Opponent Modeling and Exploitation in Poker Using Evolved Recurrent Neural Networks

Xun Li
University of Texas at Austin
xun@cs.utexas.edu

Risto Miikkulainen
University of Texas at Austin
risto@cs.utexas.edu

ABSTRACT

As a classic example of imperfect information games, Heads-Up No-limit Texas Holdem (HUNL) has been studied extensively in recent years. While state-of-the-art approaches based on Nash equilibrium have been successful, they lack the ability to model and exploit opponents effectively. This paper presents an evolutionary approach to discover opponent models based on recurrent neural networks (LSTM) and Pattern Recognition Trees. Experimental results showed that poker agents built in this method can adapt to opponents they have never seen in training and exploit weak strategies far more effectively than Slumbot 2017, one of the cutting-edge Nash-equilibrium-based poker agents. In addition, agents evolved through playing against relatively weak rule-based opponents tied statistically with Slumbot in heads-up matches. Thus, the proposed approach is a promising new direction for building high-performance adaptive agents in HUNL and other imperfect information games.

CCS CONCEPTS

• **Machine learning** → **Machine learning approaches** → Neural networks; Bio-inspired approaches→ Genetic algorithms

KEYWORDS

Decision-making, Games, Genetic Algorithms, Neural Networks

1 INTRODUCTION

Imperfect information games are an important AI problem with numerous real-world applications, such as trading, business negotiation, security, military decision-making, and table games.

As a classic imperfect information game, Texas Holdem has been studied extensively in recent years. Nash-equilibrium-approximation techniques, e.g. Counterfactual Regret Minimization (CFR) [1], have been applied to multiple variants of the game and achieved remarkable successes. Heads-Up Limit Texas Holdem has been weakly solved [2]. Many powerful poker agents for Heads-Up No-Limit Holdem (HUNL) have been built through CFR in combination with various abstraction and/or sampling techniques [3-5]. In recent Human vs. AI competitions, poker agents using approximated Nash equilibrium strategies have defeated professional human players in HUNL with statistically significant margins [6,7].

However, Nash-equilibrium-approximation approaches have three limitations.

First, for imperfect information games with large state space, the quality of approximated Nash equilibrium strategies can be far from ideal. As an example, in HUNL, most top-ranking agents using approximated equilibrium strategies are exploitable with a simple local best response method [8].

Second, although a Nash-equilibrium strategy in a two-player zero-sum game is unexploitable in theory, it does not guarantee maximum utility. In particular, even if a real Nash equilibrium strategy is found for HUNL, it is unlikely to be the most profitable strategy. The best counter-strategy against each opponent is different, and only through opponent modeling and adaptation can a player approximate such counter-strategies to exploit various opponents effectively.

Third, in imperfect information games with more than two players and multiple equilibria, if the opponents are not following the same equilibrium as approximated by an equilibrium-based agent, the agent's performance cannot be guaranteed [9]. Therefore, Nash-equilibrium-approximation approaches may not be effective in applications with three or more players.

Therefore, developing a new method for building adaptive poker agents that is not based on Nash equilibrium strategies may be the next step towards building stronger computer agents for HUNL.

Adaptive agents with opponent models can exploit their opponents' weaknesses effectively, thereby achieving high utility. In addition, if the method does not attempt to approximate Nash equilibrium strategies, the limitations of approximation quality and generalizability no longer apply.

This paper proposes such a method: opponent models based on Pattern Recognition Trees and LSTM neural networks are constructed through evolutionary optimization; such models are

GECCO'18, July 15-19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5618-3/18/07. $15.00
http://doi.org/10.1145/3205455.3205589

then integrated with decision algorithms to build poker agents that are able to adapt their behaviors to exploit opponents' weaknesses. Via this approach, a poker agent called Adaptive System of Holdem (ASHE) is constructed. Experimental results show that ASHE exploits highly to moderately exploitable opponents far more effectively than Slumbot 2017, one of the state-of-the-art poker agents based on Nash equilibrium approximation, and ties with it statistically in HUNL matches. In addition, since the proposed approach is not based on Nash equilibrium, the lack of performance guarantee of equilibrium strategies in games with more than two players does not limit its application. Thus, the paper provides an alternative method for building high-quality computer poker agents and points out a promising new direction of research in imperfect information games.

The remaining sections of this paper are organized as follows. Section 2 outlines prior work on opponent modeling in poker. Section 3 presents the architecture of ASHE, and introduces the genetic algorithm to evolve its opponent model. Section 4 presents experimental results and discusses ASHE's performance both against highly to moderately exploitable players and Slumbot 2017. Section 5 suggests directions for future work.

2 RELATED WORK

To achieve high performance in an imperfect information game such as poker, the ability to effectively model and exploit suboptimal opponents is critical.

The initial attempts to construct adaptive poker agents employed rule-based statistical models. For instance, the Loki system was able to discover and exploit certain patterns in its opponents' strategies [10]. Billings et al. [11] proposed opponent modeling with adaptive game tree algorithms: Miximax and Miximix, which computed the rank histogram of opponent's hand for different betting sequences, categorizing them as one of a broader hand groups. Bard et al. [12] proposed an end-to-end approach for building an implicit modeling agent. In addition, Bayesian model [13-15], clustering [16], and ensemble learning [17] were introduced for opponent modeling in different variants of poker.

Researchers also attempted to build adaptive poker agents by adjusting Nash-equilibrium strategies. Ganzfried and Sandholm [18] proposed an efficient real-time algorithm that observed the opponent's action frequencies and built an opponent model by combining information from an approximated equilibrium strategy with the observations. Statistical exploitation module [19], robust counter strategies [20-22], and safe exploitation techniques were proposed [23] to make adaptive agents less exploitable. While these techniques improve adaptive agents' performance against high-quality equilibrium-based opponents, they also limit adaptation and are unable to fully exploit opponent strategies.

Neural networks were introduced to model opponents by Davidson et al. [24]. The networks were used to assign each game state with a probability triple, predicting the action of the opponent in Limit Texas Holdem. The prediction had an accuracy of over 80%. Lockett and Miikkulainen [25] proposed a method to evolve neural networks to classify opponents in a continuous space. Poker agents were trained through neuroevolution both with and without the opponent models, and the players with the models conclusively outperformed the players without them.

However, most of the above methods were applied to and evaluated in Heads-Up Limit Holdem or simpler poker variants. In particular, none of them has been demonstrated experimentally to be effective in building adaptive agents for HUNL, a far more challenging poker variant due to much bigger state space.

In recent years, a type of recurrent neural network, Long Short Term Memory (LSTM) [26] has achieved great successes in many challenging AI problems [27]. Li and Miikkulainen [28] proposed to build poker agents based on LSTM for HUNL. Such agents extracted action patterns from game state sequences and modeled the opponent implicitly. The agents exploited various weak strategies effectively, but did not achieve comparable performance in HUNL matches against cutting-edge Nash-equilibrium-based agents.

This paper proposes to use LSTM neural networks and Pattern Recognition Trees to build poker agents that model the opponents explicitly. Poker agents built in this method were able to adapt to their opponents dynamically, even to those with stronger strategies than they have seen in training. In the experiments, these agents tied against Slumbot 2017, the best equilibrium-based agent that was accessible as a testing opponent, in HUNL matches. In addition, they were far more effective in exploiting highly to moderately exploitable opponents than Slumbot 2017.

Thus, this paper is an important step towards effective opponent modeling in HUNL and points out a promising new direction for building high-performance adaptive computer agents in imperfect information games.

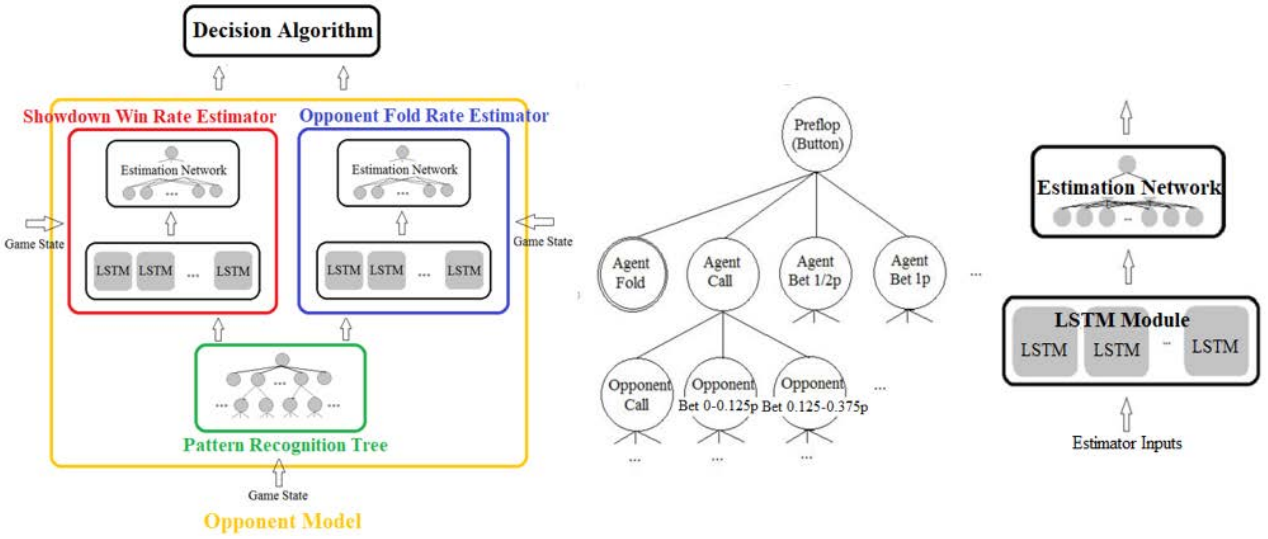
3 METHOD

This section introduces an evolutionary method for building adaptive poker agents that utilizes opponent models based on Pattern Recognition Trees and LSTM neural networks. Subsection 3.1 details the architecture of ASHE, including the Pattern Recognition Trees, the LSTM estimators, and the Decision Algorithm. Subsection 3.2 introduces the method for evolving the opponent models.

3.1. ASHE Architecture

ASHE is an adaptive poker agent designed for HUNL with a dynamic strategy based on opponent modeling. Fig. 1 (a) presents the overall architecture of ASHE.

The core of ASHE is the opponent model, which contains the Pattern Recognition Trees (PRTs) and two LSTM estimators. The PRTs maintain of a set of statistics for each sequence of actions seen in games against an opponent, collecting information on the opponent's strategy from every game. Both of the estimators receive input features extracted from the current game state and the PRTs. The Showdown Win Rate Estimator evaluates the



(a) Overall Architecture **(b) Pattern Recognition Tree** **(c) LSTM Estimator**

Figure 1: ASHE Architecture. ASHE consists of two parts: a rule-based decision algorithm and an opponent model. The opponent model has three components: Showdown Win Rate Estimator, Opponent Fold Rate Estimator, and Pattern Recognition Tree. The PRT and LSTM estimators allow ASHE to make decisions based on current game state and patterns learned from previous games.

probability of ASHE holding a hand better than the opponent's. The Opponent Fold Rate Estimator estimates the probability of the opponent folding to a raise (bet).

These estimations are sent to the Decision Algorithm, which evaluates the expected utility for each possible action based on statistical estimation and selects the action with the highest expected utility.

3.1.1 Pattern Recognition Tree. The PRT collects data on action sequences seen in all games against an opponent and organizes the data to allow fast retrieval given any game state. Such data is used to extract input features for the LSTM estimators. Fig. 1 (b) illustrates its structure.

The root of the tree represents an initial state of the game; it can be either preflop on button or preflop as the Big Blind, with no action taken by either player. Thus, in a typical HUNL match, two PRTs are created, each corresponding to one of the two positions (i.e. the Button or the Big Blind). Leaf nodes represent terminal states of the game. They can be either a showdown or a fold from a player. Non-leaf nodes represent decision points where either ASHE or its opponent must make the next move. For each non-leaf node, the path from the root to the node represents a sequence of actions from the two players. ASHE's bets are restricted to 0.5, 1.0, 1.5, 2.0 pot size, and all-in. The opponent's bets are discretized into seven buckets: (0, 0.125), (0.125, 0.375), (0.375, 0.75), (0.75, 1.25), (1.25, 2.0), (2.0, 4.0), (> 4.0), also with respect to the size of the pot.

Each non-leaf node is associated with a set of statistics called node stats, including the frequency of the node being visited (f), the opponent's average hand strength in showdowns after visiting the node ($\bar{s}$), and the number of showdowns (c_{sd}) and the opponent's folds (c_{of}) after reaching the node. These statistics outline the tendency of opponent actions at different decision points. For instance, if the opponent is over-cautious when facing consecutive raises at both the flop and the turn, the cof will be high for every node along that path, suggesting an opportunity for bluffing.

At the beginning of a match, two PRTs corresponding to each position (i.e. the button and the Big Blind) are initialized with only a root node. Nodes are inserted to the trees when their corresponding decision points occur for the first time in the match. After each game, node stats for every node along the path of the game (from the root to the leaf) are updated based on the result of the game. As the match proceeds, the PRTs grow, and the node stats become increasingly reliable, especially for decision points that are frequently visited. Note that because of action restriction and bucketing, the trees grow slowly after a few dozens of games, and most nodes are visited repeatedly.

To control the size of the trees and ensure that most nodes are visited frequently enough to provide meaningful node stats, PRTs in the current version of ASHE do not distinguish decision points by community cards. Information regarding community cards are encoded in the input features of the estimators.

3.1.2 LSTM Estimators. The estimators provide essential information for the Decision Algorithm to evaluate the expected utility of each possible action. The Showdown Win Rate Estimator (SWRE) estimates the probability of ASHE's hand beating its opponent's if a showdown occurs. The Opponent Fold Rate Estimator (OFRE) estimates the probability of the opponent folding its hand if ASHE bets or raises. Both the SWRE and the OFRE share the same LSTM-based neural network architecture, illustrated by Fig. 1 (c).

Table 1: Estimator Inputs

Feature Name	Definition	Input
Normalized Frequency	$\sigma(f/10)$, where σ is the sigmoid function.	S & F
Fold Rate	$(c_{\text{of}}+1)/(f+3)$, smoothed by $1/3$.	S & F
Showdown Rate	$(c_{\text{sd}}+1)/(f+3)$, smoothed by $1/3$.	S
Expected Strength	$\overline{s}$, if $c_{\text{sd}} > 5$, $(0.85 \cdot (5-c_{\text{sd}}) + \overline{s}c_{\text{sd}})/5$, otherwise.	S
Flush and Straight Draw	Probability of a random hand hitting flush or straight given the board.	F
Pair(s)	0: no pair on board, 0.5: one pair on board, 1.0: two pairs on board.	F
Betting Round	Binary indicators of the four rounds, i.e. preflop, flop, turn, and river.	S & F
Raw Hand Strength	Probability of ASHE's hand beating a random hand given the board.	S
Opponent Total Bet	Opponent's total bet normalized by the initial stack size.	S & F
ASHE Total Bet	ASHE's total bet normalized by the initial stack size.	S & F

In the current version of ASHE, each LSTM module contains 50 vanilla LSTM blocks as described by Greff et al. [27]. Each block contains ten cells. These LSTM blocks are organized into a single layer. The estimation network is a fully connected feed-forward neural network with one hidden layer.

All LSTM blocks are reset to their initial states at the beginning of each hand. Given a decision point, a set of input features are derived for each estimator from the game state and the node stats of the corresponding node. If the node does not exist, a set of default inputs indicating an unknown state are used instead.

Table 1 defines the input features. The first four input features are derived from node stats in the PRTs, which provides statistical information on the opponent's strategy given the same action sequence in the past. The rest of the features provide information on the current state of the game. Notations of node stats are the same as defined in the previous subsection. S and F refers to the SWRE and the OFRE, respectively.

These features collectively not only encode the current game state but also reflect the history of previous games against the opponent. These input features are sent to the LSTM modules in the estimators, whose cell states and outputs after receiving the inputs are saved as the *decision states*.

To compute estimations for an action (e.g. the probability for the opponent folding after a pot-size raise), input features from the PRT node corresponding to that action are also extracted. The LSTM modules take these inputs and send their outputs to the estimation network, which produces the estimation. After that, the LSTM modules are restored to the decision states to compute estimation for other actions. Thus, the estimators process and balance information from each decision point and compute estimations based on the states in the current game and node stats from the PRT.

3.1.3 Decision Algorithm. Fig. 2 outlines the decision algorithm. Parameter b_{opp} and b_{ASHE} refer to total chips committed in the pot by the opponent and ASHE, respectively. When estimating the expected utility of an action, the decision algorithm makes two assumptions.

Input: possible actions A = {a_i}
Output: decision a_d

Utility u = -infinity, d = 0
For each action a_i in A
  If a_i is Fold
    $u_i = -b_{\text{ASHE}}$
  If a_i is Call
    Estimate win rate w_{sd} by SWRE
    $u_i = (2w_{\text{sd}} - 1)b_{\text{opp}}$
  If a_i is Raise
    Estimate win rate w_{sd} by SWRE
    Estimate fold rate f_{opp} by OFRE
    $u_i = (1 - f_{\text{opp}})(2w_{\text{sd}} - 1)(b_{\text{ASHE}} + r) + f_{\text{opp}}b_{\text{opp}}$
  If $u_i > u$
    $u = u_i$, $d = i$
Return a_d

Figure 2: Decision Algorithm. The decision algorithm estimates the expected utilities of each available action and selects the best move. The estimations are based on the outputs of the LSTM estimators and two assumption.

First, it assumes that the estimations from the LSTM estimators are the true values. Second, it assumes that if the opponent does not fold to a raise, it will call, and the game will go to a showdown with no more chips committed to the pot. Although neither of these assumptions always hold in reality, experimental results indicate that the utility estimations based on the decision algorithm is an acceptable approximation.

In addition, a restriction is placed on ASHE's actions: ASHE responds to a four-bet (when playing as the Big Blind) or a three-bet (when playing as the Button) with a fold, a call, or an all-in.

This restriction simplifies utility estimation and reduces the size of the PRTs.

In sum, the ASHE architecture uses the PRTs to collect statistical information on the opponent's moves given different at decision points. The LSTM estimators consider such information and the game state at each decision point in the current hand to estimate ASHE's winning probability in a showdown and the probability of the opponent folding to bets/raises. The decision algorithm uses these outputs to approximate the expected utility of available actions and selects the best move accordingly.

3.2 Evolving the LSTM Estimators

The performance of ASHE depends on the accuracy of the LSTM estimators. Although it is clear what the LSTM estimators need to do, how can they be trained to do so? The correct outputs for the estimators, e.g., the accurate estimation of an opponent's folding probability given a game state and the history of previous games, are not known. Therefore, they cannot be trained through supervised methods. A major technical insight in this paper is that it is possible to evolve the LSTM estimators for their tasks through genetic algorithms. This allows the method to be applied to problems with little or no labeled training data.

In the genetic algorithm, a population of agents are created with randomly initialized LSTM estimators at first. They are evaluated based on their fitness, i.e. their average earnings against a variety of training opponents. These opponents require different counter-strategies for effective exploitation. Thus, agents with adaptive behaviors that can exploit various strategies are rewarded by higher fitness.

The agents with the highest fitness survive and reproduce via mutation and crossover. The others are replaced by the offspring. Selection and reproduction are done iteratively, thus improving the overall fitness of the population. Note that the agents do not have to know what the accurate estimations are or what actions are the best; it is sufficient to select agents that perform relatively well in the population.

Specifically, during evolution, agents are represented by theirs numeric genomes, which are constructed by concatenating all parameters in the estimators, i.e. the weights in the LSTM module, the initial states of all LSTM blocks, and the weights of the estimation network. To preserve advantageous genotypes against occasional poor luck, Tiered Survival and Elite Reproduction is adopted. Survivors are divided into two tiers based on their fitness. Agents in the top tier can reproduce and do not mutate. Agents of the second tier mutate but cannot reproduce. Thus, genotypes from more agents can be preserved without compromising the performance of the offspring [28].

Evolving agents with multiple modules can be inefficient, especially at an early stage. Advantageous genotypes of one module may be coupled with poor genotypes of another, causing low performance and loss of progress. Therefore, evolution is organized into a series of sessions, with each session dedicated to evolving one of the two estimators. When a new session begins, the champion of the previous session is duplicated to form the agent population. The weights in the estimator to be evolved are mutated, and the weights in the other estimator frozen. Thus, the genetic algorithm is able to discover effective LSTM estimators efficiently, even though there is no data for supervised training.

4 EXPERIMENTAL RESULTS

In the experiments, agents were evolved for 500 generations, and the champion at that point were used to evaluate the proposed method. The population size was 50. Mutation followed Gaussian distribution. As evolution proceeded, mutation rate descended linearly from 0.25 to 0.1, and mutation strength (variance) from 0.25 to 0.05. After each generation, agents with top 30% performance survived, and survivors whose performance was above the mean performance among all survivors were in the top tier (as defined by subsection 3.2). Note that these settings should be considered as a reference, and reasonably different settings should not affect overall performance noticeably.

Table 2: Training and Testing Opponents

Name (Code)	Exploitability	Description
Hothead Maniac (HM)	Very High	Always raises by half or one pot randomly.
Calling Machine (CM)	Very High	Always checks or calls.
Scared Limper (SL)	High	Calls when holding the nut; else, folds to any bet.
Candid Statistician (CS)	High	Bets 1/4 to one pot depending on raw hand strength, checks/calls with marginal hands, folds weak hands.
Random Gambler (RG)	High	Randomly switching from other highly exploitable strategies every 50 hands.
Loose Aggressive (LA)	High	Bets/raises aggressively with a wide range of hands.
Loose Passive (LP)	High	Calls with most hands, folds weak hands, rarely raises.
Tight Passive (TP)	High	Calls with good hands, folds most hands, rarely raises.
Tight Aggressive (TA)	Moderate	Similar to CS, with refined hand ranges and bluffing.
Half-a-Pro (HP)	Low	TA with advanced strategies such as floating, check-raising, and multi-shot bluffing.
Slumbot 2017 (SB)	Very Low	Updated version of the runner-up in ACPC 2016, a cutting-edge equilibrium-based poker agent.

In each generation, agents played two matches against every opponent. Both matches contained 1000 hands. Card decks in the first match were duplicated and played again in the second match. The players switched seats between the two matches, and the PRTs of the agents were reset before entering the second match. At the beginning of each hand, the chip stacks of the two players were reset to \$20,000. The blinds were \$50/\$100. The format of the game was consistent with HUNL matches in the Annual Computer Poker Competition (ACPC).

Table 2 describes the agents used for evolving and testing ASHE. HM, CM, SL and CS are four agents with highly exploitable strategies that require different (and in some cases, opposite) counterstrategies. LA, LP, TP, and TA are less exploitable versions of HM, CM, SL, and CS, respectively. RG, HP, and SB were used only for testing. RG is highly exploitable, but its strategy is dynamic. HP is a strong agent based on common tactics and strategies adopted by professional players. Slumbot 2017 (SB; https://www.slumbot.com) was the best Nash-equilibrium-based agent that was publicly available at the time of the experiments.

These agents form a pool of training and testing opponents with different weakness types and exploitability levels. To investigate the influence of training opponents on ASHE's performance, four instances of ASHE were evolved with different evolution settings.

The agent A1 was evolved by playing against a group of highly exploitable opponents with diversified strategies, i.e. HM, CM, SL and CS. The agent A2 was evolved by playing against less exploitable opponents with similarly diversified weaknesses, i.e. LA, LP, TP, and TA. The agent A3 was evolved against two pairs of opposite strategies (i.e maniac and passive) with different level of exploitation, HM, SL (relatively high exploitability), and LA, TP (relatively low exploitability). The agent A4 was evolved against HM, CM, SL and CS (the same as A1) for the first 200 generations, and LA, LP, TP, and TA (the same as A2) for the other 300 generations.

Table 3 presents the test results. The first four rows show the performance of A1, A2, A3, and A4 against the eleven opponents in Table 2. The last row shows Slumbot's performance against the same opponents for comparison. The last column of the second part of the table contains the agents' performance against Slumbot. Results are in blue if an agent played against the opponent during evolution. Performance is measured in milli-Big-Blind per hand (mBB/hand). The best agent, A4, ties with Slumbot and is far more effective in exploiting weaker opponents.

Every pair of agents in the table played at least 20,000 hands against each other. Another 20,000 hands were played if the margin was greater than 20% of the absolute value of the mean. Thus, approximately 1.5 million hands were played during evaluation. These hands were organized into 1000-hand matches, and ASHE's estimators were reset between consecutive matches. The rest of this subsection points out key discoveries and conclusions from these results.

First, overall, the best of the four ASHE instances is A4, followed by A2, A1, and A3 in decreasing order of their average performance.

Second, both A4 and the second best instance, A2, defeated HP by a significant margin and tied statistically with SB. These results demonstrate that ASHE, evolved by playing against relatively weak opponents, can adapt to strong opponents that are not seen during training and achieve competitive performance against top-ranking Nash equilibrium-based agents.

Third, A4 outperformed SB 40% to 2300% when facing weaker opponents. The more exploitable the opponent was, the bigger the margin tended to be. Thus, the proposed method can build poker agents that are significantly more effective in exploiting weak opponents than top-ranking agents following equilibrium strategies.

Table 3: Evaluation Results

	HM	CM	SL	CS	RG	HP
A_1	44241 ± 6111	48361 ± 5012	999 ± 1.2	9514 ± 1515	6992 ± 600	37 ± 56
A_2	40010 ± 5912	45703 ± 4998	998 ± 1.5	8071 ± 1390	8422 ± 709	225 ± 62
A_3	47086 ± 6539	22133 ± 2407	999 ± 0.3	1954 ± 286	5699 ± 545	-331 ± 73
A_4	42333 ± 6037	46114 ± 4985	999 ± 1.4	9116 ± 1403	8996 ± 721	278 ± 59
SB	4988 ± 881	2761 ± 355	702 ± 59	4512 ± 472	2102 ± 393	152 ± 50

	LA	LP	TP	TA	SB
A_1	13506 ± 988	9060 ± 598	1252 ± 70	107 ± 54	-59 ± 77
A_2	18241 ± 975	12444 ± 670	1412 ± 81	431 ± 90	-8 ± 55
A_3	23074 ± 1021	4781 ± 523	1493 ± 82	-106 ± 65	-512 ± 66
A_4	20005 ± 1006	15372 ± 634	1488 ± 91	509 ± 88	5 ± 61
SB	2449 ± 460	623 ± 41	603 ± 51	284 ± 49	—

Fourth, the performance of A3 was much worse compared to the other three instances when facing opponents whose strategies were fundamentally different from those seen during evolution (e.g. CM and CS are fundamentally different from both HM and SL). Nevertheless, instances evolved by playing against a reasonably diversified set of strategies, e.g. A1, A2, and A4, could adapt to HP or SL, whose strategies were superior to any strategies seen during evolution. These results indicate that a reasonable level of diversity of opponent strategies during evolution is necessary for evolving strong adaptive agents. With such diversity, however, agents evolved in the proposed method can generalize remarkably well to opponents that are not seen during evolution.

Fifth, both A2 and A4 demonstrated consistently better performance than A1 when playing against opponents that had not been seen during evolution, despite that their opponents during evolution had similar types of weaknesses. Hence, the level of exploitability of opponent strategies during evolution is another factor that influences ASHE's performance. A group of moderately exploitable opponents with similarly diversified weaknesses tends to be more effective in evolving strong adaptive agents than a group of highly exploitable opponents.

Sixth, ASHE's performance against RG show that agents built through the proposed approach can model and exploit opponents with a changing strategy effectively. However, game log analysis indicated that ASHE did not switch its strategy as RG did after every 50 hands; rather, as more hands were played, ASHE reached a strategy that exploited RG based on the overall distribution of its moves across the highly exploitable strategies it might randomly choose. That strategy, while being close to static after a few hundreds of hands, was much more effective in exploiting RG compared to SB's approximated equilibrium strategy.

The above observations and findings show that the proposed approach is effective in building high-quality adaptive agents for HUNL. They also provide some useful principles for its application and improvement in the future.

5 DISCUSSION AND FUTURE WORK

The proposed method for opponent modeling is not based on equilibrium strategies; it allows free adaptation and potentially more effective opponent exploitation. In addition, this method does not depend on any property of Nash Equilibria. Hence, the core idea of ASHE's architecture and the genetic algorithm to evolve it are applicable to a wider range of games. This work can be extended in four directions in the future.

First, ASHE's performance can be improved by training against stronger opponents. Second, deep neural networks can be used to extract features from sequences of raw game states, thus reducing reliance on handcrafted features. Third, neural network structures can be optimized via evolution using NEAT (Stanley and Miikkulainen 2002). Fourth, the proposed method can be applied to build adaptive agents for other imperfect information games.

6 CONCLUSIONS

This paper introduces an evolutionary approach for opponent modeling and exploitation in HUNL based on Pattern Recognition Trees and LSTM Neural Networks. Using this method, a poker agent called ASHE is built. Experimental results show that (1) ASHE is effective in modeling and exploiting weak opponents, and (2) ASHE evolved by playing against highly to moderately exploitable opponents is capable of modeling top equilibrium-based agents, achieving competitive performance in matches against them. Thus, the paper presents a new method for building high-quality adaptive poker agents, and points out a promising direction for research in imperfect information games.

ACKNOWLEDGMENTS

This research was supported in part by NSF grants DBI- 0939454 and IIS-0915038, and in part by NIH grant R01- GM105042.

REFERENCES

[1] Zinkevich, M., Johanson, M., Bowling, M., and Piccione, C. 2008. Regret Minimization in Games with Incomplete Information. In *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems* (NIPS).

[2] Bowling, M., Burch, N., Johanson, M., and Tammelin, O. 2015. Heads-up Limit Hold'em Poker is Solved. *Science*, 347 (6218).

[3] Gilpin, A., and Sandholm, T. 2008. Solving Two-Person Zero-Sum Repeated Games of Incomplete Information. In *Proceedings of the Seventh International Conference on Autonomous Agents and Multi-Agent Systems*.

[4] Brown, N., Ganzfried, S., and Sand-holm, T. 2015. Hierarchical Abstraction, Distributed Equilibrium Computation, and Post-Processing, with Application to a Champion No-Limit Texas Hold'em Agent. In *Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems*.

[5] Jackson, E. 2017. Targeted CFR, in *AAAI Work-shop on Computer Poker and Imperfect Information*, San Francisco, CA.

[6] Moravcik, M., Schmid, M., Burch, N., Lisy, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., and Bowling, M. 2017. DeepStack: Expert-Level Artificial Intelligence in No-Limit Poker, *axXiv:1701.01724*.

[7] Brown, N. and Sandholm, T. 2017. Safe and Nested Endgame Solving for Imperfect-Information Games. In *the AAAI workshop on Computer Poker and Imperfect Information Games*.

[8] Lisy, V. and Bowling, M. 2016. Equilibrium Approximation Quality of Current No-Limit Poker Bots, *arXiv preprint arXiv:1612.07547*.

[9] Ganzfried, S. 2016. Bayesian Opponent Ex-ploitation in Imperfect-Information Games, in *AAAI Workshop on Computer Poker and Incomplete Information*.

[10] Billings, D., Papps, D., Schaeffer, J., and Szafron, D. 1998. Opponent Modeling in Poker. In *Proceedings of the Joint Conference of AAAI/IAAI*.

[11] Billings, D. 2006. Algorithms and Assessment in Computer Poker. Ph.D. Dissertation, University of Alberta.

[12] Bard, N., Johanson, M., Burch, Neil., and Bowling, M. 2013. Online Implicit Agent Modeling. In *Proceedings of the 12th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2013)*, Saint Paul, Minnesota, USA.

[13] Korb, K., Nicholson, A., and Jitnah, N. 1999. Bayesian Poker. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 343-350. Morgan Kaufmann Publishers Inc.

[14] Southey, F., Bowling, M., Larson, B., Piccione, C., Burch, N., Billings, D., and Rayner, C. 2005. Bayes Bluff: Opponent Modelling in Poker. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence*.

[15] Posen, M., Ramon, J., Croonenborghs, T., Driessens, K., and Tuyls, K. 2008. Bayes Relational Learning of Opponent Models from Incomplete Information in No-Limit Poker. In *Twenty-third Conference of the Association for the Advancement of Artificial Intelligence (AAAI-08)*.

[16] Teofilo, L. F., Reis, L. P. 2011. Identifying Player's Strategies in No Limit Texas Hold'em Poker through the Analysis of Individual Moves, In *Proceedings of the 15th Portuguese Conference on Artificial Intelligence*. Lisbon, Portugal.

[17] Ekmekci, O., and Sirin, V. 2013. Learning Strategies for Opponent Modeling in Poker. In *Proceedings of Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*. Bellevue: AAAI Press (pp. 6-12).

[18] Ganzfried, S. and Sand-holm, T. 2011. Game Theory-Based Opponent Modeling in Large Imperfect-Information Games. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems*.

[19] Norris, K. and Watson, I. 2013. A Statistical Exploitation Module for Texas Hold'em and Its Benefits When Used With an Approximate Nash Equilibrium Strategy. In *Proceedings of IEEE Conference on Computational Intelligence and Games*, Niagara, Canada.

[20] Johanson, M. and Bowling, M. 2009. Data Biased Robust Counter Strategies, In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[21] Johanson, M., Zinkevich, M., and Bowling, M. 2008. Computing Robust Counter-Strategies. In *Advances in Neural Information Processing Systems 20 (NIPS)*.

[22] Ponsen, M., Jong, S., and Lanctot, M. 2011. Computing Approximate Nash Equilibria and Robust Best Responses Using Sampling. In *Journal of Artificial Intelligence Research*.

[23] Ganzfried, S. and Sandholm, T. 2015. Safe Opponent Exploitation. In *ACM Transactions on Economics and Computation*, 3(2).

[24] Davidson, A., Billings, D., Schaeffer, J., and Szafron, D. 2000. Improved Opponent Modeling in Poker. In *International Conference on Artificial Intelligence, ICAI'00*.

[25] Lockett, A., and Miikkulainen, R. 2008. Evolving Opponent Models for Texas Hold'em. In *Proceedings of the IEEE Conference on Computational Intelligence in Games*, IEEE Press.

[26] Hochreiter, S. and Schmidhuber, J. 1997. Long short-term memory. In *Neural Computation 9*, pp 1735–1780.

[27] Greff, K., Srivastava, R. K., Jan, K., Steunebrink, B. R., and Schmidhuber, J. 2015. LSTM: a Search Space Odyssey. *arXiv preprint arXiv:1503.04069*.

[28] Li, X. and Miikkulainen, R. 2017. Evolving Adaptive Poker Players for Effective Opponent Exploitation. In *AAAI Workshop on Computer Poker and Imperfect Information Games*, San Francisco, CA.