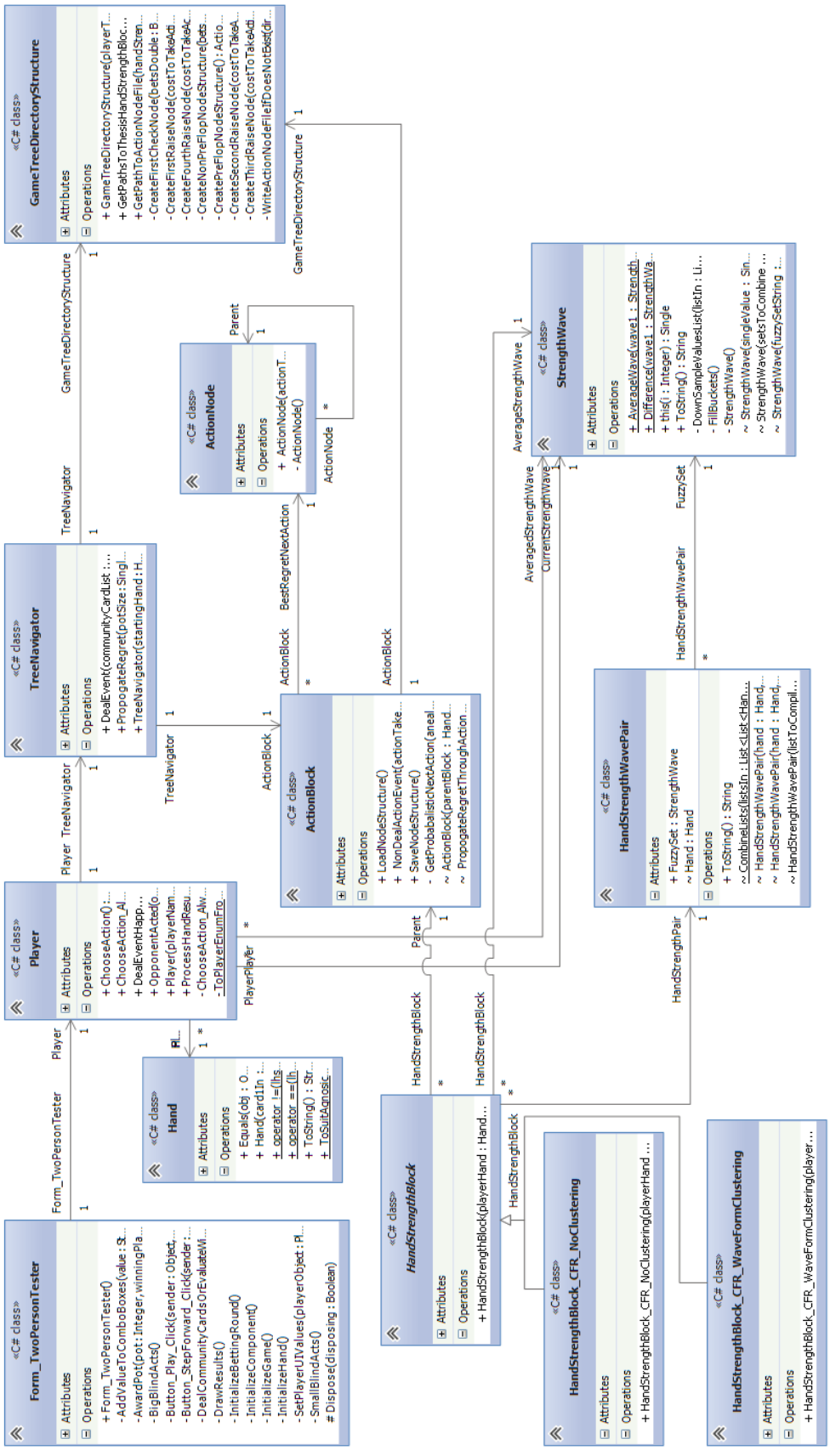


Figure 10. The above shows the significant classes in the testing framework.

CHAPTER 4

Results

To collect results, both the Experimental and Control players were trained using self play for 250,000 hands. After each hand, utility values were updated in .xml files organized in a directory structure on disk that corresponds to the extensive form abstraction. See figure 11 for an example of the directory structure on disk and figure 12 for an example of a .xml file containing utility values for a specific Bucket in the tree.

Chip Count and Saturation for the players are output after each hand to a line in a .csv file that can then be converted into the graphs found later in this chapter.

During the 250K iterations of training, a copy of the directory structure representing the memory of the players was set aside after 5 K, 10K, 20K, 50K, 100K, and 250K iterations. These copies of these memory banks were then used as input to the trials where the Experimental and Control players would play against each other. Each iteration of the competitive trials was restarted with a fresh copy of the memory banks for each player at that training level. The figures at the end of this chapter show the average Saturation and Chip Count metrics produced during

the self training of the Control and Experimental players as well as for the multiple trials of Control vs Experimental at each training slice.

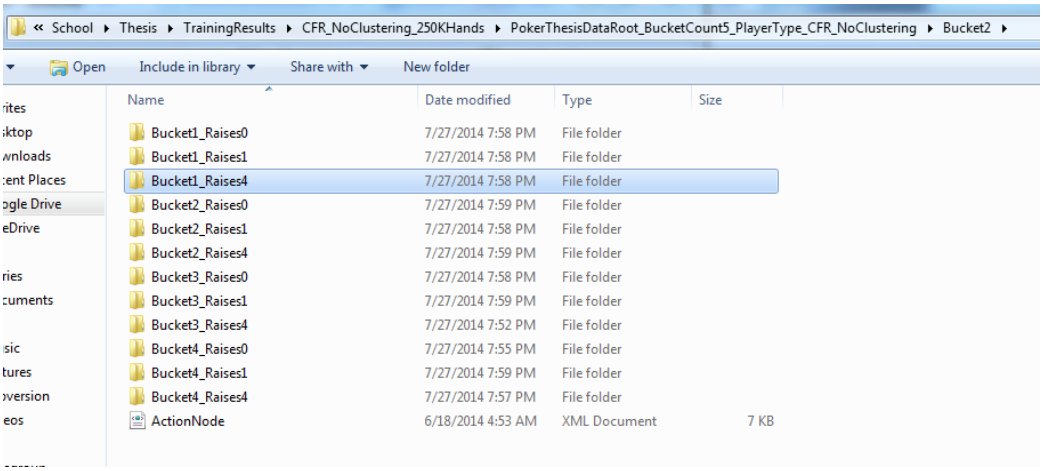


Figure 11. The above is an example of the file structure on disk used to store the memory of agents. Each directory in the directory structure represents a "Bucket" or "Card Situation" that corresponds to a strategy. The utility values for each strategy available in each Bucket are recorded in an ActionNode.xml file in each directory. Transitions to other Buckets deeper in the tree are represented by the sub directories in each directory. Transitions are determined by the number of raises that occurred during a round of betting and the card(s) dealt after betting in the current round is concluded.

ActionNode.xml

```
<Children>
  <Action>PostSmallBlind</Action>
  <Records>100</Records>
  <Regret>-0.9999933</Regret>
  <Children>
    <Action>PostBigBlind</Action>
    <Records>100</Records>
    <Regret>-0.220344886</Regret>
    <Children>
      <Action>Raise</Action>
      <Records>100</Records>
      <Regret>-0.330000043</Regret>
      <Children>
        <Action>Raise</Action>
        <Records>100</Records>
        <Regret>-0.0196023267</Regret>
        <Children>
          <Action>Raise</Action>
          <Records>97</Records>
          <Regret>2.1958766</Regret>
          <Children>
            <Action>Raise</Action>
            <Records>100</Records>
            <Regret>3.14326882</Regret>
            <Children>
              <Action>Call</Action>
              <Records>97</Records>
              <Regret>6.79381466</Regret>
              <CostToTakeAction>2</CostToTakeAction>
            </Children>
            <Children>
              <Action>Fold</Action>
              <Records>0</Records>
              <Regret>0</Regret>
```

eXtensible Markup Language file

Figure 12. The above is an example of a .xml representation of the utility values that correspond to each possible action that can be taken in a given strategic situation (classified by previous actions and cards dealt). In this situation, the agent has determined over many iterations through this section of the tree that placing blinds at the start of play has a negative utility on average (its a bet forced by the rules of play regardless of the hand held by the player) and that in this strategic situation Raising is profitable. Negative Regret is bad, positive Regret is good.

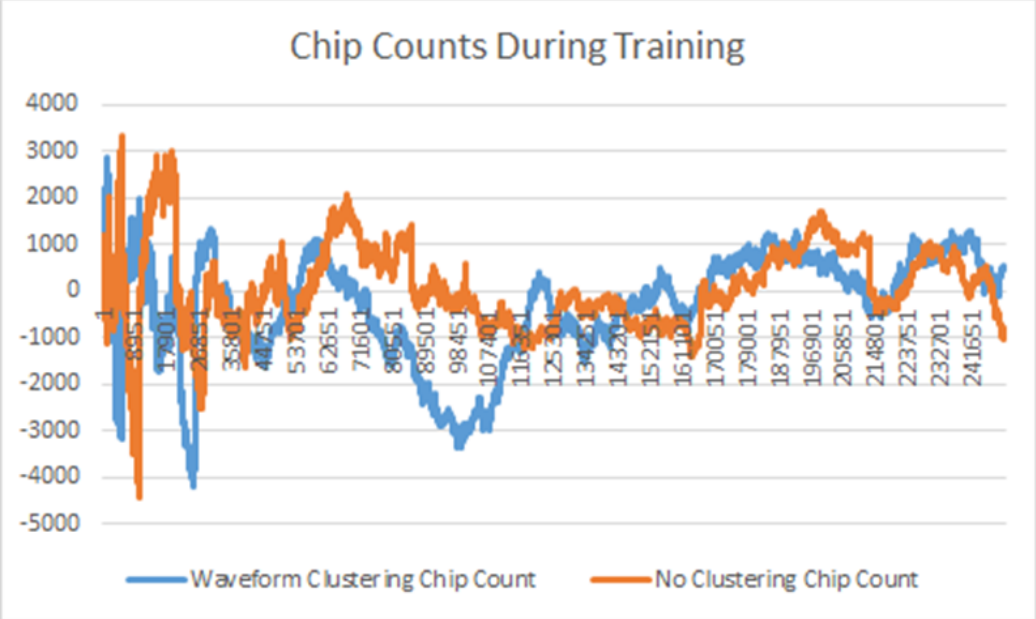


Figure 13. The above shows the variance in chip counts of the Control (No Clustering) and Experimental (Waveform Clustering) players as they trained via self play over 250K iterations. This shows the expected variance in the chip count metric that can occur during play with identical players and is a useful reference for determining the number of rounds the players need to play at various training slices when being compared to each other in order for statistically valid data to be collected.

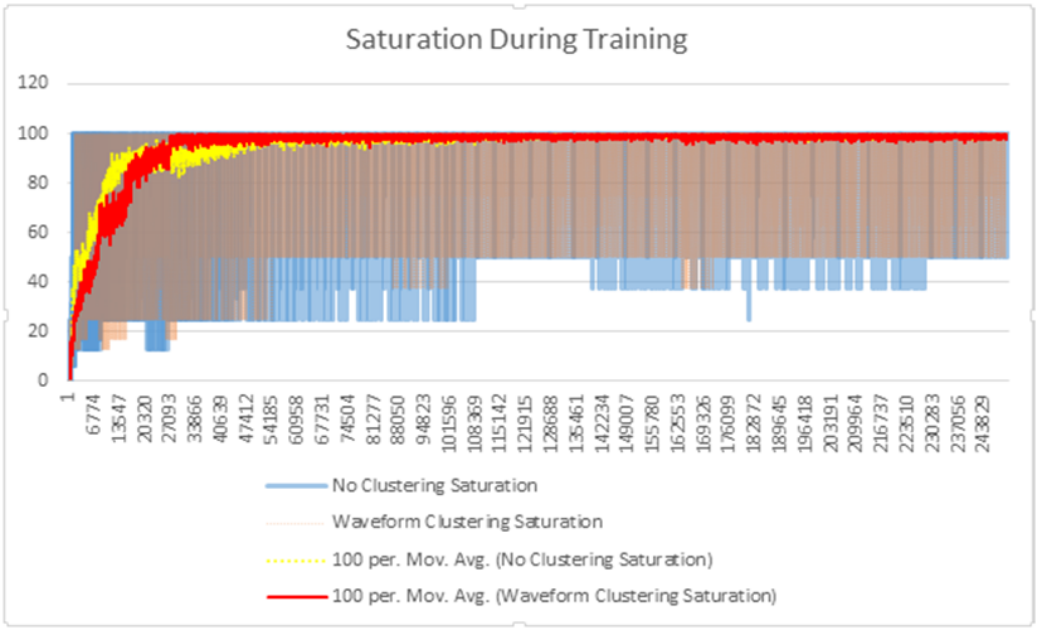


Figure 14. The above shows the variance in Saturation of the Control (No Clustering) and Experimental (Waveform Clustering)l players as they trained via self play over 250K iterations. This shows the expected variance in the Saturation metric that can occur during play with identical players and is a useful reference for determining the number of rounds the players need to play at various training slices when being compared to each other in order for statistically valid data to be collected.

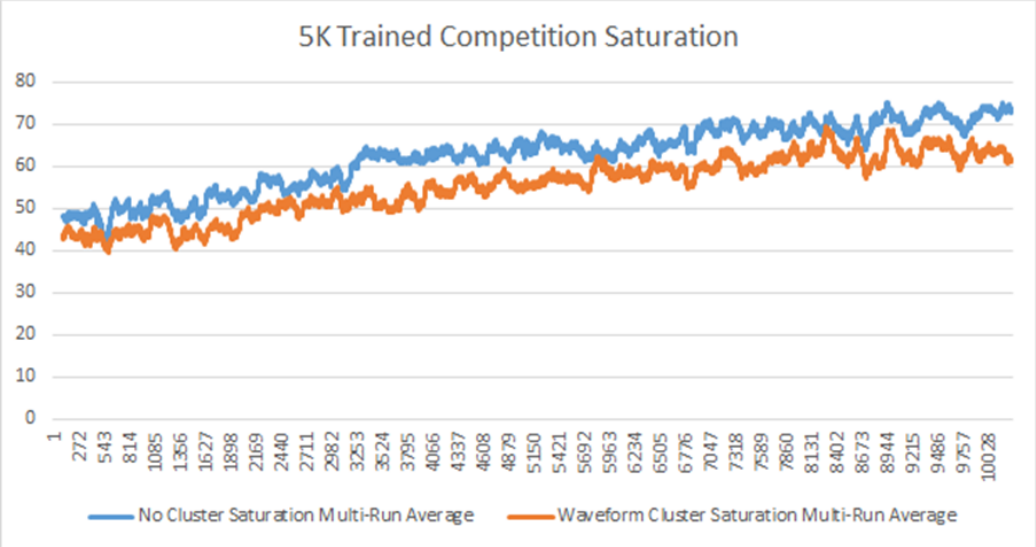


Figure 15. The above shows the Saturation of the Control (No Clustering) and Experimental (Waveform Clustering) players averaged over all rounds played with 5K iterations of training as a starting point for the memory banks of the players. The Control player consistently has higher Saturation values, suggesting that it has converged on a strategy faster during training than the Experimental player.

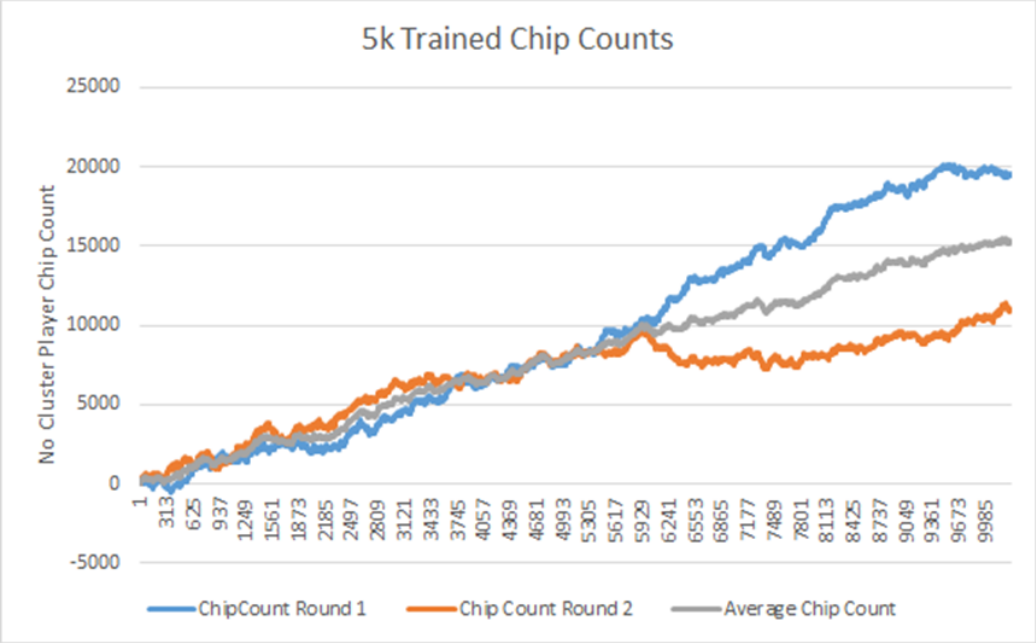


Figure 16. The above shows the chip count of the Control (No Clustering) and Experimental (Waveform Clustering) players averaged over all rounds played with 5K iterations of training as a starting point for the memory banks of the players. The Control player consistently wins at this training slice with a margin of approximately 15,000 chips.

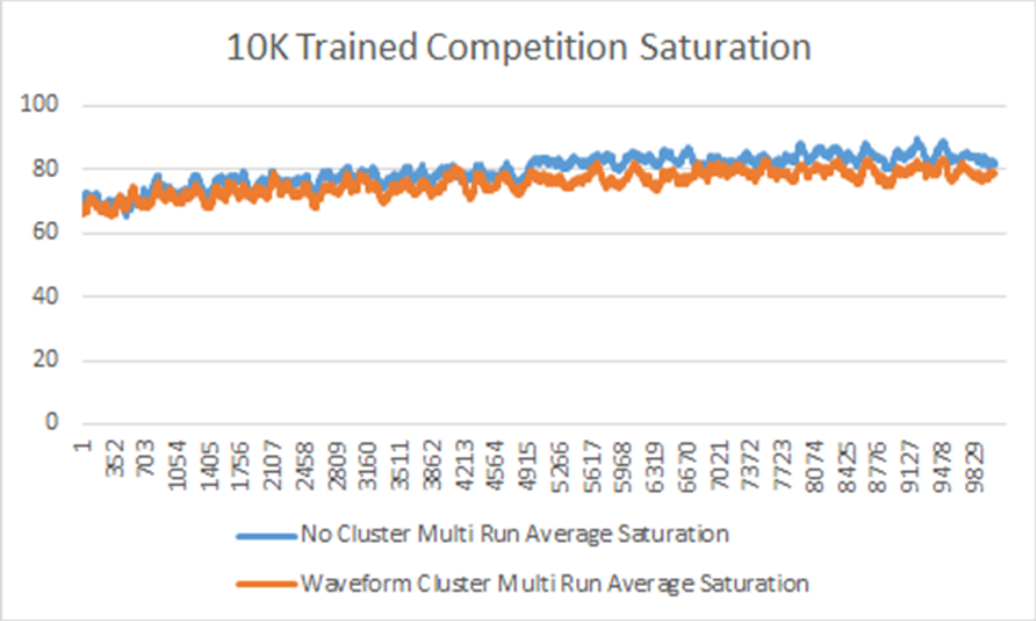


Figure 17. The above shows the Saturation of the Control (No Clustering) and Experimental (Waveform Clustering) players averaged over all rounds played with 10K iterations of training as a starting point for the memory banks of the players. The Control player consistently has higher Saturation values, suggesting that it has converged on a strategy faster during training than the Experimental player.

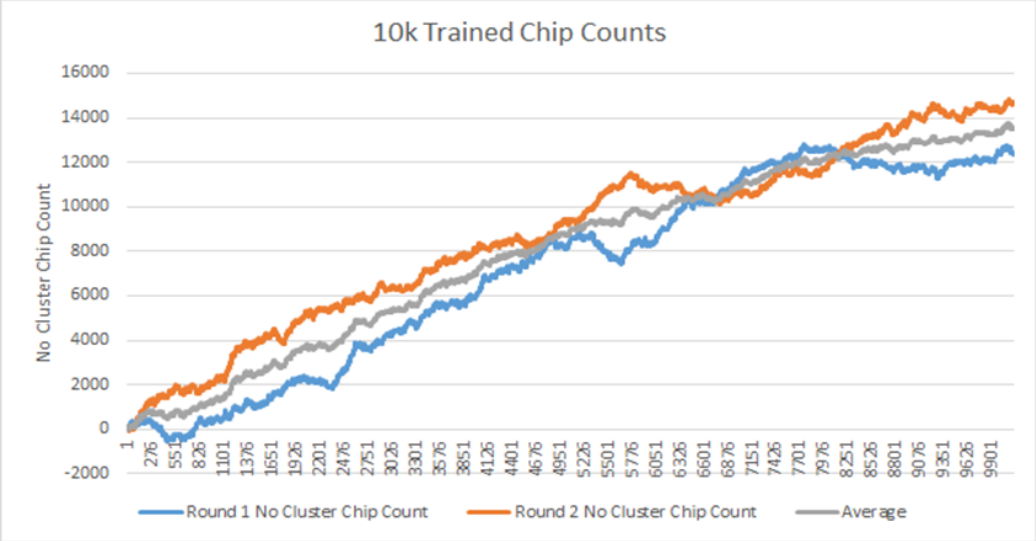


Figure 18. The above shows the chip count of the Control (No Clustering) and Experimental (Waveform Clustering) players averaged over all rounds played with 10K iterations of training as a starting point for the memory banks of the players. The Control player consistently wins at this training slice with a margin of approximately 13,000 chips.

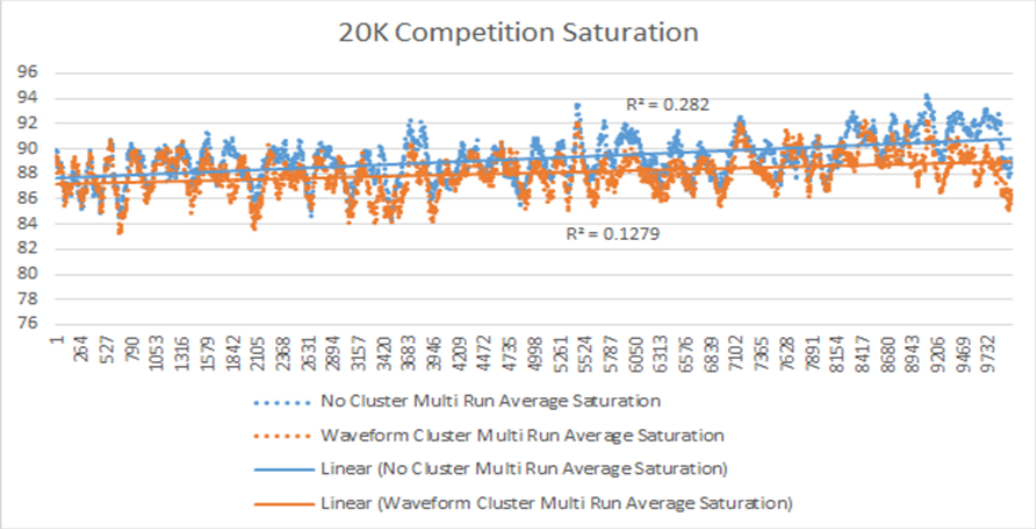


Figure 19. The above shows the Saturation of the Control (No Clustering) and Experimental (Waveform Clustering) players averaged over all rounds played with 20K iterations of training as a starting point for the memory banks of the players. The Control player consistently has higher Saturation values, suggesting that it has converged on a strategy faster during training than the Experimental player.

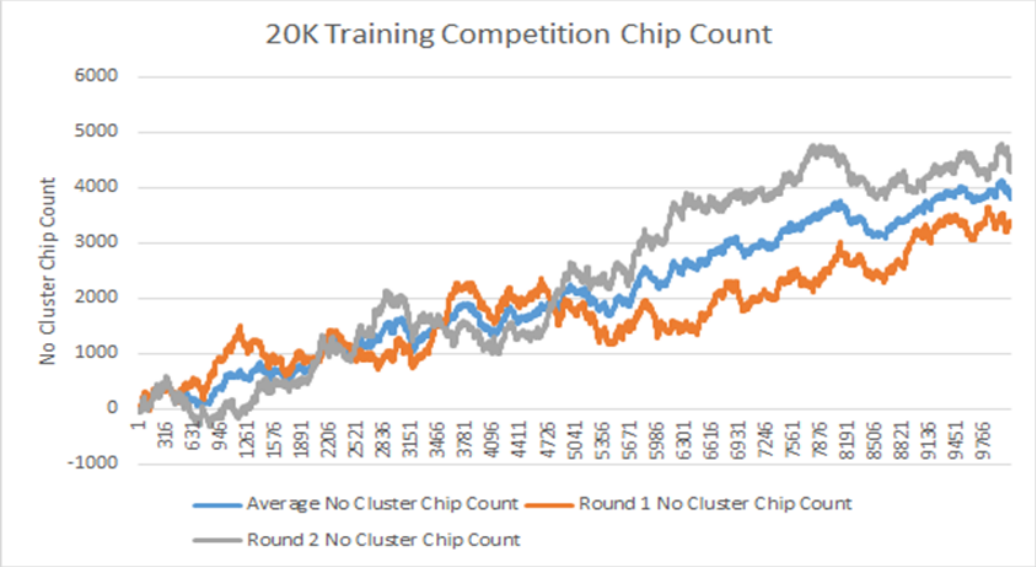


Figure 20. The above shows the chip count of the Control (No Clustering) and Experimental (Waveform Clustering) players averaged over all rounds played with 20K iterations of training as a starting point for the memory banks of the players. The Control player consistently wins at this training slice with a margin of approximately 4,000 chips.

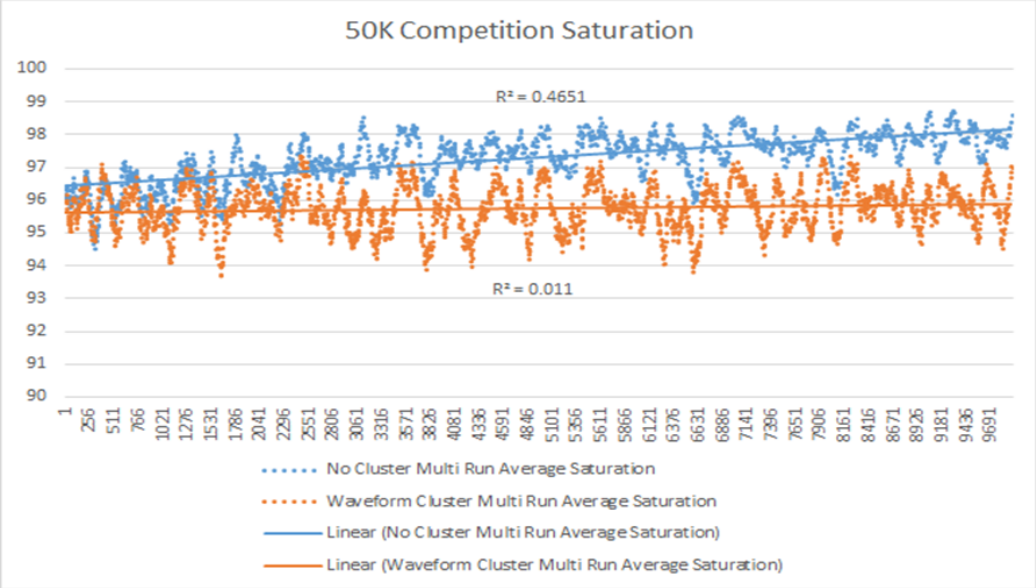


Figure 21. The above shows the Saturation of the Control (No Clustering) and Experimental (Waveform Clustering) players averaged over all rounds played with 50K iterations of training as a starting point for the memory banks of the players. The Control player consistently has higher Saturation values, suggesting that it has converged on a strategy faster during training than the Experimental player.

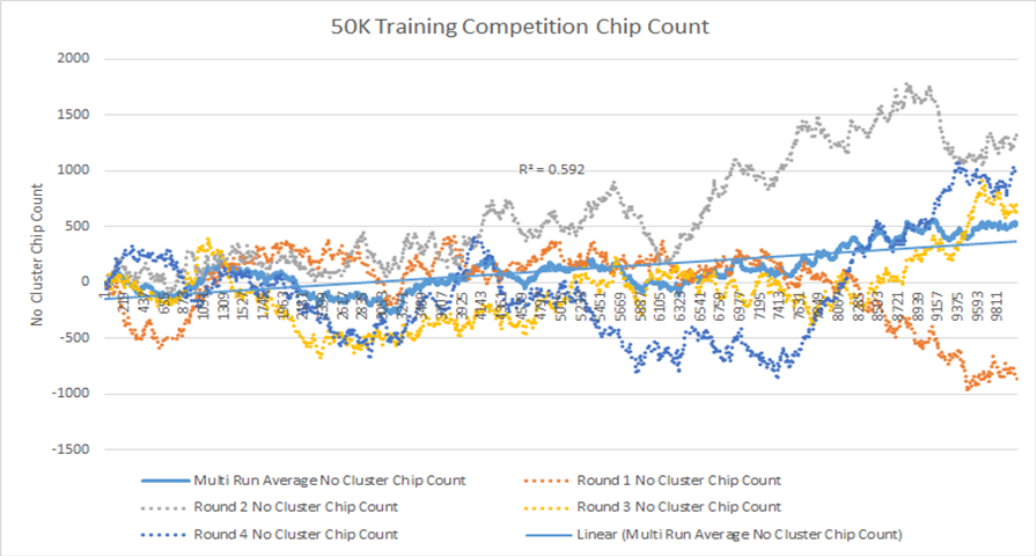


Figure 22. The above shows the chip count of the Control (No Clustering) and Experimental (Waveform Clustering) players averaged over all rounds played with 50K iterations of training as a starting point for the memory banks of the players. The Control player consistently wins at this training slice with a margin of approximately 500 chips.

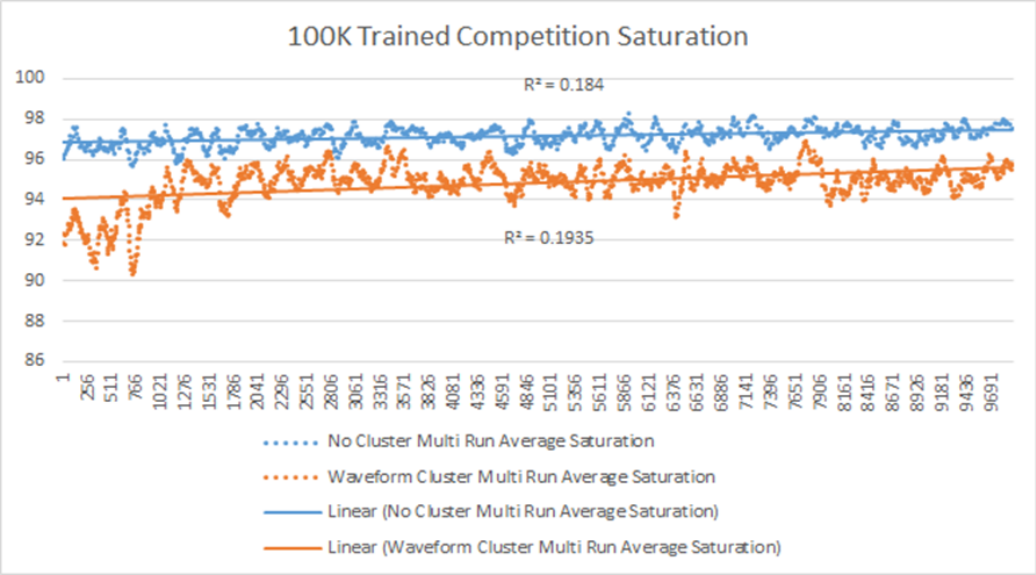


Figure 23. The above shows the Saturation of the Control (No Clustering) and Experimental (Waveform Clustering) players averaged over all rounds played with 100K iterations of training as a starting point for the memory banks of the players. The Control player consistently has higher Saturation values, suggesting that it has converged on a strategy faster during training than the Experimental player.

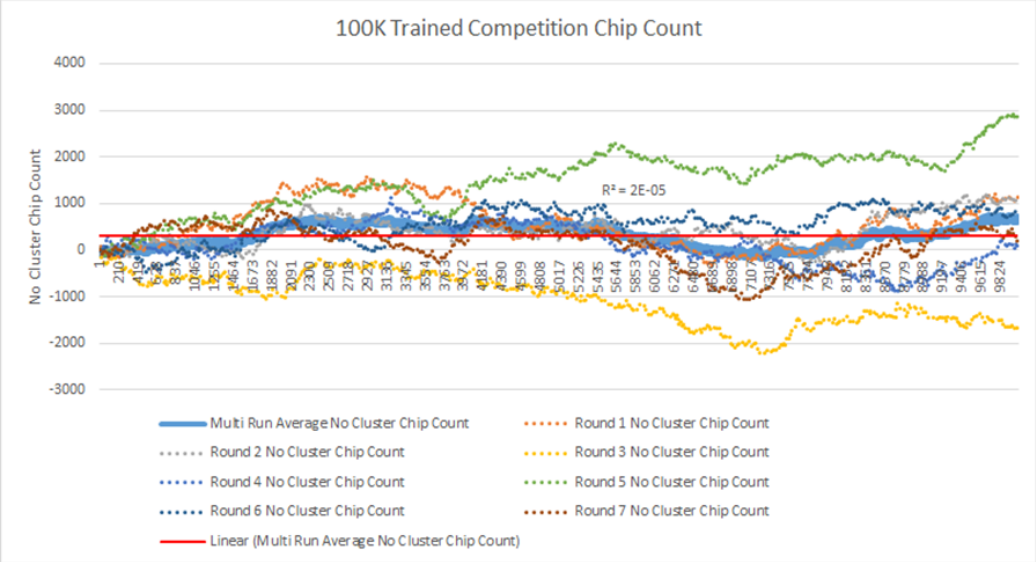


Figure 24. The above shows the chip count of the Control (No Clustering) and Experimental (Waveform Clustering) players averaged over all rounds played with 100K iterations of training as a starting point for the memory banks of the players. The Control player consistently wins at this training slice with a margin of approximately 400 chips.

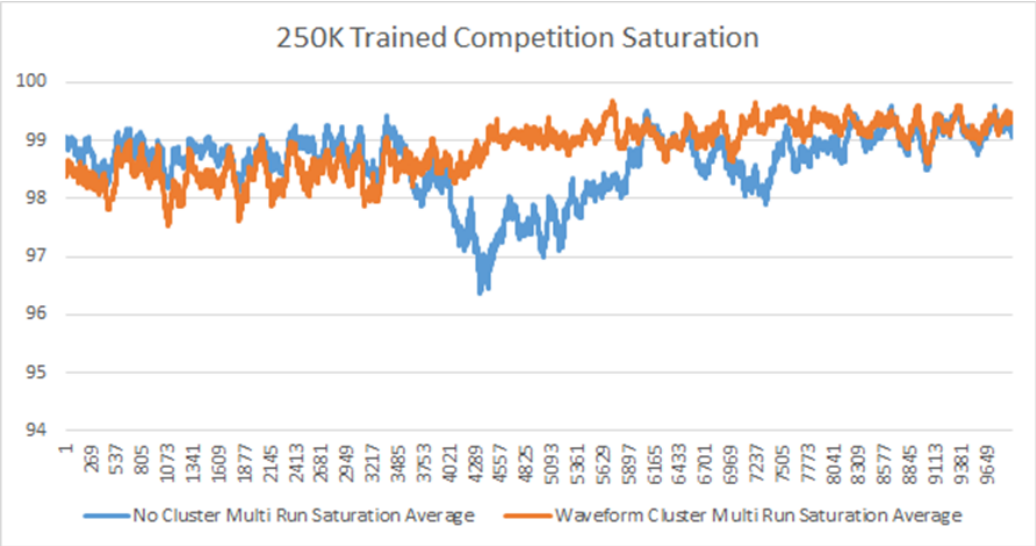


Figure 25. The above shows the Saturation of the Control (No Clustering) and Experimental (Waveform Clustering) players averaged over all rounds played with 250K iterations of training as a starting point for the memory banks of the players. The Control player and Experimental player have approximately the same Saturation through the first 4000 hands, but consistently over many rounds the Control player experiences a sharp drop in Saturation at this point and takes several thousand hands to recover. This means that the Control player consistently is forced into a relatively unfamiliar portion of its game tree while the Experimental player does not experience such a drop in confidence.

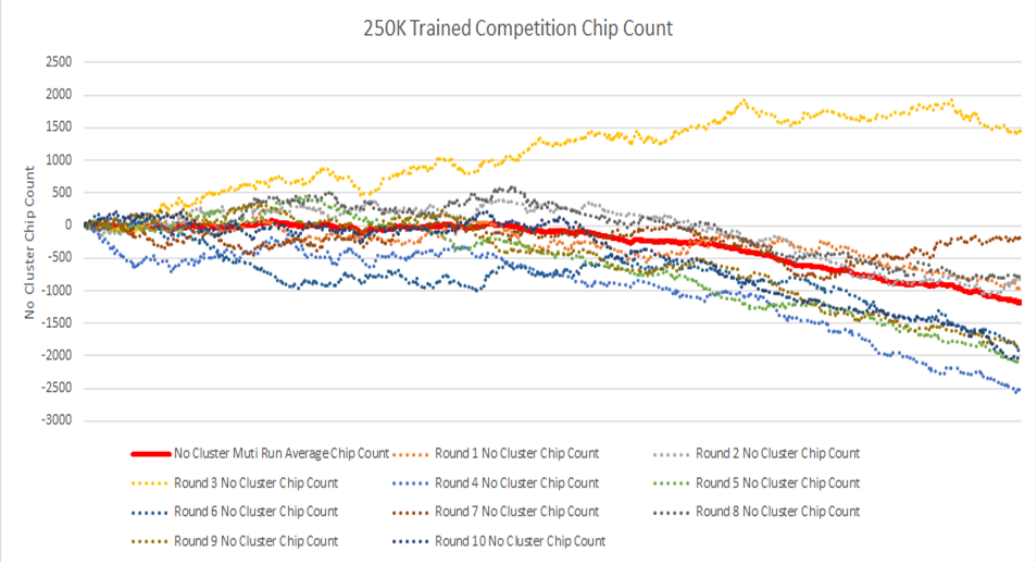


Figure 26. The above shows the chip count of the Control (No Clustering) and Experimental (Waveform Clustering)l players averaged over all rounds played with 250K iterations of training as a starting point for the memory banks of the players. The Experimental player consistently wins with a margin of approximately 1200 chips. Note the correlation between the profitability of the Experimental player and the sharp drop in the Control player's Saturation at around 5000 hands.

1. Conclusions

From the results we can see two trends. First, the Control player has faster initial training, as we can see by consistently higher Saturation values compared to the Experimental player at the early training slices. Second, we see a diminishing advantage for the Control player as the Experimental player closes the Saturation gap. This culminates in an eventual advantage for the Experimental player at higher training values. The drop in Saturation that we consistently see reported by the Control player at the 250K training slice and its correlation to profitable play for the Experimental player suggests that the Experimental player utilizes its tree better than the Control player once trained to maximum Saturation.

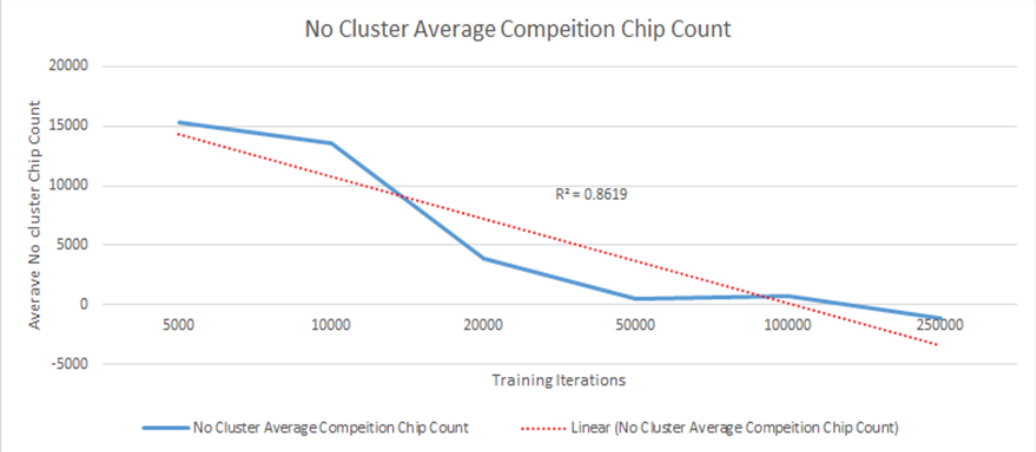


Figure 27. The above shows the trend of chip counts across the various training slices. We see a clear diminishing advantage for the Control player as training progresses, culminating in an advantage for the Experimental player.

CHAPTER 5

Future Work

Future work is called for to further validate these initial results.

- The CFR implementation shared by the players should be adjusted to set the bar for Saturation (which guides simulated annealing of the utility values) higher and lower.
- The number of buckets used in the implementation should be extended to 10, 100, and 1000 bucket implementation, resulting in exponentially larger Extensive Forms for the CFR implementation to train over and potentially revealing a different outcome in the context of a very large training space.
- We would suggest expanding the formats of Poker used for play (i.e. more players, tournament playing formats, pot-limit or no-limit playing formats).
- To organize the results in a paper that can be submitted for publication to a suitable venue.

REFERENCES

[1] Darse Billings, Neil Burch, Aaron Davidson, Robert C. Holte, Jonathan Schaeffer, Terence Schauenberg, and Duane Szafron. Approximating game-theoretic optimal strategies for full-scale poker. In Georg Gottlob and Toby Walsh, editors, *IJCAI*, pages 661–668. Morgan Kaufmann, 2003.

[2] Darse Billings, Aaron Davidson, Jonathan Schaeffer, and Duane Szafron. The challenge of poker. *Artificial Intelligence*, 134:2002, 2001.

[3] Darse Billings, Denis Papp, Jonathan Schaeffer, and Duane Szafron. Using selective-sampling simulations in poker, 1999.

[4] Cameron Browne, Edward J. Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Trans. Comput. Intellig. and AI in Games*, 4(1):1–43, 2012.

[5] Aaron Davidson, Darse Billings, Jonathan Schaeffer, and Duane Szafron. Improved opponent modeling in poker. pages 493–499. AAAI Press, 2000.

[6] Forum Discussion. Forum discussion on check raise strategy used for big

draws post flop in limit texas hold'em. http://www.cardschat.com/f57/limit-holdem-check-raising-179968.

[7] Forum Discussion. Participant algorithms of the 2012 international computer poker competition. http://www.computerpokercompetition.org/index.php/competitions/participants/91-participants-2012?showall=1&limitstart=.

[8] Forum Discussion. Participant algorithms of the 2013 international computer poker competition. http://www.computerpokercompetition.org/index.php/competitions/participants/93-participants-2013?showall=&start=1.

[9] Forum Discussion. Two plus two hand evaluation. http://archives1.twoplustwo.com/showflat.php?Cat=0&Number=8513906&page=0&fpart=1&vc=1.

[10] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. Building Watson: An overview of the DeepQA project. *AI Magazine*, 31(3):59–79, 2010.

[11] David Ferrucci, Anthony Levas, Sugato Bagchi, David Gondek, and Erik T. Mueller. Watson: Beyond Jeopardy! *Artificial Intelligence*, 199-200:93–105, June 2013.

[12] Andrew Gilpin, Javier Pea, Samid Hoda, and Tuomas Sandholm. Gradient-based

algorithms for finding nash equilibria in extensive form games. In *In Proceedings of the Eighteenth International Conference on Game Theory*, 2007.

[13] Andrew Gilpin and Tuomas Sandholm. Finding equilibria in large sequential games of imperfect information. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, EC '06, pages 160–169, New York, NY, USA, 2006. ACM.

[14] 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.

[15] Geoffrey J. Gordon. No-regret algorithms for online convex programs. In Bernhard Schlkopf, John Platt, and Thomas Hoffman, editors, *NIPS*, pages 489–496. MIT Press, 2006.

[16] The Guardian. Chips are down as man beats poker machine. `http://www.theguardian.com/technology/2007/jul/27/gambling?gusrc=rss&feed=technology`.

[17] Feng-Hsiung Hsu. *Behind Deep Blue: Building the Computer That Defeated the World Chess Champion*. Princeton University Press, Princeton, NJ, USA, 2002.

[18] Michael Johanson, Nolan Bard, Marc Lanctot, Richard Gibson, and Michael Bowling. Efficient nash equilibrium approximation through monte carlo counterfactual regret minimization. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*,

AAMAS '12, pages 837–846, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[19] Michael Johanson, Neil Burch, Richard Valenzano, and Michael Bowling. Evaluating state-space abstractions in extensive-form games. In *Proceedings of the Twelfth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2013.

[20] Michael Kearns, Michael L. Littman, and Satinder Singh. Graphical models for game theory, 2001.

[21] Daphne Koller and Nimrod Megiddo. The complexity of two-person zero-sum games in extensive form. *Games and Economic Behavior*, 4:528–552, 1990.

[22] Kevin B. Korb, Ann E. Nicholson, and Nathalie Jitnah. Bayesian poker. In *In Uncertainty in Artificial Intelligence*, pages 343–350. Morgan Kaufman, 1999.

[23] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte carlo sampling for regret minimization in extensive games. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1078–1086, 2009.

[24] Alessandro Lazaric, Enrique Munoz de Cote, Nicola Gatti, and Marcello Restelli. Reinforcement learning in extensive form games with incomplete information: the bargaining case study. In Edmund H. Durfee, Makoto Yokoo, Michael N. Huhns, and Onn Shehory, editors, *Proceedings of the International Joint*

Conference on Autonomous Agents and Multi Agent Systems (AAMAS). ACM Press, 2007.

[25] D.N.L. Levy and M. Newborn. *How Computers Play Chess*. Computer Science Press, 1991.

[26] John F. Nash, Jr. Equilibrium Points in n-Person Games. *Proceedings of the National Academy of Sciences USA*, 36:48–49, 1950.

[27] John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[28] Denis Richard Papp, Duane Szafron, and Oliver Schulte. Dealing with imperfect information in poker, 1998.

[29] Jiefu Shi and MichaelL. Littman. Abstraction methods for game theoretic poker. In Tony Marsland and Ian Frank, editors, *Computers and Games*, volume 2063 of *Lecture Notes in Computer Science*, pages 333–345. Springer Berlin Heidelberg, 2001.

[30] Ken Takusagawa. Nash equilibrium of Texas Hold'em poker, 2000.

[31] M. Zinkevich and M. Littman. The aaai computer poker compeition. *Journal of the International Computer Games Association*, 2006.

[32] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret Minimization in Games with Incomplete Information. *Advances in Neural Information Processing Systems 20 (NIPS)*, 2007.