Chapter 3

Game-Theoretic Methods

Approximating Game-Theoretic Optimal Strategies for Full-scale Poker

3.1 Introduction

Mathematical game theory was introduced by John von Neumann in the 1940s, and has since become one of the foundations of modern economics [14]. Von Neumann used the game of poker as a basic model for 2-player zero-sum adversarial games, and proved the first fundamental result, the famous Minimax Theorem. A few years later, John Nash added results for $N$-player non-cooperative games, for which he later won the Bank of Sweden Prize in Economic Sciences in Memory of Alfred Nobel [8]. Many decision problems can be modeled using game theory, and it has been employed in a wide variety of domains in recent years.

Of particular interest is the existence of optimal solutions, or Nash equilibria.

---


2 The term “optimal” is over-loaded in computer science, and is highly misleading (overly flatter- ing) in this particular context. The more neutral terms “equilibrium strategy” or “Nash equilibrium” are now preferred. An equilibrium strategy is optimal only in the sense of not being exploitable by a perfect opponent; but since it fails to exploit imperfect opponents, it can perform much worse than a maximal strategy in practice. The term “equilibrium” is used in several places where “optimal” appeared in the original publication. However, the term “pseudo-optimal” has been retained.
An equilibrium solution provides a \textit{randomized mixed strategy}, which is basically a recipe of how to play in each possible situation. Using this strategy ensures that an agent will obtain at least the game-theoretic value of the game, regardless of the opponent’s strategy. Unfortunately, finding exact equilibrium solutions is limited to relatively small problem sizes, and is not practical for most real domains.

This paper explores the use of highly abstracted mathematical models which capture the most essential properties of the real domain, such that an exact solution to the smaller problem provides a useful approximation of an equilibrium strategy for the real domain. The application domain used is Limit Texas Hold’em.

Due to the computational limitations involved, only simplified poker variations have been solved in the past (\textit{e.g.}, \cite{7, 9}). While these are of theoretical interest, the same methods are not feasible for real games, which are too large by many orders of magnitude \cite{6}.

Selby \cite{10} computed an equilibrium solution for the abbreviated game of \textit{Pre-flop Hold’em}. Shi and Littman \cite{11} investigated abstraction techniques to reduce the large search space and complexity of the problem, using a simplified variant of poker. Takusagawa \cite{13} created approximate equilibrium strategies for the play of three specific Hold’em flops and betting sequences.

Using new abstraction techniques, we have produced viable “pseudo-optimal” strategies for the game of 2-player Texas Hold’em. The resulting poker-playing programs have demonstrated a tremendous improvement in performance. Whereas the previous best poker programs were easily beaten by any competent human player, the new programs are capable of defeating very strong players, and can hold their own against world-class opposition.

Although some domain-specific knowledge is an asset when creating accurate reduced-scale models, analogous methods can be developed for many other imperfect information domains and generalized game trees. We describe a general method of problem reformulation that permits the independent solution of sub-trees by estimating the conditional probabilities needed as input for each computation.

This paper makes the following contributions:

1. Abstraction techniques that can reduce a $\Theta(10^{18})$ poker search space to a
manageable $\Theta(10^7)$, without losing the most important properties of the game.

2. A poker-playing program that is a major improvement over previous efforts, and is capable of competing with world-class opposition.

### 3.2 Game Theory

Game theory encompasses all forms of competition between two or more agents. Unlike chess or checkers, poker is a game of imperfect information and chance outcomes. It can be represented with an imperfect information game tree having chance nodes and decision nodes, which are grouped into information sets.

Since the nodes in this game tree are not independent, divide-and-conquer methods for computing sub-trees (such as the alpha-beta search algorithm) are not applicable. More detailed descriptions of imperfect information game tree structure are available elsewhere (e.g., [4]).

A strategy is a set of rules for choosing an action at every decision node of the game tree. In general, this will be a randomized mixed strategy, which is a probability distribution over the various alternatives. A player must use the same policy across all nodes in the same information set, since from that player’s perspective they are indistinguishable from each other (differing only in the hidden information component).

The conventional method for solving such a problem is to convert the descriptive representation, or extensive form, into a system of linear equations, which is then solved by a linear programming (LP) system such as the Simplex algorithm. The equilibrium solutions are computed simultaneously for all players, ensuring the best worst-case outcome for each player.

Traditionally, the conversion to normal form was accompanied by an exponential blow-up in the size of the problem, meaning that only very small problem instances could be solved in practice. Koller [5] described an alternate LP representation, called sequence form, which exploits the property of perfect recall (wherein all players know the preceding history of the game), to obtain a system of equations and unknowns that is only linear in the size of the game tree. This exponential
reduction in representation has re-opened the possibility of using game-theoretic analysis for many domains. However, since the game tree itself can be very large, the LP solution method is still limited to moderate problem sizes (normally less than a billion nodes).

3.3 Texas Hold’em

A game (or hand) of Texas Hold’em consists of four stages, each followed by a round of betting:

1. **Pre-flop**: Each player is dealt two private cards face down (the hole cards).

2. **Flop**: Three community cards (shared by all players) are dealt face up.

3. **Turn**: A single community card is dealt face up.

4. **River**: A final community card is dealt face up.

After the betting, all active players reveal their hole cards for the showdown. The player with the best five-card poker hand formed from their two private cards and the five public cards wins all the money wagered (ties are possible).

The game starts off with two forced bets (the blinds) put into the pot. When it is a player’s turn to act, they must either **bet/raise** (increase their investment in the pot), **check/call** (match what the opponent has bet or raised), or **fold** (quit and surrender all money contributed to the pot).

The best-known non-commercial Texas Hold’em program is POKI. It has been playing online since 1997 and has earned an impressive winning record, albeit against generally weak opposition [3]. The system’s abilities are based on enumeration and simulation techniques, expert knowledge, and opponent modeling. The program’s weaknesses are easily exploited by strong players, especially in the 2-player game.
3.4 Abstractions

Texas Hold’em has an easily identifiable structure, alternating between chance nodes and betting rounds in four distinct stages. A high-level view of the imperfect information game tree is shown in Figure 3.1.

Hold’em can be reformulated to produce similar but much smaller games. The objective is to reduce the scale of the problem without severely altering the fundamental structure of the game, or the resulting equilibrium strategies. There are many ways of doing this, varying in the overall reduction and in the accuracy of the resulting approximation.

Some of the most accurate abstractions include suit equivalence isomorphisms (offering a reduction of at most a factor of $4! = 24$), rank equivalence (only under certain conditions), and rank near-equivalence. The equilibrium solutions to these abstracted problems will either be exactly the same or will have a small bounded error, which we refer to as near-optimal solutions. Unfortunately, the abstractions that produce an exact or near-exact reformulation do not produce the very large
reductions required to make full-scale poker tractable.

A common method for controlling the game size is *deck reduction*. Using less than the standard 52-card deck greatly reduces the branching factor at chance nodes. Other methods include reducing the number of cards in a player’s hand (e.g., from a 2-card hand to a 1-card hand), and reducing the number of board cards (e.g., a 1-card flop), as was done by Shi and Littman [11] for the game of Rhode Island Hold’em.\(^3\) Koller and Pfeffer [6] used such parameters to generate a wide variety of tractable games to solve with their Gala system.

We have used a number of small and intermediate sized games, ranging from eight cards (two suits, four ranks) to 24 cards (three suits, eight ranks) for the purpose of studying abstraction methods, comparing the results with known exact or near-optimal solutions. However, these smaller games are not suitable for use as an approximation for Texas Hold’em, as the underlying structures of the games are different. To produce good playing strategies for full-scale poker, we look for abstractions of the real game which do not alter that basic structure.

The abstraction techniques used in practice are powerful in terms of reducing the problem size, and subsume those previously mentioned. However, since they are also much cruder, we call their solutions *pseudo-optimal*, to emphasize that there is no guarantee that the resulting approximations will be accurate, or even reasonable. Some will be low-risk propositions, while others will require empirical testing to determine if they have merit.

### 3.4.1 Betting Round Reduction

The standard rules of Limit Hold’em allow for a maximum of four bets per player per round.\(^4\) Thus, in 2-player Limit poker there are 19 possible betting sequences, of which two do not occur in practice.\(^5\) Of the remaining 17 sequences, 8 end in a fold (leading to a terminal node in the game tree), and 9 end in a call (carrying forward

\(^3\) Recently, Gilpin and Sandholm introduced an automated abstraction technique called *gameshrink*, which was used to solve the game of Rhode Island Hold’em (see Chapter 6).

\(^4\) Some rules allow unlimited raises when only two players are involved. However, occasions with more than three legitimate raises are rare, and do not greatly alter an equilibrium strategy.

\(^5\) Technically, a player may fold even though there is no outstanding bet. This is logically dominated, and therefore does not occur in an equilibrium strategy, and is seldom seen in practice.
to the next chance node). Using $k$ = check, $b$ = bet, $f$ = fold, $c$ = call, $r$ = raise, and capital letters for the second player, the tree of possible betting sequences for each round is:

$kK$ $kBf$ $kBc$ $kBrF$ $kBrC$ $kBrRf$ $kBrRc$ $kBrRrF$ $kBrRrC$

$bF$ $bC$ $bRf$ $bRc$ $bRrF$ $bRrC$ $bRrRf$ $bRrRc$

We call this local collection of decision nodes a *betting tree*, and represent it diagramatically with a triangle (see Chapter 1 Figure 1.2).

With *betting round reduction*, each player is allowed a maximum of three bets per round, thereby eliminating the last two sequences in each line. The effective branching factor of the betting tree is reduced from nine to seven. This does not appear to have a substantial effect on play, or on the *expected value* (EV) for each player. This observation has been verified experimentally. In contrast, we computed the corresponding *post-flop models* with a maximum of two bets per player per round, and found radical changes to the equilibrium strategies, strongly suggesting that that level of abstraction is not safe.

### 3.4.2 Elimination of Betting Rounds

Large reductions in the size of a poker game tree can be obtained by *elimination of betting rounds*. There are several ways to do this, and they generally have a significant impact on the nature of the game. First, the game may be *truncated*, by eliminating the last round or rounds. In Hold’em, ignoring the last board card and the final betting round produces a 3-round model of the actual 4-round game. The solution to the 3-round model loses some of the subtlety involved in the true equilibrium strategy, but the degradation applies primarily to advanced tactics on the turn. There is a smaller effect on the flop strategy, and the strategy for the first betting round may have no significant changes, since it incorporates all the outcomes of two future betting rounds. We use this particular abstraction to define an appropriate strategy for play in the first round, and thus call it a *pre-flop model* (see Figure 3.2).

The effect of truncation can be lessened through the use of *expected value leaf nodes*. Instead of ending the game abruptly and awarding the pot to the strongest
hand at that moment, we compute an average conclusion over all possible chance outcomes. For a 3-round model ending on the turn, we roll-out all 44 possible river cards, assuming no further betting (or alternately, assuming one bet per player for the last round). Each player is awarded a fraction of the pot, corresponding to the probability of winning the game. In a 2-round pre-flop model, we roll-out all 990 2-card combinations of the turn and river.

The most extreme form of truncation results in a 1-round model, with no foresight of future betting rounds. Since each future round provides a refinement to the approximation, this will not reflect a correct strategy for the real game. In particular, betting plans that extend over more than one round, such as deferring the raise of a very strong hand, are lost entirely. Nevertheless, even these simplistic models can be useful when combined with expected value leaf nodes.

Alex Selby computed an equilibrium solution for the game of Pre-flop Hold’em, which consists of only the first betting round followed by an EV roll-out of the five board cards to determine the winner [10]. Although there are some serious limitations in the strategy based on this 1-round model, we have incorporated the Selby pre-flop system into one of our programs, PSOPT11, as described later in this section.

In contrast to truncating rounds, we can bypass certain early stages of the game. We frequently use post-flop models, which ignore the pre-flop betting round, and use a single fixed flop of three cards (see Figure 3.2).

It is natural to consider the idea of independent betting rounds, where each phase of the game is treated in isolation. Unfortunately, the betting history from previous rounds will almost always contain contextual information that is critical to making appropriate decisions. The probability distribution over the hands for each player is strongly dependent on the path that led to that decision point, so it cannot be ignored without risking a considerable loss of information. However, the naive independence assumption can be viable in certain circumstances, and we do implicitly use it in the design of PSOPT11 to bridge the gap between the 1-round pre-flop model and the 3-round post-flop model.

Another possible abstraction we explored was merging two or more rounds into
a single round, such as creating a combined 2-card turn/river. However, it is not clear what the appropriate bet size should be for this composite round. In any case, the solutions for these models (over a full range of possible bet sizes), all turned out to be substantially different from their 3-round counterparts, and the method was therefore rejected.

3.4.3 Composition of Pre-flop and Post-flop models

Although the nodes of an imperfect information game tree are not independent in general, some decomposition is possible. For example, the sub-trees resulting from different pre-flop betting sequences can no longer have nodes that belong to the same information set. The sub-trees for our post-flop models can be computed in isolation, provided that the appropriate preconditions are given as input. Unfortunately, knowing the correct conditional probabilities would normally entail solving the whole game, so there would be no advantage to the decomposition.

For simple post-flop models, we dispense with the prior probabilities. For the post-flop models used in PSOPT10 and PSOPT11, we simply ignore the implications of the pre-flop betting actions, and assume a uniform distribution over all possible hands for each player. Different post-flop solutions were computed for initial pot sizes of two, four, six, and eight bets (corresponding to pre-flop sequences with zero, one, two, or three raises, but ignoring which player initially made each raise). In PSOPT11, the four post-flop solutions are simply appended to the Selby pre-flop strategy (Figure 3.2). Although these simplifying assumptions are technically wrong, the resulting play is still surprisingly effective.

A better way to compose post-flop models is to estimate a set of conditional probabilities, using the solution to a pre-flop model. With a tractable pre-flop model, we have a means of estimating an appropriate strategy at the root, and thereby determine the consequent probability distributions.

In PSOPT12, a 3-round pre-flop model was designed and solved. The resulting

---

6 To see this, each decision node of the game tree can be labeled with all the cards known to that player, and the full path that led to that node. Nodes with identical labels differ only in the hidden information, and are therefore in the same information set. Since the betting history is different for these sub-trees, none of the nodes are inter-dependent.
pseudo-optimal strategy for the pre-flop (which was significantly different from the Selby strategy) was used to determine the corresponding distribution of hands for each player in each context. This provided the necessary input parameters for each of the seven pre-flop betting sequences that carry over to the flop stage. Since each of these post-flop models has been given (an approximation of) the perfect recall knowledge of the full game, they are fully compatible with each other, and are properly integrated under the umbrella of the pre-flop model (Figure 3.2). In theory, this should be equivalent to computing the much larger tree, but it is limited by the accuracy and appropriateness of the proposed pre-flop betting model.\(^7\)

\(^7\) Actually, this is only true if there is a *unique* equilibrium strategy, whereas most interesting games have an *infinite* number of equilibria. This caveat has serious implications on the theoretical and practical value of this approach. Please see Chapter Endnote 3.7.1 for more information on this important limitation.
3.4.4 Abstraction by Bucketing

The most important method of abstraction for the computation of our pseudo-optimal strategies is called *bucketing*. This is an extension of the natural and intuitive concept that has been applied many times in previous research (*e.g.*, [12, 13, 11]). The set of all possible hands is partitioned into equivalence classes (also called *buckets* or *bins*). A *many-to-one mapping function* determines which hands will be grouped together. Ideally, the hands should be grouped according to *strategic similarity*, meaning that they can all be played in a similar manner without much loss in EV.

If every hand was played with a particular *pure strategy* (*i.e.*, only one of the available choices), then a perfect mapping function would group all hands that follow the same plan, and 17 equivalence classes for each player would be sufficient for each betting round. However, since a *mixed strategy* may be indicated for equilibrium play in some cases, we would like to group hands that have a similar probability distribution over action plans.

One obvious but rather crude bucketing function is to group all hands according to strength (*i.e.*, its *rank* with respect to all possible hands, or the probability of currently being in the lead). This can be improved by considering the roll-out of all future cards, giving an (unweighted) estimate of the chance of winning the game.

However, this is only a one-dimensional view of hand types, in what can be considered to be an $N$-dimensional space of strategies, with a vast number of different ways to classify them. A superior practical method would be to project the set of all hands onto a two-dimensional space consisting of (roll-out) hand strength and hand potential (similar to the hand assessment used in POKI [3]). Clusters in the resulting scattergram suggest reasonable groups of hands to be treated similarly.

We eventually settled on a simple compromise. With $n$ available buckets, we allocate $n - 1$ to *roll-out hand strength*. The number of hand types in each class is not uniform; the classes for the strongest hands are smaller than those for mediocre and weak hands, allowing for better discrimination of the smaller fractions of hands that should be raised or re-raised.

One special bucket is designated for hands that are low in strength but have *high...*
potential, such as good draws to a flush or straight. This plays an important role in identifying good hands to use for bluffing (known as semi-bluffs [12]). Comparing post-flop solutions that use six strength buckets to solutions with five strength plus one high-potential bucket, we see that most bluffs in the latter are taken from the special bucket, which is sometimes played in the same way as the strongest bucket. This confirmed our expectations that the high-potential bucket would improve the selection of hands for various betting tactics, and increase the overall EV.

The number of buckets that can be used in conjunction with a 3-round model is very small, typically six or seven for each player (i.e., 36 or 49 pairs of bucket assignments). Obviously this results in a very coarse-grained abstract game, but it may not be substantially different from the number of distinctions an average human player might make. Regardless, it is the best we can currently do given the computational constraints of this approach.

The final requirement to sever the abstract game from the underlying real game tree are the transition probabilities. The chance node between the flop and turn represents a particular card being dealt from the remaining stock of 45 cards. In the abstract game, there are no cards, only buckets. The effect of the turn card in the abstract game is to dictate the probability of moving from one pair of buckets on the flop to any pair of buckets on the turn. Thus, the collection of chance nodes in the game tree is represented by an \((n \times n)\) to \((n \times n)\) transition network as shown in Figure 3.3. For post-flop models, this can be estimated by walking the entire tree, enumerating all transitions for a small number of characteristic flops. For pre-flop models, the full enumeration is more expensive (encompassing all \(\binom{48}{3} = 17296\) possible flops), so it is estimated either by sampling, or by (parallel) enumeration of a truncated tree.

For a 3-round post-flop model, we can comfortably solve abstract games with up to seven buckets for each player in each round. Changing the distribution of buckets, such as six for the flop, seven for the turn, and eight for the river, does not appear to significantly affect the quality of the solutions, better or worse.

The final linear programming solution produces a large table of mixed strategies (probabilities for fold, call, or raise) for every reachable scenario in the abstract
game. To use this, the poker-playing program looks for the corresponding situation based on the same hand strength and potential measures, and randomly selects an action from the mixed strategy.

The large LP computations typically take less than a day (using CPLEX with the barrier method), and use up to two Gigabytes of RAM. Larger problems will exceed available memory, which is common for large LP systems. Certain LP techniques such as constraint generation could potentially extend the range of solvable instances considerably, but this would probably only allow the use of one or two additional buckets per player.

3.5 Experiments

3.5.1 Testing Against Computer Players

A series of matches between computer programs was conducted, with the results shown in Table 3.1. Win rates are measured in small bets per game (sb/g). Each match was run for at least 20,000 games (and over 100,000 games in some cases). The variance per game depends greatly on the styles of the two players involved, but is typically +/- 6 sb. The standard deviation for each match outcome is not shown, but is normally less than +/- 0.03 sb/g.

The “bot players” were:

**PSoPT12**, composed of a hand-crafted pre-flop model, which was solved by linear programming to provide conditional probability distributions to each of seven
<table>
<thead>
<tr>
<th>Program</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 PsOpti1</td>
<td>+0.090</td>
<td>+0.091</td>
<td>+0.251</td>
<td>+0.156</td>
<td>+0.047</td>
<td>+0.546</td>
<td>+0.635</td>
<td></td>
</tr>
<tr>
<td>2 PsOpti2</td>
<td>-0.090</td>
<td>+0.069</td>
<td>+0.118</td>
<td>+0.054</td>
<td>+0.045</td>
<td>+0.505</td>
<td>+0.319</td>
<td></td>
</tr>
<tr>
<td>3 PsOpti0</td>
<td>-0.091</td>
<td>-0.069</td>
<td>+0.163</td>
<td>+0.135</td>
<td>+0.001</td>
<td>+0.418</td>
<td>+0.118</td>
<td></td>
</tr>
<tr>
<td>4 Aadapti</td>
<td>-0.251</td>
<td>-0.118</td>
<td>-0.163</td>
<td>+0.178</td>
<td>+0.550</td>
<td>+0.905</td>
<td>+2.615</td>
<td></td>
</tr>
<tr>
<td>5 Anti-Poki</td>
<td>-0.156</td>
<td>-0.054</td>
<td>-0.135</td>
<td>-0.178</td>
<td>+0.385</td>
<td>+0.143</td>
<td>+0.541</td>
<td></td>
</tr>
<tr>
<td>6 Poki</td>
<td>-0.047</td>
<td>-0.045</td>
<td>-0.001</td>
<td>-0.550</td>
<td>-0.385</td>
<td>+0.537</td>
<td>+2.285</td>
<td></td>
</tr>
<tr>
<td>7 Always Call</td>
<td>-0.546</td>
<td>-0.505</td>
<td>-0.418</td>
<td>-0.905</td>
<td>-0.143</td>
<td>-0.537</td>
<td>=0.000</td>
<td></td>
</tr>
<tr>
<td>8 Always Raise</td>
<td>-0.635</td>
<td>-0.319</td>
<td>-0.118</td>
<td>-2.615</td>
<td>-0.541</td>
<td>-2.285</td>
<td>=0.000</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: Computer vs computer matches (small bets per game).

3-round post-flop models (Figure 3.2). All models in this prototype used six buckets per player per round.

**PsOpti1**, composed of four 3-round post-flop models under the naive uniform distribution assumption, with seven buckets per player per round. Selby’s equilibrium solution for *Pre-flop Hold’em* is used to play the pre-flop [10].

**PsOpti0**, composed of a single 3-round post-flop model, wrongly assuming uniform distributions and an initial pot size of two bets, with seven buckets per player per round. This program used an always-call policy for the pre-flop betting round.

**Poki**, the University of Alberta poker program. This older version of **Poki** was not designed to play the 2-player game, and can be defeated rather easily, but is a useful benchmark.

**Anti-Poki**, a rule-based program designed to beat **Poki** by exploiting its weaknesses and vulnerabilities in the 2-player game. Any specific counter-strategy can be even more vulnerable to exploitation by adaptive players.

**AAdapti**, a relatively simple adaptive player, capable of slowly learning and exploiting persistent patterns in play.

**Always_Call**, a very weak benchmark strategy.

**Always_Raise**, a very weak benchmark strategy.

It is important to understand that a game-theoretic equilibrium player is, in principle, *not designed to win*. Its purpose is to *not lose*. An implicit assumption is that the opponent is also playing an equilibrium, and nothing can be gained by observing
the opponent for patterns or weaknesses.

In a simple game like RoShamBo (also known as Rock-Paper-Scissors), playing the equilibrium strategy ensures a break-even result, regardless of what the opponent does, and is therefore insufficient to defeat weak opponents, or to win a tournament ([2, 1]). Poker is more complex, and in theory an equilibrium player can win, but only if the opponent makes dominated errors. Any time a player makes any choice that is part of a randomized mixed strategy of any game-theoretic equilibrium policy, that decision is not dominated. In other words, it is possible to play in a highly sub-optimal manner, but still break even against an equilibrium player, because those choices are not strictly dominated.

Since the pseudo-optimal strategies do no opponent modeling, there is no guarantee that they will be especially effective against very bad or highly predictable players. They must rely only on these fundamental strategic errors, and the margin of victory might be relatively modest as a result.

The critical question is whether such errors are common in practice. There is no definitive answer to this question yet, but preliminary evidence suggests that dominated errors occur often enough to gain a measurable EV advantage over weaker players, but may not be very common in the play of very good players.

The first tests of the pseudo-optimal solutions were done with PSOPTI0 playing Post-flop Hold'em, where both players agree to simply call in the pre-flop (thereby matching the exact pre-conditions for the post-flop solution). In those preliminary tests, the author played more than 2000 games, and was unable to defeat the pseudo-optimal strategy. In contrast, POKI had been beaten consistently at a rate of over 0.8 sb/g (which is more than would be lost by simply folding every hand).

Using the same no-bet pre-flop policy, PSOPTI0 was able to defeat POKI at a rate of +0.144 sb/g (compared to +0.001 sb/g for the full game including pre-flop), and defeated AADAPTI at +0.410 sb/g (compared to +0.163 sb/g for the full game).

All of the pseudo-optimal players play substantially better than any previously existing computer programs. Even PSOPTI0, which is not designed to play the full game, earns enough from the post-flop betting rounds to offset the EV losses from the pre-flop round (where it never raises good hands, nor folds bad ones).
<table>
<thead>
<tr>
<th>Player</th>
<th>Games</th>
<th>Posn 1</th>
<th>Posn 2</th>
<th>sb/g</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master-1 early</td>
<td>1147</td>
<td>-0.324</td>
<td>+0.360</td>
<td>+0.017</td>
</tr>
<tr>
<td>Master-1 late</td>
<td>2880</td>
<td>-0.054</td>
<td>+0.396</td>
<td>+0.170</td>
</tr>
<tr>
<td>Experienced-1</td>
<td>803</td>
<td>+0.175</td>
<td>+0.002</td>
<td>+0.088</td>
</tr>
<tr>
<td>Experienced-2</td>
<td>1001</td>
<td>-0.166</td>
<td>-0.168</td>
<td>-0.167</td>
</tr>
<tr>
<td>Experienced-3</td>
<td>1378</td>
<td>+0.119</td>
<td>-0.016</td>
<td>+0.052</td>
</tr>
<tr>
<td>Experienced-4</td>
<td>1086</td>
<td>+0.042</td>
<td>-0.039</td>
<td>+0.002</td>
</tr>
<tr>
<td>Intermediate-1</td>
<td>2448</td>
<td>+0.031</td>
<td>+0.203</td>
<td>+0.117</td>
</tr>
<tr>
<td>Novice-1</td>
<td>1277</td>
<td>-0.159</td>
<td>-0.154</td>
<td>-0.156</td>
</tr>
<tr>
<td>All Opponents</td>
<td>15125</td>
<td></td>
<td></td>
<td>-0.015</td>
</tr>
</tbody>
</table>

Table 3.2: Human vs PSOPT12 matches.

It is suspicious that PSOPT11 outperformed PSOPT12, which in principle should be a better approximation. Subsequent analysis of the play of PSOPT12 revealed some programming errors, and also suggested that the bucket assignments for the pre-flop model were flawed. This may have resulted in an inaccurate pseudo-optimal pre-flop strategy, and consequent imbalances in the prior distributions used as input for the post-flop models. We expect that this will be rectified in future versions, and that the PSOPT12 design will surpass PSOPT11 in performance.  

3.5.2 Testing Against Human Players

While these results are encouraging, none of the non-pseudo-optimal computer opponents are better than intermediate strength at 2-player Texas Hold’em. Therefore, matches were conducted against human opponents.

More than 100 participants volunteered to play against the pseudo-optimal players on our public web applet (www.cs.ualberta.ca/~games/poker/), including many experienced players, a few masters, and one world-class player. The programs provided some fun opposition, and ended up with a winning record overall. The results are summarized in Table 3.2 and Table 3.3.  

---

8 Although the overlayed architecture did overtake PSOPT11 in later incarnations, the side-effects of having a specific pre-flop model are a serious impediment to top-level play, as discussed in Chapter Endnote 3.7.1.

9 Note that the overall averages in these two tables should not be compared directly to each other, as they are based on a different mixture of opponents and different circumstances. For example, some players may have played against PSOPT12 only after gaining experience against PSOPT11, thus playing better.
Table 3.3: Human vs PSOPT11 matches.

Darse Billings, Experienced-1 is Aaron Davidson).

In most cases, the relatively short length of the match leaves a high degree of uncertainty in the outcome, limiting how much can be safely concluded. Nevertheless, some players did appear to have a definite edge, while others were clearly losing.

A number of interesting observations were made over the course of these games. It was obvious that most people had a lot of difficulty learning and adjusting to the computer’s style of play. In poker, knowing the basic approach of the opponent is essential, since it will dictate how to properly handle many situations that arise. Some players wrongly attributed intelligence where none was present. After losing a 1000-game match, one experienced player commented “the bot has me figured out now”, indicating that its opponent model was accurate, when in fact the pseudo-optimal player is oblivious and does no modeling at all.

It was also evident that these programs do considerably better in practice than might be expected, due to the emotional frailty of their human opponents. Many players commented that playing against the pseudo-optimal opponent was an exasperating experience. The bot routinely makes unconventional plays that confuse
and confound humans. Invariably, some of these “bizarre” plays happen to coincide with a lucky escape, and several of these bad beats in quick succession will often cause strong emotional reactions (sometimes referred to as “going on tilt”). The level of play generally goes down sharply in these circumstances.

This suggests that a perfect game-theoretic equilibrium poker player could perhaps beat even the best humans in the long run, because any EV lost in moments of weakness would never be regained. However, the win rate for such a program could still be quite small, giving it only a slight advantage. Thus, it would be unable to exert its superiority convincingly over the short term, such as the few hundred games of one session, or over the course of a world championship tournament. Since even the best human players are known to have biases and weaknesses, opponent modeling will almost certainly be necessary to produce a program that surpasses all human players.

3.5.3 Testing Against a World-class Player

The elite poker expert was Gautam Rao, who is known as “thecount” or “Count-Dracula” in the world of popular online poker rooms. Mr. Rao is the #1 all-time winner in the history of the oldest online game, by an enormous margin over all other players, both in total earnings and in dollar-per-game rate. His particular specialty is in short-handed games with five or fewer players. He is recognized as one of the best players in the world in these games, and is also exceptional at 2-player Hold’em. Like many top-flight players, he has a dynamic ultra-aggressive style.

Mr. Rao agreed to play an exhibition match against PSOPT11, playing more than 7000 games over the course of several days. The graph in Figure 3.4 shows the progression of the match.

The pseudo-optimal player started with some good fortune, but lost at a rate of about $-0.2$ sb/g over the next 2000 games. Then, there was a sudden reversal, following a series of fortuitous outcomes for the program. Although “thecount” is renown for his mental toughness, an uncommon run of bad luck can be very frustrating even for the most experienced players. Mr. Rao believes he played below his best level during that stage, which contributed to a dramatic drop where he lost
300 sb in less than 400 games. Mr. Rao resumed play the following day, but was unable to recover the losses, slipping further to \(-200\) sb after 3700 games. At this point, he stopped play and did a careful reassessment.

It was clear that his normal style for maximizing income against typical human opponents was not effective against the pseudo-optimal player. Whereas human players would normally succumb to a lot of pressure from aggressive betting, the bot was willing to call all the way to the showdown with as little as a Jack or Queen high card. That kind of play would be folly against most opponents, but is appropriate against an extremely aggressive opponent. Most human players fail to make the necessary adjustment under these atypical conditions, but the program has no sense of fear.

Mr. Rao changed his approach to be less aggressive, with immediate rewards, as shown by the +600 sb increase over the next 1100 games (some of which he credited to a good run of cards). Mr. Rao was able to utilize his knowledge that the computer player did not do any opponent modeling. Knowing this allows a human player to systematically probe for weaknesses, without any fear of being punished for playing...
in a methodical and highly predictable manner, since an oblivious opponent does not exploit those patterns and biases.

Although he enjoyed much more success in the match from that point forward, there were still some “adventures”, such as the sharp decline at 5400 games. Poker is a game of very high variance, especially between two opponents with sharp styles, as can be seen by the dramatic swings over the course of this match. Although 7000 games may seem like a lot, Mr. Rao’s victory in this match was still not statistically conclusive.

We now believe that a human poker master can eventually gain a sizable advantage over these pseudo-optimal prototypes (perhaps +0.20 sb/g or more is sustainable). However, it requires a good understanding of the design of the program and its resulting weaknesses. That knowledge is difficult to learn during normal play, due to the good information hiding provided by an appropriate mixture of plans and tactics. This “cloud of confusion” is a natural barrier to opponent learning. It would be even more difficult to learn against an adaptive program with good opponent modeling, since any methodical testing by the human would be easily exploited. This is in stark contrast to typical human opponents, who can often be accurately modeled after only a small number of games.

3.6 Conclusions and Future Work

The pseudo-optimal players presented in this paper are the first complete approximations of a game-theoretic equilibrium strategy for a full-scale variation of real poker.

Several abstraction techniques were explored, resulting in the reasonably accurate representation of a large imperfect information game tree having $\Theta(10^{18})$ nodes with a small collection of models of size $\Theta(10^7)$. Despite these massive reductions and simplifications, the resulting programs play respectably. For the first time ever, computer programs are not completely outclassed by strong human opposition in the game of 2-player Texas Hold’em.

Useful abstractions included betting tree reductions, truncation of betting rounds
combined with EV leaf nodes, and bypassing betting rounds. A 3-round model an-
chored at the root provided a pseudo-optimal strategy for the pre-flop round, which
in turn provided the proper contextual information needed to determine conditional
probabilities for post-flop models. The most powerful abstractions for reducing the
problem size were based on bucketing, a method for partitioning all possible hold-
ings according to strategic similarity. Although these methods exploit the particular
structure of the Texas Hold’em game tree, the principles are general enough to be
applied to a wide variety of imperfect information domains.

Many refinements and improvements will be made to the basic techniques in
the coming months. Further testing will also continue, since accurate assessment in
a high variance domain is always difficult.

The next stage of the research will be to apply these techniques to obtain ap-
proximations of Nash equilibria for $N$-player Texas Hold’em. This promises to be
a challenging extension, since multi-player games have many properties that do not
exist in the 2-player game.

Finally, having reasonable approximations of equilibrium strategies does not
lesser the importance of good opponent modeling. Learning against an adaptive
adversary in a stochastic game is a challenging problem, and there will be many
ideas to explore in combining the two different forms of information. That will
likely be the key difference between a program that can compete with the best, and
a program that surpasses all human players.

Quoting “thecount”:

“You have a very strong program. Once you add opponent modeling to
it, it will kill everyone.”

**Acknowledgments**

The authors would like to thank Gautam Rao, Sridhar Mutyala, and the other poker
players for donating their valuable time. We also wish to thank Daphne Koller,
Michael Littman, Matthew Ginsberg, Rich Sutton, David McAllester, Mason Mal-
muth, and David Sklansky for their valuable insights in past discussions.
This research was supported in part by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC), the Alberta Informatics Circle of Research Excellence (iCORE), and an Izaak Walton Killam Memorial postgraduate scholarship.

3.7 Chapter 3 Endnotes

3.7.1 Overlaying Models

Following the original publication of this paper, a serious flaw was discovered with the proposition of overlaying a 3-round pre-flop model with a 3-round post-flop model, as in the architecture of PSOPT2. Although this technique can produce systems that play reasonably well in practice, it is not well-grounded in theory.

In general there are an infinite number of possible equilibrium solutions to a complex game. This is certainly true in Texas Hold’em poker, which gives rise to the wide variety of playing styles that are completely viable.

The equilibrium solution to any one pre-flop model may be inappropriate for use as a model of any specific opponent, or as a generic opponent model. Combining a pre-flop equilibrium strategy with a post-flop equilibrium strategy is not guaranteed to produce an equilibrium strategy to the whole game, because the two strategies may not be consistent with each other.\footnote{This observation was formally proven by Neil Burch, using much simpler imperfect information domains to generate counterexamples.} In general, there is no reason to assume that two independently derived approximations will be harmonious with each other. Moreover, using any particular pre-flop strategy as a rigid model of one specific opponent can lead to seriously incorrect beliefs about the distribution of possible hands they can hold after the flop.

This inconsistency creates a noticeable “schism” or “gap” between the pre-flop and post-flop play in the pseudo-optimal players. For example, the program may re-raise in the pre-flop, building a large pot, but then fold on the flop to a single small bet. This is almost always incorrect, even if the flop was of no help whatsoever. In practice, the play in the pre-flop often provides key clues that make it easier to accurately deduce the computer player’s range of hands later in the game.
In our experience with both game-theoretic and adaptive poker algorithms, it has repeatedly been demonstrated that assuming one particular model of the opponent’s play can be far worse than having no model at all. That is, the naive assumption of a uniform distribution over opponent holdings is often safer and superior to having a plausible but counter-indicated model of the opponent.

### 3.7.2 Reverse-Mapping Strategies

Once an equilibrium strategy for an abstracted game has been obtained, there are many ways it can be used to play the real game. The most obvious is to simply do the reverse mapping of the abstraction to translate the strategy back onto the full game. However, numerous embellishments are worth consideration. One method in particular, called bucket splitting, was implemented but was not discussed in the original paper, due to space limitations.

Suppose that for a particular context we have a bucket 4 hand, and the equilibrium strategy gives us a mixed strategy of \( \{0.00, 0.60, 0.40\} \) for fold, call, and raise, respectively. The straightforward way to use this information is to spin a spinner (i.e., generate a random number in the range \(0.0 - 1.0\)) and play the corresponding action. However, this treats all hands in the bucket identically. That was a necessary assumption for accomplishing the coarse-granularity abstraction, but when the time comes to choose an action for the real game, it might be advantageous to distinguish between hands within that wide bucket range.

In simple bucket splitting, we distinguish between the top half and the bottom half of the hands in the bucket, and then bias our actions in a way that should produce better results. In the given example, we might call 100% of the time with hands from the bottom half, and raise 80% of the time with hands from the top half. Thus, we maintain the same recommended game-theoretic frequency (ratio) of actions overall, but we improve our hand selection for those designated actions. More generally, we can split the class into as many subdivisions as we like, up to the total number of distinct hands in the class. We call this more general process bucket splintering. In the extreme case, almost all hands in the class can be assigned a pure strategy, while the mixed strategy is still maintained for the class as a whole.
Technically, aligning the actions with an ordering of hands by strength could violate the equilibrium, and could potentially leak information. However, in practice the differences might be almost impossible to detect. Ironically, we can take advantage of the stochastic noise and partial observability of the game (which are usually impediments to us), to provide more than adequate obfuscation of our slightly biased actions. Moreover, in view of the crudeness of the entire approach, we can only claim that pseudo-optimal playing strategies are “in the spirit” of equilibrium strategies. Going to any length to preserve the integrity of the abstracted mixed strategies would almost certainly be needlessly pedantic.
Bibliography


