Opponent Modeling in Poker

Darse Billings, Denis Papp, Jonathan Schaeffer, Duane Szafron
Department of Computing Science
University of Alberta
Edmonton, Alberta Canada T6G 2H1
{darse, dpapp, jonathan, duane}@cs.ualberta.ca

Abstract

Poker is an interesting test-bed for artificial intelligence research. It is a game of imperfect knowledge, where multiple competing agents must deal with risk management, agent modeling, unreliable information and deception, much like decision-making applications in the real world. Agent modeling is one of the most difficult problems in decision-making applications and in poker it is essential to achieving high performance. This paper describes and evaluates *Loki*, a poker program capable of observing its opponents, constructing opponent models and dynamically adapting its play to best exploit patterns in the opponents' play.

Introduction

The artificial intelligence community has recently benefited from the tremendous publicity generated by the development of chess, checkers and Othello programs that are capable of defeating the best human players. However, there is an important difference between these board games and popular card games like bridge and poker. In the board games, players always have complete knowledge of the entire game state since it is visible to both participants. This property allows high performance to be achieved by brute-force search of the game trees. Bridge and poker involve imperfect information since the other players' cards are not known; search alone is insufficient to play these games well. Dealing with imperfect information is the main reason why research about bridge and poker has lagged behind other games. However, it is also the reason why they promise higher potential research benefits.

Until recently, poker has been largely ignored by the computing science community. However, poker has a number of attributes that make it an interesting domain for mainstream AI research. These include imperfect knowledge (the opponent's hands are hidden), multiple competing agents (more than two players), risk management (betting strategies and their consequences), agent modeling (identifying patterns in the opponent's strategy and exploiting them), deception (bluffing and varying your style of play), and dealing with unreliable information (taking into account your opponent's deceptive plays). All of these are challenging dimensions to a difficult problem.

There are two main approaches to poker research. One approach is to use simplified variants that are easier to analyze. However, one must be careful that the simplification does not remove challenging components of the problem. For example, Findler (1977) worked on and off for 20 years on a poker-playing program for 5-card draw poker. His approach was to model human cognitive processes and build a program that could learn, ignoring many of the interesting complexities of the game.

The other approach is to pick a real variant, and investigate it using mathematical analysis, simulation, and/or ad-hoc expert experience. Expert players with a penchant for mathematics are usually involved in this approach (Sklansky and Malmuth 1994, for example).

Recently, Koller and Pfeffer (1997) have been investigating poker from a theoretical point of view. They implemented the first practical algorithm for finding optimal randomized strategies in two-player imperfect information competitive games. This is done in their *Gala* system, a tool for specifying and solving problems of imperfect information. Their system builds trees to find the optimal game-theoretic strategy. However the tree sizes prompted the authors to state that "...we are nowhere close to being able to solve huge games such as full-scale poker, and it is unlikely that we will ever be able to do so."

We are attempting to build a program that is capable of beating the best human poker players. We have chosen to study the game of Texas Hold'em, the poker variation used to determine the world champion in the annual World Series of Poker. Hold'em is considered to be the most strategically complex poker variant that is widely played.

Our initial experience with a poker-playing program was positive (Billings et al. 1997). However, we quickly discovered how adaptive human players were. In games played over the Internet, our program, *Loki*, would perform quite well initially. Some opponents would detect patterns and weaknesses in the program's play, and they would alter their strategy to exploit them. One cannot be a strong poker player without modeling your opponent's play and adjusting to it.

Although opponent modeling has been studied before in the context of games (for example: Carmel and Markovitch 1995; Iida et al. 1995; Jansen 1992), it has not yet produced tangible improvements in practice. Part of the reason for this is that in games such as chess, opponent modeling is not critical to achieving high performance. In poker, however, opponent modeling is essential to success.

This paper describes and evaluates opponent modeling in *Loki*. The first sections describe the rules of Texas

Hold'em and the requirements of a strong Hold'em program as it relates to opponent modeling. We then describe how *Loki* evaluates poker hands, followed by a discussion of how opponents are modeled and how this information is used to alter the assessment of our hands. The next section gives some experimental results. The final section discusses ongoing work on this project. The major research contribution of this paper is that it is the first successful demonstration of using opponent modeling to improve performance in a realistic game-playing program.

Texas Hold'em

A hand of Texas Hold'em begins with the *pre-flop*, where each player is dealt two *hole cards* face down, followed by the first round of betting. Three community cards are then dealt face up on the table, called the *flop*, and the second round of betting occurs. On the *turn*, a fourth community card is dealt face up and another round of betting ensues. Finally, on the *river*, a fifth community card is dealt face up and the final round of betting occurs. All players still in the game turn over their two hidden cards for the *showdown*. The best five card poker hand formed from the two hole cards and the five community cards wins the pot. If a tie occurs, the pot is split. Texas Hold'em is typically played with 8 to 10 players.

Limit Texas Hold'em uses a structured betting system, where the order and amount of betting is strictly controlled on each betting round.[1] There are two denominations of bets, called the small bet and the big bet ($2 and $4 in this paper). In the first two betting rounds, all bets and raises are $2, while in the last two rounds, they are $4. In general, when it is a player's turn to act, one of five betting options is available: fold, call/check, or raise/bet. There is normally a maximum of three raises allowed per betting round. The betting option rotates clockwise until each player has matched the current bet or folded. If there is only one player remaining (all others having folded) that player is the winner and is awarded the pot without having to reveal their cards.

Requirements for a World-Class Poker Player

We have identified several key components that address some of the required activities of a strong poker player. However, these components are not independent. They must be continually refined as new capabilities are added to the program. Each of them is either directly or indirectly influenced by the introduction of opponent modeling.

Hand strength assesses how strong your hand is in relation to the other hands. At a minimum, it is a function of your cards and the current community cards. A better hand strength computation takes into account the number of players still in the game, position at the table, and the history of betting for the hand. An even more accurate calculation considers the probabilities for each possible opponent hand, based on the likelihood of each hand being played to the current point in the game.

Hand potential assesses the probability of a hand improving (or being overtaken) as additional community cards appear. For example, a hand that contains four cards in the same suit may have a low hand strength, but has good potential to win with a flush as more community cards are dealt. At a minimum, hand potential is a function of your cards and the current community cards. However, a better calculation could use all of the additional factors described in the hand strength computation.

Betting strategy determines whether to fold, call/check, or bet/raise in any given situation. A minimum model is based on hand strength. Refinements consider hand potential, pot odds (your winning chances compared to the expected return from the pot), bluffing, opponent modeling and trying to play unpredictably.

Bluffing allows you to make a profit from weak hands,[2] and can be used to create a false impression about your play to improve the profitability of subsequent hands. Bluffing is essential for successful play. Game theory can be used to compute a theoretically optimal bluffing frequency in certain situations. A minimal bluffing system merely bluffs this percentage of hands indiscriminately. In practice, you should also consider other factors (such as hand potential) and be able to predict the probability that your opponent will fold in order to identify profitable bluffing opportunities.

Unpredictability makes it difficult for opponents to form an accurate model of your strategy. By varying your playing strategy over time, opponents may be induced to make mistakes based on an incorrect model.

Opponent modeling allows you to determine a likely probability distribution for your opponent's hidden cards. A minimal opponent model might use a single model for all opponents in a given hand. Opponent modeling may be improved by modifying those probabilities based on collected statistics and betting history of each opponent.

There are several other identifiable characteristics which may not be necessary to play reasonably strong poker, but may eventually be required for world-class play.

The preceding discussion is intended to show how integral opponent modeling is to successful poker play. Koller and Pfeffer (1997) have proposed a system for constructing a game-theoretic optimal player. It is important to differentiate an *optimal* strategy from a *maximizing* strategy. The optimal player makes its decisions based on game-theoretic probabilities, without regard to specific context. The maximizing player takes into account the opponent's sub-optimal tendencies and adjusts its play to exploit these weaknesses.

In poker, a player that detects and adjusts to opponent weaknesses will win more than a player who does not. For example, against a strong conservative player, it would be correct to fold the probable second-best hand. However, against a weaker player who bluffs too much, it would be

[1] In No-limit Texas Hold'em, there are no restrictions on the size of bets.

[2] Other forms of deception (such as calling with a strong hand) are not considered here.

an error to fold that same hand. In real poker it is very common for opponents to play sub-optimally. A player who fails to detect and exploit these weaknesses will not win as much as a better player who does. Thus, a maximizing program will out-perform an optimal program against sub-optimal players because the maximizing program will do a better job of exploiting the sub-optimal players.

Although a game-theoretic optimal solution for Hold'em would be interesting, it would in no way "solve the game". To produce a world-class poker program, strong opponent modeling is essential.

Hand Assessment

Loki handles its play differently at the pre-flop, flop, turn and river. The play is controlled by two components: a hand evaluator and a betting strategy. This section describes how hand strength and potential are calculated and used to evaluate a hand.

Pre-flop Evaluation

Pre-flop play in Hold'em has been extensively studied in the poker literature (Sklansky and Malmuth 1994). These works attempt to explain the play in human understandable terms by classifying all the initial two-card pre-flop combinations into a number of categories. For each class of hands a suggested betting strategy is given, based on the category, number of players, position at the table, and type of opponents. These ideas could be implemented as an expert system, but a more systematic approach would be preferable, since it could be more easily modified and the ideas could be generalized to post-flop play.

For the initial two cards, there are {52 choose 2} = 1326 possible combinations, but only 169 distinct hand types. For each one of the 169 possible hand types, a simulation of 1,000,000 poker games was done against nine random hands. This produced a statistical measure of the approximate *income rate* (profit expectation) for each starting hand. A pair of aces had the highest income rate; a 2 and 7 of different suits had the lowest. There is a strong correlation between our simulation results and the pre-flop categorization given in Sklansky and Malmuth (1994).

Hand Strength

An assessment of the strength of a hand is critical to the program's performance on the flop, turn and river. The probability of holding the best hand at any time can be accurately estimated using enumeration techniques.

Suppose our hand is A♦-Q♣ and the flop is 3♥-4♣-J♥. There are 47 remaining unknown cards and therefore {47 choose 2} = 1,081 possible hands an opponent might hold. To estimate hand strength, we developed an enumeration algorithm that gives a percentile ranking of our hand (Figure 1). With no opponent modeling, we simply count the number of possible hands that are better than, equal to, and worse than ours. In this example, any three of a kind, two pair, one pair, or A-K is better (444 cases), the remaining A-Q combinations are equal (9 cases), and the rest of the hands are worse (628 cases). Counting ties as half, this corresponds to a percentile ranking, or hand strength (HS), of 0.585. In other words, there is a 58.5% chance that our hand is better than a random hand.

```
HandStrength(ourcards,boardcards)
{ ahead = tied = behind = 0
  ourrank = Rank(ourcards,boardcards)
  /* Consider all two card combinations of    */
  /* the remaining cards.                     */
  for each case(oppcards)
  { opprank = Rank(oppcards,boardcards)
    if(ourrank>opprank)          ahead += 1
    else if(ourrank=opprank)      tied += 1
    else /* < */                behind += 1
  }
  handstrength = (ahead+tied/2)
               / (ahead+tied+behind)
  return(handstrength)
}
```

Figure 1. HandStrength calculation

The hand strength calculation is with respect to one opponent but can be extrapolated to multiple opponents by raising it to the power of the number of active opponents. Against five opponents with random hands, the adjusted hand strength (HS_5) is $.585^5 = .069$. Hence, the presence of additional opponents has reduced the likelihood of our having the best hand to only 6.9%.

Hand Potential

In practice, hand strength alone is insufficient to assess the quality of a hand. Consider the hand 5♥-2♥ with the flop of 3♥-4♣-J♥. This is currently a very weak hand, but there is tremendous potential for improvement. With two cards yet to come, any heart, Ace, or 6 will give us a flush or straight. There is a high probability (over 50%) that this hand will improve to become the winning hand, so it has a lot of value. In general, we need to be aware of how the potential of a hand affects the effective hand strength.

We can use enumeration to compute this positive potential (Ppot), the probability of improving to the best hand when we are behind. Similarly, we can also compute the negative potential (Npot) of falling behind when we are ahead. For each of the possible 1,081 opposing hands, we consider the {45 choose 2} = 990 combinations of the next two community cards. For each subcase we count how many outcomes result in us being ahead, behind or tied (Figure 2).

The results for the example hand A♦-Q♣ / 3♥-4♣-J♥ versus a single opponent are shown in Table 1. The rows are labeled by the status on the flop. The columns are labeled with the final state after the last two community cards are dealt. For example, there are 91,981 ways we could be ahead on the river after being behind on the flop. Of the remaining outcomes, 1,036 leave us tied with the best hand, and we stay behind in 346,543 cases. In other words, if we are behind a random hand on the flop we have roughly a 21% chance of winning the showdown.

In Figure 2 and Table 1, we compute the potential based on two additional cards. This technique is called two-card lookahead and it produces a $Ppot_2$ of 0.208 and an $Npot_2$ of 0.274. We can do a similar calculation based on one-card lookahead ($Ppot_1$) where there are only 45 possible upcoming cards (44 if we are on the turn) instead of 990 outcomes. With respect to one-card lookahead on the flop, $Ppot_1$ is 0.108 and $Npot_1$ is 0.145.

```
HandPotential(ourcards,boardcards)
{ /* Hand potential array, each index repre-  */
  /* sents ahead, tied, and behind.          */
  integer array HP[3][3]   /* initialize to 0 */
  integer array HPTotal[3] /* initialize to 0 */

  ourrank = Rank(ourcards,boardcards)
  /* Consider all two card combinations of    */
  /* the remaining cards for the opponent.    */
  for each case(oppcards)
  { opprank = Rank(oppcards,boardcards)
    if(ourrank>opprank)          index = ahead
    else if(ourrank=opprank)     index = tied
    else /* < */                 index = behind
    HPTotal[index] += 1

    /* All possible board cards to come.      */
    for each case(turn,river)
    { /* Final 5-card board */
      board = [boardcards,turn,river]
      ourbest = Rank(ourcards,board)
      oppbest = Rank(oppcards,board)
      if(ourbest>oppbest)      HP[index][ahead]+=1
      else if(ourbest=oppbest) HP[index][tied]+=1
      else  /* < */           HP[index][behind]+=1
    }
  }
  /* Ppot: were behind but moved ahead.       */
  Ppot = (HP[behind][ahead]+HP[behind][tied]/2
                          +HP[tied][ahead]/2)
       / (HPTotal[behind]+HPTotal[tied])
  /* Npot: were ahead but fell behind.        */
  Npot = (HP[ahead][behind]+HP[tied][behind]/2
                          +HP[ahead][tied]/2)
      / (HPTotal[ahead]+HPTotal[tied])
  return(Ppot,Npot)
}
```

Figure 2. HandPotential calculation

5 Cards	7 Cards			
	Ahead	Tied	Behind	Sum
Ahead	449005	3211	169504	621720 = 628x990
Tied	0	8370	540	8910 = 9x990
Behind	91981	1036	346543	439560 = 444x990
Sum	540986	12617	516587	1070190 = 1081x990

Table 1. A♦-Q♣ / 3♥-4♣-J♥ potential

These calculations provide accurate probabilities that take every possible scenario into account, giving smooth, robust results. However, the assumption that all two-card opponent hands are equally likely is false, and the computations must be modified to reflect this.

Betting Strategy

When it is our turn to act, how do we use hand strength and hand potential to select a betting action? What other information is useful and how should it be used? The answers to these questions are not trivial and this is one of the reasons that poker is a good test-bed for artificial intelligence. The current betting strategy in *Loki* is unsophisticated and can be improved (Billings et al. 1997). It is sufficient to know that betting strategy is based primarily on two things:

1. *Effective hand strength* (EHS) includes hands where we are ahead, and those where we have a Ppot chance that we can pull ahead:
$$EHS = HS_n + (1 - HS_n) \times Ppot$$
2. *Pot odds* are your winning chances compared to the expected return from the pot. If you assess your chance of winning to be 25%, you would call a $4 bet to win a $16 pot (4/(16+4) = 0.20) because the pot odds are in your favor (0.25 > = 0.20).

Opponent Modeling

In strategic games like chess, the performance loss by ignoring opponent modeling is small, and hence it is usually ignored. In contrast, not only does opponent modeling have tremendous value in poker, it can be the distinguishing feature between players at different skill levels. If a set of players all have a comparable knowledge of poker fundamentals, the ability to alter decisions based on an accurate model of the opponent may have a greater impact on success than any other strategic principle.

Having argued that some form of opponent modeling is indispensable, the actual method of gathering information and using it for betting decisions is a complex and interesting problem. Not only is it difficult to make appropriate inferences from certain observations and then apply them in practice, it is not even clear how statistics should be collected or categorized.

Weighting the Enumeration

Many weak hands that probably would have been folded before the flop, such as 4♥-J♣, may form a very strong hand with the example flop of 3♥-4♣-J♥. Giving equal probabilities to all starting hands skews the hand evaluations compared to more realistic assumptions. Therefore, for each starting hand, we need to define a probability that our opponent would have played that hand in the observed manner. We call the probabilities for each of these 1,081 subcases *weights* since they act as multipliers in the enumeration computations.[1]

The use of these weights is the first step towards opponent modeling since we are changing our computations based on the relative probabilities of different cards that our opponents may hold. The simplest approach to determining these weights is to treat all opponents the same, calculating a single set of weights to reflect “reasonable” behavior, and use them for all opponents. An initial set of weights was determined by ranking the 169 distinct starting hands and assigning a probability commensurate with the strength (income rate) of each hand (as determined by simulations).

There are two distinct ways to improve the accuracy of the calculations based on these weights. First, an opponent’s betting actions can be used to adjust the

[1] The probability that an opponent holds a particular hand is the weight of that subcase divided by the sum of the weights for all the subcases.

weights. For example, if an opponent raises on the flop, the weights for stronger hands should be increased and the weights for weaker hands should be decreased. We call this *generic* modeling since the model is identical for all opponents in the same situation. Second, we can maintain a separate set of weights for each opponent, based on their betting history. We call this technique *specific* modeling, because it differentiates between opponents.

Each opponent is assigned an array of weights indexed by the two-card starting hands. Each time an opponent makes a betting action, the weights for that opponent are modified to account for the action. For example, a raise increases the weights for the strongest hands likely to be held by the opponent given the flop cards, and decreases the weights for the weaker hands. This means that at any point during the hand, the weight reflects the relative probability that the opponent has that particular hand.

If these weights are used for opponent modeling, the algorithms of Figures 1 and 2 are only slightly modified. Each of the increments ("+= 1") is replaced with the code "+= Weight[oppcards]". There are two problems that must be solved to make this form of opponent modeling work. First, what should the initial weights be? Second, what transformation functions should be applied to the weights to account for a particular opponent action?

Computing Initial Weights

The initial weights are based on the starting hands each opponent will play. The most important observed information is the frequency of folding, calling and raising before the flop. We deduce the mean (μ, representing the median hand) and variance (σ, for uncertainty) of the threshold needed for each player's observed action. These are interpreted in terms of income rate values, and mapped onto a set of initial weights for the opponent model.

Suppose an opponent calls 30% of all hands, and this translates to a median hand whose income rate is +200 (roughly corresponding to an average of 0.2 bets won per hand played). If we assume a σ that translates to an income rate of +/-100, then we would assign a weight of 1.0 to all hands above +300, a weight of 0.01 to all hands below +100, and a proportional weight for values between +100 and +300. The median hand at +200 is thus given a 0.50 weight in the model. A weight of 0.01 is used for "low" probabilities to avoid labeling any hand as "impossible". While this approach will not reveal certain opponent-specific tendencies, it does provide reasonable estimates for the probability of each possible starting hand.

To classify the opponent's observed actions, we consider the action (fold, check/call, bet/raise) taken by the opponent, how much the action cost (bets of 0, 1, or > 1) and the betting round in which it occurred (pre-flop, flop, turn, river). This yields 36 different categories. Some of these actions do not normally occur (e.g. folding to no bet) and others are rare. Each betting action an opponent makes results in one of these categories being incremented. These statistics are then used to calculate the relevant frequencies.

Re-weighting

Each time an opponent makes a betting action, we modify the weights by applying a transformation function. For simplicity we do not do any re-weighting on the pre-flop, preferring to translate income rates into weights. For the betting rounds after the flop, we infer a mean and variance (μ and σ) of the threshold for the opponent's observed action. However, we can no longer map our μ and σ to a list of ranked starting hands. Instead, we must rank all of the five card hands that are formed from each starting hand and the three community cards. To do this, we use EHS.

For example, based on observed frequencies, we may deduce that an opponent needs a median hand value of 0.6 to call a bet, with a lower bound of 0.4 and an upper bound of 0.8. In this case, all hands with an EHS greater than 0.8 are given re-weighting factors of 1.0. Any hand with a value less than 0.4 is assigned a re-weighting factor of 0.01, and a linear interpolation is performed for values between 0.4 and 0.8. Figure 3 shows the algorithm used for computing the re-weighting factors for a given μ and σ.

Recall that EHS is a function of both HS and Ppot. Since $Ppot_2$ is expensive to compute, we currently use a crude but fast function for estimating potential, which produces values within 5% of $Ppot_2$, 95% of the time. Since these values are amortized over the 1,081 five-card hands, the overall effect of this approximation is small.

For each five-card hand, the computed re-weighting factor is multiplied by the initial weight to produce the updated weight. The process is repeated for each observed betting decision during the hand. By the last round of betting, a certain opponent may have only a small number of hands that have relatively high weights, meaning that the program has zeroed in on a narrow range of possible hands.

Table 2 illustrates how the re-weighting is performed on some selected examples for a flop of 3♥-4♣-J♥, with μ=0.60 and σ=0.20. The context considers an opponent who called a bet before the flop and then bet after the flop. For each possible hand we note the initial weight (Weight), unweighted hand rank (HR), hand strength (HS_1), approximate $Ppot_2$ ($\sim PP_2$), effective hand strength (EHS), the re-weighting factor based on μ = 0.6 and σ = 0.2 (Rwt), and the new overall weight (Nwt).

```
constant  low_wt  0.01
constant high_wt  1.00

Reweight(μ,σ,weight,boardcards)
{ /* interpolate in the range μ +- σ. */
  for each case(oppcards)
  { EHS=EffectiveHandStrength(oppcards,boardcards)
    reweight = (EHS-μ+σ)/(2*σ)
    /* Assign low weights below (μ-σ). */
    if(reweight<low_wt)  reweight = low_wt
    /* Assign high weights above (μ+σ). */
    if(reweight>high_wt) reweight = high_wt
    weight[subcase] = weight[subcase]*reweight
  }
}
```

Figure 3. Computing the re-weighting factors

Hand	Weight	HR	HS_1	$\sim PP_2$	EHS	Rwt	Nwt	Comment
J♣ 4♥	0.01	0.993	0.990	0.04	0.99	1.00	0.01	very strong, but unlikely
A♣ J♣	1.00	0.956	0.931	0.09	0.94	1.00	1.00	strong, very likely
5♥ 2♥	0.20	0.004	0.001	0.35	0.91	1.00	0.20	weak, but very high potential
6♠ 5♠	0.60	0.026	0.006	0.21	0.76	0.90	0.54	weak, good potential
5♠ 5♥	0.70	0.816	0.736	0.04	0.74	0.85	0.60	moderate, low potential
5♠ 3♠	0.40	0.648	0.671	0.10	0.70	0.75	0.30	mediocre, moderate potential
A♣ Q♦	1.00	0.585	0.584	0.11	0.64	0.60	0.60	mediocre, moderate potential
7♠ 5♠	0.60	0.052	0.012	0.12	0.48	0.20	0.12	weak, moderate potential
Q♠ T♠	0.90	0.359	0.189	0.07	0.22	0.01	0.01	weak, little potential

Table 2. Re-weighting various hands after a 3♥-4♣-J♥ flop ($\mu = 0.6$, $\sigma = 0.2$)

Consider the case of Q♠ T♠. In the pre-flop this is a fairly strong hand, as reflected by its income rate of +359 and weighting of 0.90. However, these cards do not mesh well with the flop cards, resulting in low hand strength (0.189) and low potential (0.07). This translates to an effective hand strength of 0.22. Given that observations of the opponent show that they will only bet with hands of strength μ=0.60 (+/-σ=0.20), we assign this a re-weighting of 0.01 (since $0.22<\mu-\sigma$). Hence, we will consider this hand to be unlikely for this opponent from this point on.

The opponent's decisions may actually be based on a different metric than EHS, resulting in an imperfect model. New techniques can improve the results, but the current method does capture much of the information conveyed by the opponent's actions.

In competitive poker, opponent modeling is more complex than portrayed here. One also wants to fool the opponent into constructing a poor model. For example, early in a session a strong poker player may try to create the impression of being very conservative, only to exploit that image later when the opponents are using incorrect assumptions. In two-player games, the M* algorithm allows for recursive definitions of opponent models, but it has not been demonstrated to improve performance in practice (Carmel and Markovitch 1995).

Experiments

Self-play simulations offer a convenient method for the comparison of two or more versions of the program. Our simulations use a duplicate tournament system, based on the same principle as duplicate bridge. Since each hand can be played with no memory of preceding hands, it is possible to replay the same deal, but with the participants holding a different set of hole cards each time. Our tournament system simulates a ten-player game, where each deal is replayed ten times, shuffling the seating arrangement so that every participant has the opportunity to play each set of hole cards once. This arrangement greatly reduces the "luck element" of the game, since each player will have the same number of good and bad hands. The differences in the performance of players will therefore be based more strongly on the quality of the decisions made in each situation. This large reduction in natural variance means that meaningful results can be obtained with a much smaller number of trials than in a typical game setting. Nevertheless, it is important to not over-interpret the results of one simulation.

Figure 4 shows the results of self-play simulations between several versions of *Loki* playing $2/$4 Hold'em. Two copies of five different programs played 100,000 hands. The average profit is plotted against the number of hands played. Each data point is the average of the two programs of that type.

We started with our previous best version of *Loki*, representing a basic player (BPM). BPM uses a crude system for assigning initial weights, so some generic modeling is done for the pre-flop hand selection[1] but there is no re-weighting. To add diversity to the simulated game, the betting parameters of *Loki* were modified to include two additional styles of play (BPT and BPL). BPT is a "tight" (conservative) player, while BPL is a "loose" (more liberal) player, but are otherwise identical to BPM. Two opponent modeling variations of BPM were then added (GOM and SOM). GOM uses generic opponent modeling with re-weighting, and the default models for the opponent are based on how GOM itself plays. SOM uses specific opponent modeling with re-weighting, starting with the GOM defaults but basing its decisions entirely on observed behavior after 20 data points have been collected.

Very quickly, the two opponent modeling programs asserted their superiority over the non-modeling versions. GOM is able to exploit the basic players, as expected, because its model of how they play is accurate, and is used to make better decisions. GOM might not perform as well against players with very different styles of play, because its model would be less accurate, but it would be better than using no modeling at all.

SOM is more successful using observed frequencies rather than a good default model. Against players with very different styles of play, as typically seen in human competition, SOM's advantage over GOM will be magnified.

Another experiment was performed with a single copy of SOM against nine copies of BPM. After 100,000 hands the SOM version was ahead roughly $5,000, while the BPM player had lost more than $550 on average. The advantage

[1] This feature was not removed because having no information at all would be a large handicap.

of good opponent modeling is clear – *Loki* with opponent modeling is a stronger program than without it.

Loki has also been tested in more realistic games against human opposition. For this purpose, the program participates in an on-line poker game, running on the Internet Relay Chat (IRC). Human players connect to IRC and participate in games conducted by dedicated server programs. No real money is at stake, but bankroll statistics on each player are maintained. Both the new opponent modeling versions of *Loki* and the previous versions with no re-weighting win consistently when playing on the IRC server. The natural variance in these games is very high, and the results depend strongly on which players happen to be playing. Consequently, not enough information has been gathered to safely conclude that the opponent modeling versions of *Loki* are outperforming the previous best program in these games.

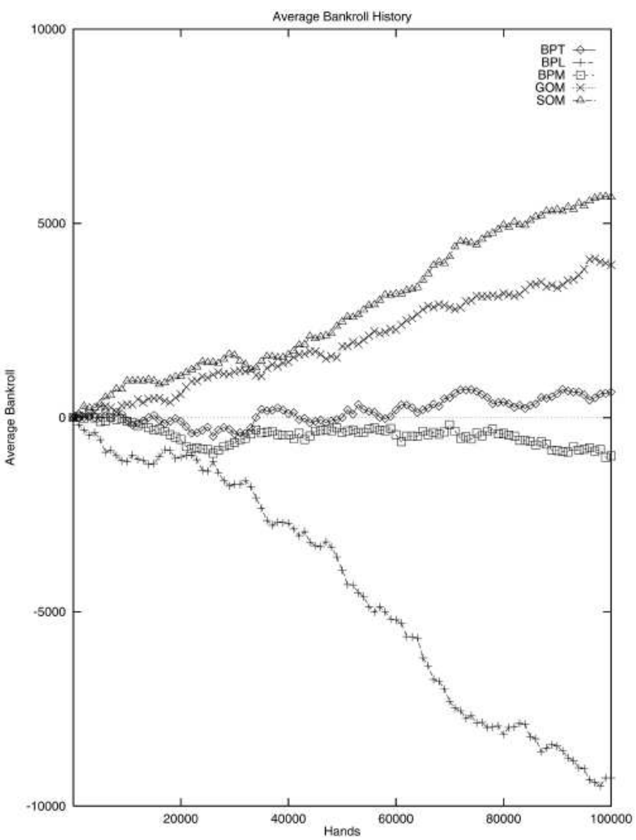

Figure 4. Experiments with different versions of *Loki.*

Conclusions and Work in Progress

Loki successfully uses opponent modeling to improve its play. This is the first demonstration of beneficial opponent modeling in a high-performance game-playing program.

However, it does not necessarily follow that opponent modeling will be just as successful in games against human players as it was in the closed experiments. Humans are also very good at opponent modeling, and can be less predictable than the players in these simulations. We have not yet investigated modeling opponents who vary their strategy over time.

Specific opponent modeling was hampered by the crude method used for collecting and applying observed statistics. Much of the relevant context was ignored for simplicity, such as combinations of actions within the same betting round. A more sophisticated method for observing and utilizing opponent behavior would allow for a more flexible and accurate opponent model.

Poker is a complex game. Strong play requires the player to excel in all aspects of the game. Developing *Loki* seems to be a cyclic process. We improve one aspect of the program until it becomes apparent that another aspect is the performance bottleneck. That problem is then tackled until it is no longer the limiting factor, and a new weakness in the program's play is revealed. We have made our initial foray into opponent modeling and are pleased with the results, although it is far from a completed subject. It is now apparent that the program's betting strategy is the major cause for concern. Hence we will now focus our efforts on that topic, and opponent modeling will be revisited in the future.

Acknowledgments

We gratefully thank the referees for their insightful comments. Regretfully, some of their excellent suggestions were omitted due to space constraints.

This research was supported by the Natural Sciences and Engineering Council of Canada.

References

Billings, D., Papp, D., Schaeffer, J. and Szafron, D. 1997. Poker as a Testbed for AI Research, AI'98. To appear.

Carmel, D. and Markovitch, S. 1995. Incorporating Opponent Models into Adversary Search. AAAI, 120-125.

Findler, N. 1977. Studies in Machine Cognition Using the Game of Poker. *CACM* 20(4):230-245.

Iida, H. , Uiterwijk, J. , van den Herik, J. and Herschberg, I. 1995. Thoughts on the Application of Opponent-Model Search. In *Advances in Computer Chess 7*, University of Maastricht, 61-78.

Jansen, P. 1992. Using Knowledge about the Opponent in Game-Tree Search. Ph.D. diss., Dept. of Computer Science, Carnegie-Mellon University.

Koller, D. and Pfeffer, A. 1997. Representations and Solutions for Game-Theoretic Problems. *Artificial Intelligence* 94(1-2), 167-215.

Sklansky, D. and Malmuth, M. 1994. *Hold'em Poker for Advanced Players.* Two Plus Two Publishing.