

# Evolving Opponent Models for Texas Hold 'Em

Alan J. Lockett and Risto Miikkulainen

**Abstract**—Opponent models allow software agents to assess a multi-agent environment more accurately and therefore improve the agent's performance. This paper makes use of coarse approximations to game-theoretic player representations to improve the performance of software players in Limit Texas Hold 'Em poker. A 10-parameter model, intended to model a combination, or mixture, of various strategies is developed to represent the opponent. A 'mixture identifier' is then evolved using the NEAT neuroevolution method to estimate values of these parameters for arbitrary opponents. To evaluate this approach, two poker players, represented as neural networks, were evolved under the same conditions, one with the mixture identifier, and one without. The player trained with access to the identifier achieved consistently higher and more stable fitness during evolution compared with the player without the identifier. Further, the player with the identifier outplays the other in a heads-up match after training, winning on average 60% of the money at the table. These results demonstrate that opponent modeling is effective even with low-dimensional models and conveys an advantage to players trained to use these models.

## I. INTRODUCTION

An increasing number of applications of artificial intelligence require the ability to build and maintain computational models of autonomous agents. Autonomous vehicles must be able to construct accurate models of agents in their environment quickly so that they can respond adaptively. Software agents managing financial transactions must be able to identify fraudulent behavior in a timely manner. As computer programs are increasingly placed in the role of a decision maker, computational methods are increasingly needed to analyze the motives and intent of the agents with which they interact.

Within the field of artificial intelligence, games have traditionally provided a ready test bed for new ideas and approaches to difficult decision-making problems, since games allow for testing in complex and ever more realistic environments with relatively low start-up costs and little risk to human safety. Within AI, poker is perhaps the most appropriate target for opponent modeling, since identifying optimal strategies for poker has proven elusive. In poker,

the opponent's behavior provides the primary window into the opponent's state. Further, the natural incentives for deception inherent in the game make this window a rather opaque one, and thus any method that can effectively decipher a poker player's actions in order to obtain therefrom a reasonable and useful model of the opponent should generalize well to other environments.

There are two fundamentally different approaches to opponent modeling in poker and other similar games. On the one hand, one might seek a direct *predictive* model that would construct some probability distribution over the future actions or current state of the agent being modeled. A model of this sort might also be used to estimate hidden state in situations where another agent in the environment has access to information only available to the observer through that agent's actions. In the context of poker, this approach might involve estimating the cards likely present in the opponent's hand or perhaps attempting to guess whether or not the opponent is bluffing or slowplaying on a particular hand. Modeling opponent actions or state in this fashion is transparent and immediately useful; that is, the output has a known interpretation that impinges directly on the decision-making problem at hand. However, previous work indicates that predictive models may be unstable, and it is not clear a priori how to build such a model in practice [1, 2, 3].

Another approach, and the one pursued in this paper, would attempt to *classify* the opponent by type, either as belonging to a specific class out of some discrete set of categories, or as a point (or region) within a continuous description space. Whereas a predictive model tries to identify what the opponent will do next, a classification model will attempt to identify what the opponent is like by analogy with previously observed opponents. Intuitively, this concept resembles how people approach game strategy, by identifying opponents in terms of past experience, and reasoning forward from these analogies to anticipate opponent action, in essence using a classification model as a means to obtain a predictive one. In poker, a common categorization (although not used in this paper) might be to identify the strategy of the opponent as tight or loose, and passive or aggressive. One drawback of this approach is that classification models are not necessarily transparent or direct; i.e. there may be no obvious interpretation that can be given from the classification output to game decisions.

Alan Lockett is with the Department of Computer Sciences at the University of Texas, Austin, TX 78712 USA (e-mail: alockett@cs.utexas.edu).

Risto Miikkulainen is with the Department of Computer Sciences, University of Texas, Austin, TX 78712 USA (e-mail: risto@cs.utexas.edu).

For statistical methods (including neuroevolution), however, transparency is irrelevant, since these methods cannot take into account the intensional semantics of the model per se. Classification models are simpler to construct and learn than predictive models in practice [3]. While it may not seem clear from the outset how to select useful opponent categories or how to assign actual opponents to these categories on-line, it is in fact quite feasible, as will be shown later in this paper.

This paper demonstrates the successful application of a continuous classification approach to opponent modeling in Texas Hold 'Em poker. The models use a coarse approximation to game-theoretic opponent representations to provide a parameterized description of a large subclass of poker players. It is important to note that in this research, classification is performed in a continuous space rather than over some discrete set of opponents. Evolutionary algorithms are shown to be able to train a neural network to reliably associate a model with an adversary. In addition, poker players are trained using neuroevolution both with and without access to the estimated models. The players with the models conclusively outperform the players without models in three distinct aspects: (1) they attain higher maximum and average fitness under the same fitness function, (2) their fitness is more stable, i.e. it varies less across generations, and (3) they routinely outplay the non-opponent modeling players in heads-up matches after equivalent training.

## II. RELATED WORK

A significant body of opponent modeling work has been done in poker, much of it using an explicit approach [4, 5]. For instance, Billings *et al.* [1] used statistical methods to estimate the strength of the opponent's hand given his history of calling, raising, or folding. They also developed predictive models to assess what decision a specific opponent would make when holding a given hand. Although the original predictor was only 51% accurate, Davidson *et al.* [2], [6] used a neural network trained with backpropagation to increase its accuracy to 81%. These data are especially interesting since their poker player, Loki, gathered statistics on actual human players by playing online poker games. However, the approach required a significant history of data for training and therefore could not be used online. In contrast, the mixture-based approach does not require additional training in order to generalize to new players, since previously unseen opponents can be interpolated from the training experience using the mixture models.

More recently, Bard and Bowling [7] formulated opponent modeling as a dual state estimation problem in Kuhn poker, a simplified, three-card version of two-player

poker. There are only five non-dominated strategies: three for the first player and two for the second. These opponent models represent mixed strategies, a feature held in common with this current work.

In substance, the mixture method of this paper shares broad similarities with what Bard and Bowling term 'static' opponent models. In fact, it provides a first approach to a reasonably-sized approximation of complete opponent models for poker, a development which they suggest as a next step. It differs significantly, however, in how the mixture models are used once obtained. Rather than trying to solve explicitly for a mixed strategy to exploit the opponent, this work uses neuroevolution in order to search for effective game players. This is a fundamentally distinct methodology, based on the point of view that the opponent models will invariably have some stochastic bias or error that is best handled by using stochastic methods for interpreting them. Exact computations based on the outcome of the mixture identification problem could lead a computer player astray, turning an attempt to exploit the opponent's weaknesses into a trap. An algorithm that makes use of stochastic measurements should also be able to assess and mitigate risk. An exact computational method affords no such flexibility.

The mixture approach to opponent modeling is based on that applied by Lockett *et al.* [3] in a simpler card game called Guess It. In their approach, a set of four *cardinal opponents* was used to define an *opponent space*, with all possible opponents being represented as probability distributions over these four basic opponents. These distributions were termed *mixture opponents*, and at each turn, the distribution was sampled to decide which of the four cardinal opponents would make the decision for that turn. A *mixture identifier* was trained to estimate the sampling distribution of a mixture opponent from the current game state. Using neuroevolution, Lockett *et al.* were able to train the mixture identifier to an accuracy of about 85 percent. Two separate neural networks were then trained, one of which took in the game state plus the output of the mixture identifier, and another that took in the game state only. While the network with only the game state achieved greater fitness against the mixture opponents, the authors found that the networks with both the game state and the mixture identifier consistently won against a bank of previously unseen players, including the network with just the game state. The players that were trained to use the mixture identifier were able to generalize to unseen opponents because they developed an exhaustive and continuous representation of opponents encoded in the mixture identifier.

In this paper, the mixture approach provides a continuous classification system for opponents that should generalize

well because unseen opponents can be viewed as interpolations of previously seen opponents. This work extends the mixture approach from Guess It [3] to the domain of poker by clarifying the nature of approximate opponent representations and providing a means to generate such representations for new domains with minimal effort. These improvements make the approach theoretically clear and scale it up to two-player Limit Texas Hold 'Em poker, a more complex and difficult domain.

The neural networks are trained to play poker using NeuroEvolution of Augmenting Topologies (NEAT), developed by Stanley and Miikkulainen [8]. In this approach, only inputs and outputs are specified for the neural network. The appropriate internal topology is discovered through a search using a genetic algorithm. Connections and hidden nodes are added and changed with a given probability, and are retained in the population if they improve the performance of the network against a fitness function. In theory, the capability of the algorithm to iteratively add structure (or *complexify*) allows it to adjust to new situations without losing old capabilities. The details of NEAT will not be discussed here (see [8] instead), partly because the opponent modeling architecture advocated in this paper is independent of the particular algorithm used to implement it. However, since NEAT has been used effectively in various game-playing approaches in the past, it was a natural choice for the opponent modeling approach as well.

As a final note, similar opponent representations for poker have been previously employed by Barone and While [9]. Their emphasis, however, was on using evolution to find good poker players from among these representations, whereas the players evolved in this work are not limited to the mixture representations, which are only used to model opponents encountered by the automated player.

### III. TEXAS HOLD 'EM POKER

Texas Hold 'Em poker is currently the most popular version of poker in casinos and tournaments. In this, and in most poker research, the actual game studied is Limit Texas Hold 'Em, where the bets are of predetermined fixed size. The game begins when each player *buys in* to the table by presenting a fixed amount of money for play. In Texas Hold 'Em, one player is always designated as the dealer, and the dealer position rotates with each hand. In a hand, each player is initially dealt two cards face down, called the *hole cards*. Before seeing the cards, the player to the left of the dealer must add a forced bet to the pot called the *small blind*, and the player to that player's left must place a bet usually twice this size called the *big blind*.

Once the hole cards have been dealt, a betting round ensues, starting with the player placing the small blind.

Each player in turn has the choice to *fold*, conceding the game and losing all prior bets in the hand, to *call*, matching the largest bet in the pot at the time (initially the size of the big blind), or to *raise* the cost of playing for the pot by the size of the big blind. If any player raises, then all prior players have the opportunity to take another turn.

Once all the players have placed their bets, then the round advances to the *flop*, where three community cards are dealt face up. Another betting round follows, this time starting with the player to the dealer's left. Two additional options become available: this player can *check*, passing the opportunity to bet but leaving open the option to meet future raises, or *bet*, adding money to the pot and placing a cost on remaining in the game.

After the flop, the cost of a bet doubles, and there are two more betting rounds, the *turn* and the *river*, with one community card added during each. If at any point only one player remains in the hand, then that player wins the pot, which is then added to his *bankroll*. If more than one player remains in the hand after the river, then each player must show his or her two private cards, and the player with the best five-card poker hand formed from all seven cards in play wins. This step is called the *showdown*.

In this research, all games consist of 250 hands of two-player poker with a buy-in of \$200 and \$2 blinds. Automated players are judged according to how much of the \$400 at the table they own at the end of 250 hands. For practical purposes, *check* and *call* are identical, as well as *bet* and *raise*, so that the players here have three strategies available at each turn: *fold*, *call*, or *raise*.

If poker were a game of pure chance with completely stochastic outcomes, it would be possible over the long term to maintain positive winnings simply by playing according to the statistics. The expected winnings for an individual turn of poker can be computed as

$$E(W) = pr - (1-p)c$$

where  $p$  is the probability of having the best cards at the table, termed here the *win probability*,  $r$  is the amount of money in the pot,  $c$  is the cost of betting, and  $W$  is a random variable for the amount of money won on this decision, equal to  $r$  in case of a win, and  $-c$  in case of a loss. Often, the equation above is transformed into a ratio by setting it equal to zero and shifting terms. This leads to the familiar concept of *pot odds*, which are basically an estimate of the statistical breakpoint between earning and losing money on a bet.

In order to estimate the pot odds, it is necessary to have available an estimate of the win probability,  $p$ . For human players, this estimate is usually obtained by considering the number of cards needed in order to complete specific hands (termed *outs*), keeping in mind the opponent's likely best hand as well. It is possible to compute  $p$  exactly under the

assumption of a fair deck with uniform probability over the cards. However, an exact computation is too involved to compute directly, so the win probability must be estimated. In this paper, an approximation to a roll-out of the current state is used, estimating the win probability assuming that no players will fold prior to the showdown. This estimate is obtained by combining the probability of each player's best hand belonging to one of 315 exact hand types with the probability of winning the match given these hand types.

The neural networks trained to play poker all possess the same basic input structure consisting of:

- (1) The estimated win probability,  $p$ ,
- (2) The ratio of expected winnings if CALL is selected,
- (3) The ratio of expected winnings if RAISE is selected,
- (4) The current round,
- (5) The size of the pot,
- (6) The size of their own bankroll,
- (7) The number of raises made by the opponent this round,
- (8) The required cost of a bet.

These eight inputs are collectively considered to be the game state for poker from the point of view of the network. While it might be desirable to provide the network with greater visibility into the cards held than just that provided by the win probability, it is also necessary to keep the number of parameters as small as possible in order for training to be feasible, and there is no obvious compact parameterization of the cards that would be sufficiently small and useful enough to justify including them as inputs to the network. Notable among these inputs is the number of raises this round by the opponent (7), which provides the network with its strongest clue as to the value of the cards held by the opponent. This representation contains sufficient detail to evaluate the effectiveness of opponent modeling in Texas Hold 'Em.

#### IV. A MIXTURE-BASED APPROACH

In this paper, low-dimensional approximations to a full model will be developed that can then be used both to generate training opponents and to incorporate knowledge of the opponent into an automated player. In terms of classical game theory, these opponent models represent approximations to mixed strategies, whence these opponents are termed as *mixture opponents*. An 18-parameter model for poker opponents is developed for this research, which is later pared down to 10 parameters.

As discussed above, for both practical and theoretical reasons, useful opponent models need to be relatively compact, so that parameters describing the current opponent can be identified quickly, and so that the models can be trained accurately in reasonable time. The mixture parameters were obtained by partitioning the poker game

state. As shown in Figure 1, the state space was broken into nine independent regions based on the win probability  $p$  and the expected winnings,  $E(W)$ . In each region there are two degrees of freedom initially, but more useful models are obtained by deterministically folding losing hands, leaving 10 parameters corresponding the probability of betting or calling in the remaining five state regions.

This partition was chosen to classify players based on how they utilize two pieces of information: the win probability and the expected winnings. These two pieces of information were chosen because they represent criteria that human poker players often use to judge the value of a hand; other metrics could also be used if desired. In each state region, the player has the option to FOLD, CALL, or RAISE. These choices are represented by two parameters, the probability of raising and the probability of calling; the fold probability is fully determined by these two. In addition, the model is simplified by assuming the player will deterministically fold when holding a very bad hand, leaving five regions of the state space where the opponent may take a probabilistic action. This reasoning results in a 10-parameter model generally expressing the opponent's willingness to bet based on the strength of his hand. There are numerous other aspects of poker players that one could wish to model, e.g. aggressiveness vs. passivity. Such aspects are beyond the scope of the current research; the current goal is to validate the mixture-based approach for poker in general terms. Using this approach, potential opponents can be sampled in a straightforward manner. These opponents can be chosen to provide a diverse training set from the outset, which represents an improvement over training by self-play or against a set of manually constructed opponents. With genetic algorithms, these generated opponents can be used exclusively, as is done in this paper, or in conjunction with competitive coevolution to provide greater diversity and robustness to the fitness function from the start.

Mixture opponents were initially generated uniformly at random within these 10 parameters, subject to the constraint that the pair of parameters corresponding to each of the nine state regions must sum to a number between zero and one. However, many of these mixtures did not produce viable opponents. To surmount this difficulty, a sample of 1000 uniformly generated mixtures was played against randomly generated neural networks taking the game state as input. Out of these networks, a sub-sample of 84 mixtures was kept, all of which had managed to retain at least \$10 out of an initial \$200 after 250 hands. These 84 mixtures were used to construct a 10-dimensional multivariate Gaussian using the sample mean of the 84 mixtures along with the sample covariance. This Gaussian effectively restricted the generated opponents to more

$E(W bet) > 0$	FOLD	FOLD CALL RAISE	FOLD CALL RAISE
$E(W call) > 0$	FOLD	FOLD CALL RAISE	FOLD CALL RAISE
$E(W) < 0$	FOLD	FOLD	FOLD CALL RAISE
	$p < .35$	$.35 < p < .65$	$p > .65$

Fig 1. A 10-dimensional opponent space for poker. Vertical axis is expected winnings. Horizontal axis is the probability of winning the hand based on the visible cards. Only five of the nine regions have more than one viable action, and a probability distribution over each of these has two degrees of freedom, for a total of 10 parameters in the model.

viable mixtures, mainly by reducing the likelihood of generating mixtures that fold strong hands or bet poor hands. Most of the 10 components varied significantly in value. Their variances were between 0.03 and 0.04, or about 20% on either side of the mean, with most components having virtually zero correlation with other components. This Gaussian virtually eliminated mixtures with a propensity for folding an extremely strong hand or betting an extremely weak hand. Outside of these two extreme states, both the parameter means stayed close to their raw expectation of 0.33. Using this Gaussian to sample mixture opponents, the first generation average winnings for random networks fell to about 60 percent of the money in play, so that although several random networks were still stronger than random mixtures, there was still plenty of room for training.

## V. TRAINING THE MIXTURE IDENTIFIER

The purpose of developing the opponent models is to provide computer players with a view into the nature of their opponent. In order to make this goal possible, there must be a module that can map the observable portion of the game state and the opponent's actions into an estimated opponent model. This module is the mixture identifier. Opponent models cannot be estimated directly because the crucial part of the game state – the win probability – is hidden from the player. The goal, then, is to find the best approximation to the opponent model given the information that is available to the player. The longer the match, the more information becomes available, including the win

probability in the case where a hand reaches the showdown, since players must then reveal cards.

One possibility would be to use a particle filter in order to estimate the parameters. However, since both the win probability and the opponent model are hidden, this approach is not practically feasible until multiple hands reach the showdown, at which point one could construct a reasonable observation model using the definitions of the model parameters. Also, a large number of particles would be required to obtain an accurate estimate in a 10-dimensional space, which would make the player sluggish, especially if used online during play.

A neural network was therefore trained to estimate the mapping using NEAT. While this approach requires significant time for training, during online play, an estimate of the opponent model can be obtained simply by activating the network.

The task of the mixture identifier is to estimate an opponent model describing the opponent. Random mixture models generated according to the scheme above can provide a supervised data set against which candidate mixture identifiers can be evaluated. A deterministic poker player with a conservative strategy of betting based on expected winnings was created to serve as a harness. Each time the generated mixture opponents made a decision, the mixture identifier was queried for an estimate of the correct model. Whenever the showdown was reached, the mixture identifier would be allowed to revise its estimates for that round using the estimated win probability based on the opponent's cards. The average of the models obtained in this way over the course of play was considered to be the candidate mixture identifier's best guess of the correct mixture model.

While this style of evaluation suggests that a supervised training strategy might be more effective, there are good reasons to use a semi-supervised method such as NEAT instead. First, a correct value needs to be found by settling over a sequence of network activations, which requires recurrency. Common supervised training methods for neural networks do not work well on recurrent structures, whereas NEAT naturally develops recurrent networks when these networks are better than competing non-recurrent networks. The inputs given to the mixture identifier are the same 8 inputs given to the poker players. The starting topology for the mixture identification problem in this case was a fully connected network with 8 inputs and 10 outputs, for a total of  $8 \times 10 = 80$  parameters. NEAT efficiently tunes these parameters with relatively few evaluations. The best networks found by evolution included additional inhibitory links between competing output nodes, and excitatory links between mutually reinforcing nodes. Thus solutions to the mixture identification problem were helped

by recurrency.

In the mixture identifier experiment, then, a mixture identifier network was trained to guess the mixture governing a generated opponent on average. In each generation, each candidate mixture identifier was evaluated against 50 sampled mixtures, playing 250 hands against each mixture. The fitness of the candidate was determined by calculating the Euclidean distance  $d$  of the candidate's average estimate from the actual mixture parameters controlling the generated opponents. To give better mixture identifiers higher fitness, this distance was subtracted from a distance of 3.5, chosen to be greater than the Euclidean distance from the origin to the farthest corner of the unit hypercube in 10 dimensions. The 50 values of  $(3.5 - d)$  were added up and scaled to a percentage value. In 11 trials of 50 generations each, the average maximum percentage achieved was 84.6%, or  $d = 0.56$ , an average component-wise error of 0.16. All trials except one achieved maximum fitness  $> 82\%$ . In other words, the probability estimates for opponent behavior were off by approximately 16 percent on average for each of the 10 parameters, a strong performance given the difficulty of the task. A graph of a typical run of the experiment is given in Figure 2. In this run, the maximum fitness grew from 77.8% to 84.6%, with percentages calculated based on the average Euclidean distance of mixture estimates from actual mixture values for generated opponents.

Overall, these results demonstrate that for a reasonably selected set of mixture parameters, it is possible to train a mixture identifier that provides a usable approximation of the mixed strategies employed by an opponent. The next step is to train a player to use this approximation.

## VI. TRAINING THE POKER PLAYERS

Once the mixture identifier has been trained successfully, the main experiment is to validate whether this module can provide an advantage to a computer player learning to play poker. In order to test this hypothesis, two separate networks were trained: one, termed the *mixture-based player*, took both the game state and the mixture identifier's output as an input, and another, termed the *control player*, took the only the game state as an input. Both networks were trained using NEAT. In each generation, each network was required to play 250 hands against each of 50 randomly generated mixture opponents. During each of the matches, blinds were fixed at \$2 and each player started with \$200. The fitness of each candidate network was assigned as a percentage of the \$400 won on average against all opponents.

The two networks were trained for 100 generations on 11 separate trials. At the end of training, the control player achieved an average maximum fitness of 93.0% (variance

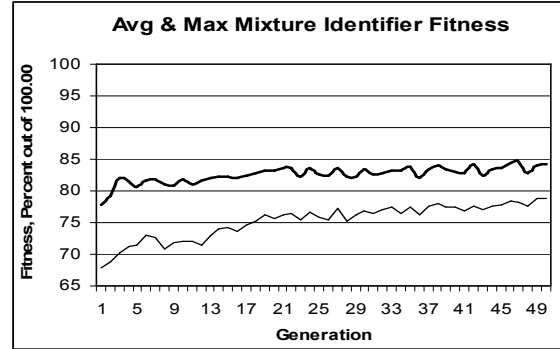


Fig 2. Average and maximum fitness graphs for a typical run of mixture identifier training. Fitness is based on the average Euclidean distance of the mixture identifier's estimate of the opponent's parameters from the actual parameters. These values are then scaled to percentage values, with 100% indicating zero distance. This training process produced mixture identifiers with an average error of about 16% for each of the opponent's parameters.

0.005%) while the mixture-based player achieved an average maximum fitness of 95.4% (variance 0.011%), with percentages indicating the percent of money won. A paired, two-tailed t-test shows an 80% chance that the mixture-based player achieves greater maximum fitness in general. Typical graphs of fitness growth for both types of players are shown in Figure 3. The fitness values are stochastic, depending both on the opponents selected and the hands drawn. Despite such stochasticity, the mixture-based player's fitness is remarkably stable, especially compared with the rather choppy oscillation observed for the control players. This result suggests that the mixture-based player's access to the mixture identifier helps smooth out the stochasticity of the fitness function. By contrast, control players had a spike in fitness (as observed in generation 66 in Figure 3) in all eleven trials, and as such control players with fitness over 90% were somewhat anomalous and failed to take over the population. Overall, during a typical training run, the maximum fitness of the mixture-based player rarely dropped below 90% after the first few generations, whereas the maximum fitness of the control player rarely exceeded 90%.

After the mixture-based and control players were trained, the two networks were played against each other. This is the true test of the mixture-based player, since the original goal for this research was to develop poker players that could model their opponents' behavior in order to generalize to unseen opponents. The competition consisted of 150 matches of 250 hands each, again starting with \$200 each and \$2 blinds. Each match was played twice with the two players holding opposite hands to eliminate any advantage due to luck. Over the 11 trials, the mixture-based player won an average 61.4% of the money, with the control

player taking the remaining 38.6% (this result is significant with  $p < 0.02$ ). The actual numbers are shown in Table 1. While the margins vary considerably, the mixture based player did manage to win more than half the money in all but one trial. These results strongly indicate that the mixture identifier does indeed allow the player to adjust to unseen opponents, significantly improving its performance.

## VII. DISCUSSION

These results demonstrate a mixture-based approach that creates low-dimensional opponent models by partitioning the state space. The 10-parameter opponent model significantly strengthens play in Texas Hold 'Em poker. In particular, using a mixture identifier improves fitness against generated opponents conforming to the model. Beyond just improving fitness, however, the mixture identifier – even a somewhat noisy one – smoothes out stochastic effects of evaluation in a random environment.

This work compares favorably with the results previously obtained in Guess It by Lockett *et al.* [3]. The mixture identifiers developed for poker have higher accuracy than those developed for Guess It, despite a three-fold increase in the number of mixture parameters. This improvement results from a rigorous scheme for generating the parameter space, as opposed to the somewhat ad hoc introduction of a fixed bank of cardinal opponents. The cardinal opponents in [3] are not independent, in that two of their four parameters overlap, whereas the 10-parameter model allows only one set of action probabilities to be active for each distinct state. Thus distinct 10-parameter models play from distinct probability distributions and can therefore be identified uniquely. Interestingly, Lockett *et al.* found that the mixture-based players for Guess It attained lower fitness than the control players, ostensibly because there are more parameters to tune. By contrast, in poker, the mixture-based players have higher fitness during training than the control players. One explanation is that poker inherently rewards opponent modeling more strongly when playing against non-optimal opponents, further strengthening the claim that opponent modeling is an important part of playing poker.

A possible criticism of opponent modeling in general is that it is only intended to maximize winnings against weak or average opponents, and not intended to produce optimal players. Opponent modeling is thus used to obtain *maximal* strategies in contradistinction to *optimal* strategies that are intended to play well against all opponents generally. However, this distinction does not hold in general, or specifically for the kind of opponent modeling in this paper. In essence, such modeling extends the game state space by appending parameters that classify the opponent. The training of the opponent-modeling player then optimizes play against representative opponents. While such methods may find local rather than global optima, the goal is still to

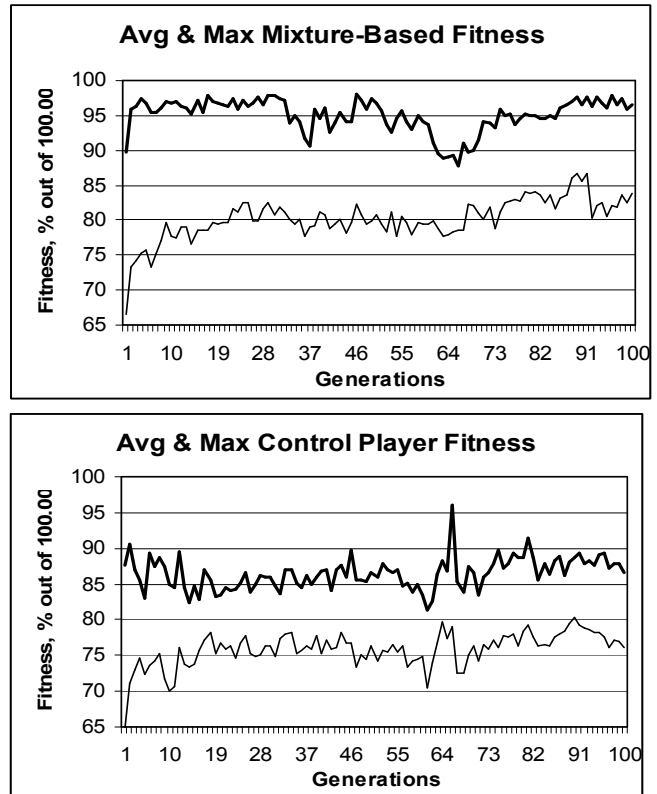


Fig 3. Fitness graphs for the mixture-based and control players on a sample trial run. Fitness is the average percentage of money won by the player. Average and maximum fitness for the mixture-based player is higher and more stable in general.

develop optimal players. In the context of game theory, Nash equilibria on the standard state space may or may not defeat Nash equilibria on the state space extended with the opponent model, depending on the game and the quality of the opponent models. For instance, Lockett *et al.* [3] present a situation in Guess It where the optimal strategy on the non-extended state space is much worse than even sub-optimal strategies on the extended state space. Thus, it is not clear that an optimal strategy in poker will defeat an opponent modeling strategy in general.

## VIII. FUTURE WORK

The mixture-based approach to opponent modeling can be further validated by taking more parameters into account. The 10-parameter opponent model only represents the

TABLE 1. PERCENTAGE OF MONEY WON BY THE MIXTURE-BASED PLAYER IN 11 TRIALS

% won										
51.6	49.2	77.5	55.1	53.2	80.2	53.2	59.6	51.2	59.4	85.1
<b>Average</b>										61.4

immediate statistical aspects of the game. If the goal is to train poker players that can model opponents in tournament play with humans, much more refined models will be needed. These models would vary their play depending on the round, the size of their bankroll, the actions of the opponent and more.

Looking further forward, there are several ways to build on this promising approach. One obvious area is to develop models automatically from records of human play. Transcripts of poker matches are widely available online, and the premier event of the game, the World Series of Poker, has been televised for several years. It may be possible to develop an algorithm that partitions the state space according to some objective criterion drawn from data sets of human play, such as maximizing the Kullback-Leibler divergence between distinct regions on a given data set. This method would allow more realistic opponents to be generated, and the techniques employed by a human player to be measured more accurately, which should lead to stronger automatic poker players.

Another point of departure involves using higher dimensional parameter spaces with a small set of opponent clusters within this space. The higher dimensional space can be used to generate mixture opponents that can play more refined strategies. The mixture identifier would map opponents according to its likelihood of belonging to each of the clusters. These clusters would represent certain normative playing patterns, balancing the need to generate more detailed opponents with the need for low-dimensional opponent representations. In this way, more complex models could be naturally handled using the methods of this paper.

A fourth category of extensions examines probability distributions over opponents. For instance, a data set including play from average poker players would likely have very different characteristics from a set of tournament transcripts. When training a computer to play poker, it is important that generated opponents represent the set of players the computer is likely to encounter, possibly with greater weight placed on more difficult opponents. One might also develop a training program that increases in difficulty over time, with simpler opponents appearing more often in lower generations, and with successive generations moving through a Pareto dominance hierarchy.

Extensions such as these may eventually lead to computer players that play at human levels and even exceed them. Similar techniques of opponent modeling could also bear fruit in other games and in multi-agent systems generally, leading to intelligent agents that interact more effectively with other agents in their environment to improve their decision making ability.

## IX. CONCLUSION

This study shows that opponent modeling using the mixture approach is practical and beneficial, resulting in increased fitness of players trained to play Texas Hold 'Em

poker. The mixture approach consists of identifying and defeating previously unknown opponents by representing them as a mixture over a low-dimensional parameter space that approximates objective aspects of the opponent's play. Mixture-based opponent models are effective because they give computer players insight into aspects of the game that would otherwise be hidden from them, such as deceptive or misleading play. The same process leveraged to obtain these results should apply not only to poker, but also to many environments where there is a need to understand the intent or purpose of other agents, making opponent modeling an increasingly important component of AI research in general.

## ACKNOWLEDGMENTS

The authors wish to thank Charles L. Chen for his assistance and contributions to this research. This research was supported in part by NSF grant IIS-0757479 and THECB grant 003658-0036-2007.

## REFERENCES

- [1] Billings, D., Papp, D., Schaeffer, J., and Szafron, D. Opponent Modeling in Poker. *Proceedings of 15<sup>th</sup> National Conference of the American Association on Artificial Intelligence*. AAAI Press, Madison, WI, 1998, 493-498.
- [2] Davidson, A., Billings, D., Schaeffer, J., and Szafron, D. Improved Opponent Modeling in Poker. *Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI'2000)*. 1999, 1467-1473.
- [3] Lockett, A., Chen, C., and Miikkulainen, R. Evolving Explicit Opponent Models in Game Playing. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-07)*. Kaufmann, San Francisco, 2007, 2106-2113.
- [4] Korb, K., Nicholson, A., and Jitnah, N. Bayesian Poker. *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI-99)*. 1999, 343-350.
- [5] Southey, F., Bowling, M., Larson, B., Piccione, C., Burch, N., and Billings, D. Bayes' Bluff: Opponent Modeling in Poker. *Proceedings of the 21<sup>st</sup> Conference on Uncertainty in Artificial Intelligence (UAI-05)*. 2005, 550-558.
- [6] Davidson, A. Using Artificial Neural Networks to Model Opponents in Texas Hold 'Em. Unpublished manuscript; <http://spaz.ca/aaron/poker/nnpoker.pdf>. 1999.
- [7] Bard, N. and Bowling, M. Particle Filtering for Dynamic Agent Modeling in Simplified Poker. *Proceedings of the 22<sup>nd</sup> Conference on Artificial Intelligence*. AAAI Press, Madison, WI, 2007, 515-521.
- [8] Stanley, K. and Miikkulainen, R. Continual Coevolution Through Complexification. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*. Kaufmann, San Francisco, 2002, 113-120.
- [9] Barone, L. and While, L. Adaptive Learning for Poker. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*. Kaufmann, San Francisco, 2000, 566-573.
- [10] DiPietro, A., Barone, L., and While L. Learning In RoboCup Keepaway Using Evolutionary Algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*. Kaufmann, San Francisco, 2002, 1065-1072.
- [11] Hoehn, B., Southey, F., Holte, R. C., and Bulitko, V. Effective Short-Term Opponent Exploitation in Simplified Poker. *Proceedings of the 20<sup>th</sup> National Conference on Artificial Intelligence*. AAAI Press, Madison, WI, 2007, 783-788.
- [12] Riley, P., and Veloso, A. Planning for Distributed Execution Through Use of Probabilistic Opponent Models. *IJCAI-2001 Workshop PRO-2: Planning under Uncertainty and Incomplete Information*. 2001.