Adaptive Learning for Poker

Luigi Barone and Lyndon While
Department of Computer Science,
The University of Western Australia,
Western Australia, 6907
{luigi, lyndon}@cs.uwa.edu.au

Abstract

Evolutionary algorithms are more than function optimisers — they adapt and learn in dynamic environments. In this paper, we use this implicit learning characteristic of evolutionary algorithms to create a computer poker player capable of adapting to different opponent strategies. We identify several important poker principles and use these as the basis for a hypercube of evolving populations of poker playing candidates. We simulate the most commonly employed strategies of human players and report experiments that demonstrate the emergent adaptive behaviour of our evolving poker player. In particular, we show that our evolving poker player develops different techniques to counteract the different playing styles employed by its opponents in order to maximise personal winnings. We compare the strategies evolved by our poker player with a competent static player to highlight the importance of adaptation and demonstrate the improved performance of our approach. Finally we discuss a real-world implementation of our model that recently competed in the annual rec.gambling.poker Hold'em poker elimination tournament. Our poker player had some success, winning a few hands, finishing in the top 22% of players.

1 Introduction

Evolutionary computation unifies under one umbrella term the different studies in computer science that use neo-Darwinian evolution as an adaptive optimisation tool to solve problems in computers. These algorithms employ a population of candidate solutions that evolve under the influence of a fitness metric until a solution emerges that satisfies the criteria of the problem.

Beyond the genetic level of modelling evolution [9] lies the field of evolutionary algorithms, where evolution is modelled as behavioural linkages between parents and offspring. Encompassing evolutionary programming [8] and evolutionary strategies [12], evolutionary algorithms model problems in terms of phenotypic behavioural traits instead of populations of genetic material. The underlying complex genetic transformations are ignored, with changes to phenotypic traits assumed to follow a Gaussian distribution with zero mean difference and a given standard deviation. While evolutionary programming deals with the creation of finite state machines, the field of evolutionary strategies uses a vector based representation of behavioural traits. There are two common forms [15]: a $(\mu + \lambda)$ strategy uses μ parents to generate λ offspring, with the next generation chosen from the combined population of parents and offspring. A (μ, λ) strategy, μ parents again generate λ offspring, but the next generation is chosen only from the offspring population.

Poker is a card game in which the players are dealt cards and then take turns to bet money into a communal pot. The aim of the game is to win as much money as possible from opponent players. At each turn, a player may either:

1. *fold*: conceding all interest in the pot,
2. *call*: matching the (possibly zero) previous bet,
3. *raise*: making a bet that exceeds the previous bet.

Play proceeds until each active player has placed an equal amount of money in the pot. At the conclusion of the deal, the player forming the best *hand* is deemed the winner and is awarded the pot. Variants of poker use different numbers of betting rounds, interspersed with the replacement, receipt, or revelation of cards.

Poker is interesting because it is a game of imperfect information. Unlike games of perfect information (e.g. chess), in which players have complete knowledge about the state of the game, players of games of imperfect information must infer their relative strength in the game using only the public information available to them. With incomplete or imperfect information, players can attempt to deceive their opponents by falsely portraying strength (or weakness) about their position in the game — the correct handling of this imperfect information is essential for optimal play. Fundamental to good poker is the ability to deduce opponents' playing styles in order to exploit their weaknesses. This requires adaptive learning in order to maximise winnings against different playing styles.

In this work, we study the most widely played poker variant, Texas Hold'em [10]. As with most poker variants, a maximum of three raises is allowed during each betting round.

In previous papers [2, 3], we presented two frameworks for designing adaptive learning poker players using evolutionary algorithms. Both our previous works restricted the adaptation to one opponent strategy at a time and used a simplified version of the game, limiting the game to one betting round. In this paper, we lift these restrictions. Section 2 reviews previous attempts at computer based poker players. Section 3 describes our updated model for adaptive learning in the full game of poker. Section 4 describes experimental results that demonstrate the adaptive behaviour of our approach. In particular, we demonstrate that the model evolves strategies capable of winning against vastly different playing styles — the specialisation allowing the player to win more than the generalised strategy it was based on. Section 5 concludes the work.

2 Previous Poker Models

The first theoretical work with the game of poker was conducted by von Neumann and Morgenstern in the 1940s [18]. Applying game theory to a very simplified version of poker, they demonstrated the need for deception, called bluffing, for competent play. Attempts were made to adapt their approach to more realistic versions of the game [1, 13], but with only limited success. Typically these approaches fail due to their inadequate handling of several poker principles (e.g. betting position) that are fundamental to the game.

Many books have been written on how to play poker, included some by today's experts [10, 16]. Suggesting a simple mechanical approach, and while typically statistically inclined, these books are usually not very systematic or rigorous in their approach. As these approaches are non-adaptive, considering the average result of each hand, models based on this methodology can never be optimal.

The earliest research into designing computer-based poker players was by Findler and was based on a series of simple heuristics and statistical models [5, 7]. Findler devised a number of algorithms that bet in a well-defined deterministic manner, revealing information that could be exploited by a competent human player. He concluded that static models were unsuccessful and that adaptive algorithms were required for successful play [6]. Findler conducted a number of experiments to estimate commonly-arising probabilities, using them to define game-specific heuristics for his poker model. He progressed onto examining some simple learning strategies and assigned a subjective evaluation of their relative performance in comparison to some existing mathematical models, but did not draw any conclusive results.

Work at The University of California [11] has concentrated on using a variant of the game-tree approach used in games of perfect information to solve games of imperfect information. They have applied their system to a very simplified game of poker using a reduced deck and two players, with some interesting results. However, the size of the game tree seems to be a limiting factor, with the authors conceding it is unlikely that they will ever solve the complete game.

The Computer Poker Research group at the University of Alberta is currently working towards creating a world-class poker player [14]. After initially creating a poker program based on static expert knowledge, they concluded adaptive play and opponent modelling are essential for strong play [4]. Using an adaptable array of weights to modify opponent starting hand probabilities, their program attempts to adapt to different playing styles by observing the betting tendencies of opponents.

3 An Adaptive Poker Player

We propose the use of evolutionary algorithms as an adaptive learning mechanism in the design of an evolvable poker player. As a result of competition for space in a population, population members need to learn which poker-playing strategies are successful in order to propagate into future generations. We embed this implicit evolutionary learning process in our poker player to make it adaptive — the adaptation is essential to learn opponent playing styles and to exploit opponent weaknesses in order to maximise personal winnings. We structure our poker player as a hypercube of populations of different betting suggestions with a $(1+1)$ evolutionary strategy as the learning mechanism.

3.1 Poker Principles

We incorporate five important principles of the game of poker into our model. The first principle is *hand strength*. With a strong hand, a player should often raise, trying to maximise the stakes when there is high probability of winning. Correspondingly, when holding a weak hand, a player should normally fold.

The second principle is *opponent strategy*. As previously suggested, different opponents can employ vastly different strategies. A generalised strategy, while potentially sound, will not be as successful against an opponent as a strategy optimised from play against that opponent. To maximise winnings, a player must be able to counteract all different opponent playing styles and recognise when to use each counter-measure. This competence is responsible for noting which opponent is using which strategy.

The third principle is *position*. This competence considers where in the betting round the player is forced to act. Being first or second to act (called early position) places a player at a disadvantage as the player has no idea how subsequent players will act. However, a player in late position has seen how others have acted and already has information to start inferring possible hand strengths of opposing players. Late position players are often able to play hands more aggressively.

The fourth principle is *risk management*. Risk management is responsible for examining how much money is required to remain in contention to win. When considering risk management, a player determines if a call or raise is justified in terms of expectation, comparing the size of the potential winnings with the size of the stake required to remain in contention to win (i.e. the odds). When there are many players contesting a pot, a player can often afford to risk a small amount of money on a long shot because the size of the pot justifies the small risk — the expectation is positive.

The last principle is *game stage*. This competence is responsible for noting that different strategies are required at different stages in the game. As a hand progresses, more information is revealed and the final result becomes more certain. For example, a small pair before the flop is quite favourable and worthy of a raise, but with a non-favourable flop and against many opponents, this hand should probably be folded.

3.2 Determining a Game Action

At each betting round, each player in turn must decide whether to fold, call, or raise. For good players, the decision is based on their ability to estimate (either by calculation or from experience) the probability of winning from any given game state. We equip our evolving poker player with this ability by allowing it to enumerate all possible opponent starting hands to determine the correct number of hands lost to, tied with, and won against. Note however, that the calculated probability refers to the likelihood of winning against random opponent hands and does not incorporate information inferred about opponent hand strength from opponent betting patterns. For example, to simply ensure positive expectation may give up potential winnings, especially against players who can be easily bluffed.

We observe the following about expert players:

- Their likelihood of raising increases as the probability of winning increases.
- Their likelihood of folding decreases as the probability of winning increases.
- Their likelihood of calling is maximal when they are unsure about their relative strength in the game, decreasing as their confidence increases. This allows for the opportunity of winning, while minimising potential losses if beaten.

Using these observations, we model action determination with the following functions:

$$\begin{aligned}
eval(c) &= \frac{c}{1.0 - c} \\
fold(x) &= \exp(-eval(b) \times (x - a)) \\
call(x) &= eval(c) \times \exp(-eval(b)^2 \times (x - a)^2) \\
raise(x) &= \exp(eval(b) \times (x + a - 1.0))
\end{aligned}$$

where x is the independent variable corresponding to the probability of winning from a given game state.

The constants a, b, and c define the shape of each function. For the $fold(x)$ and $raise(x)$ functions, a defines the intersection point with the function $f(x) = 1$, while b controls the gradient of the function. For the $call(x)$ function, a defines the mid-point of the Gaussian, b controls the width of the function, while c controls the maximum allowable value. The $eval(c)$ function maps a number in the range $[0..1]$ to the range $[0..\infty)$ with $eval(0.5) = 1$. Hence, all constants can be constrained to the range $[0..1]$.

These three functions determine the game action of our poker player. The probability of winning from the current game state is calculated and is used to determine a response for each betting action by using the functions defined above. The responses from each function are transformed into probabilities and are used to probabilistically determine the action of the poker player. We group these seven constants together and call the grouping a *candidate*.

The computational requirements for an accurate calculation of the probability of winning from a given game state exceeds current computing resources and hence some approximation is required. We employ a lookup table for pre-flop strength calculation and only ever calculate the probability of winning against a single opponent. Following Billings et al. [4], we approximate the probability of winning against n players by raising the single-opponent probability by n.

3.3 The Evolutionary Structure

No one poker principle is sufficient by itself. A good poker player will considered all the poker principles outlined above in order to make an informed decision about the game. We use a hierarchical structure for our evolving poker player — the poker player consisting of a hypercube of populations of candidates. We segment our hypercube into four dimensions: one representing position, one representing risk management, one representing game stage, and one representing opponent strategy. The hand strength competence is considered in the strength calculations mentioned above.

The position dimension is sub-divided into three divisions: early, middle, and late. The risk management dimension is classified into four divisions: one for each of the possible number of bets to call, starting from zero, up to and including three bets. The game stage dimension is segmented into the four betting rounds in the game: pre-flop, post-flop, post-turn, and post-river (or showdown). The opponent strategy dimension is sub-divided into segments for each different opponent, but may be reduced by combining segments of like-minded opponents. We note that we can easily expand our model to include more poker principles by adding extra dimensions to the hypercube.

Each element of the hypercube consists of a population of candidates, with each candidate consisting of seven values in the range [0..1]: two each for the constants in the fold and raise functions, and three for the constants in the call function. These values evolve over time in our evolutionary model and are used to determine the action of the poker player.

Excluding the opponent strategy dimension, the discretised hypercube uniquely describes one population of candidates for each possible game state. When our poker player is asked to act, the game state is discretised and the corresponding population is used to determine the action. From the selected population, one candidate is chosen as representative for the player. The choice of candidate cycles through all population members as the population is consulted from hand to hand. At the end of a hand, feedback about the success in the hand (how much was won or lost) is reported directly back to the chosen population. The representative candidate, along with all other candidates from the same population suggesting the same action, receives this feedback — the feedback used to build a fitness score. As the player is often required to make multiple actions per hand, feedback about poker playing success is reported to all used candidates.

This unique discretisation of game state fails when considering play against multiple opponents. Depending on its position, our poker player will often be competing against multiple active (un-acted opponents are assumed active) players. For each active opponent, we determine a betting action as before assuming this to be the only opponent in the game. Requiring to beat all opponents, we choose the tightest (least costly) action reported as the action for our poker player. Feedback is only reported to the population used.

Evolution of the population occurs when each candidate has been used a fixed number of times. Evolution occurs via a $(1+1)$ evolutionary strategy, with mutation occurring on all seven real values of the parent.

4 Experimental Results

Our experiments use an evolutionary strategy approach to extend the results of previous research into computer poker. We follow the same experimental structure as our previous works on adaptive learning for the game of poker [2, 3].

4.1 Opponents

Research by Sun and Wu [17] indicates that genetic algorithms, when used to play the game of Othello, over-adapt to the weaknesses of the current competition, resulting in poor performance against opponents with previously unencountered strategies. They suggest using a number of different coaches to train evolving players to ensure the evolution of a strong general player. However, in the game of poker, different opponent playing styles require different strategies in order to maximise winnings against each opponent. We take the view that our poker player should start from a prebuilt general strategy and adapt to the weaknesses of that opponent's playing style — the specialisation essential for discovering opponent weaknesses.

Real-world poker players are typically classified as one of four types (listed in ascending order of ability):

- *Loose-Passive*: over-values its hands, but rarely raises, being fearful of large pots.
- *Loose-Aggressive*: over-values its hands, raising often to increase potential winnings.
- *Tight-Passive*: plays few hands, usually with a high probability of winning, but rarely raises.
- *Tight-Aggressive*: plays few hands, usually with a high probability of winning, raising often.

For our experiments, we use a table configuration, called the *mixed table*, consisting of one evolving poker player, three tight-passive players, and six loose-aggressive players. For reference, we define two tables containing only one type of opponent strategy. The *loose table* has one evolvable poker player and nine loose-aggressive players. The *tight table* has one evolvable poker player and nine tight-passive players. When playing against loose-aggressive players, our evolving poker player needs to counteract the high variance play of the loose-aggressive players. Against tight-passive players, the evolving poker player needs to exploit their timidness in order to maximise personal winnings.

4.2 Experiment 1

Experiment 1 demonstrates the learning ability of evolutionary algorithms in the game of poker. Starting with a randomly constructed poker player, we show that our model rapidly evolves an competitive player against a range of different opponents. Each epoch, each player starts with a "stack" of 1000 units. An epoch comprises 100 hands of play. Each population in the hypercube consists of 20 candidates. Each population member is used 25 times before the population is subjected to reproduction.

Figure 1 plots the stack size of the evolving poker player at the end of each epoch at the mixed table. The graph shows an upward trend in the return of the evolving poker player, corresponding to a fall in the return of both the loose-aggressive and tight-passive players. Initially, the evolving randomly-defined poker player plays rather badly and loses a large proportion of its stack. However, at the 71st epoch, the return of the evolving poker player is first positive and a rapid increase in profit is realised. After approximately 200 epochs, the rate of increase levels off, but the number of losing sessions decreases over time.

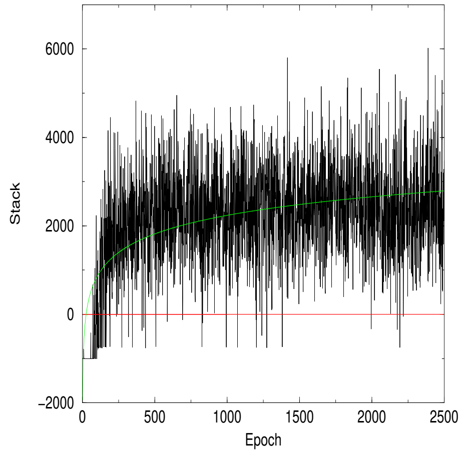

Figure 1: Stack Size Versus Epoch for the Evolving Poker Player at the Mixed Table

4.3 Experiment 2

Experiment 2 demonstrates adaptive behaviour from a fixed playing style. As expressed previously, we take the view that our poker player should start from a generalised playing strategy and adapt to the weaknesses of the competition at hand. This adaptation is essential in order to maximise winnings — a general strategy will not be able to exploit the weaknesses in an opponent's playing style as well as a strategy optimised from play against that opponent. We used the ideas of Findler's static poker players [6] to design a simple, but competent, tight-aggressive player as our starting point. We name this player A.

We evolved A for 1000 epochs at the mixed table, naming the strategy evolved for play against the tight-passive players at the end of the last epoch A^{tight}. We name the strategy evolved for play against the loose-aggressive players at the end of the last epoch A^{loose}.

Figure 2 shows the post-river evolved controller for A^{tight}. Each plot represents the average candidate of each hypercube element in the evolving poker player. The x-axis of each plot is the probability of winning, and is the independent variable in the betting functions of the candidate. Each plot shows the probability of following each of the three betting actions: fold (thin solid line), call (thick solid line), and raise (thin broken line).

As expected, the evolved controller of A^{tight} suggests playing a tighter strategy (playing less hands) as the number of bets to call increases (left to right) — the player has realised that opponent raises indicate that they probably hold a good hand. The evolved controller also suggests playing a tighter strategy as the

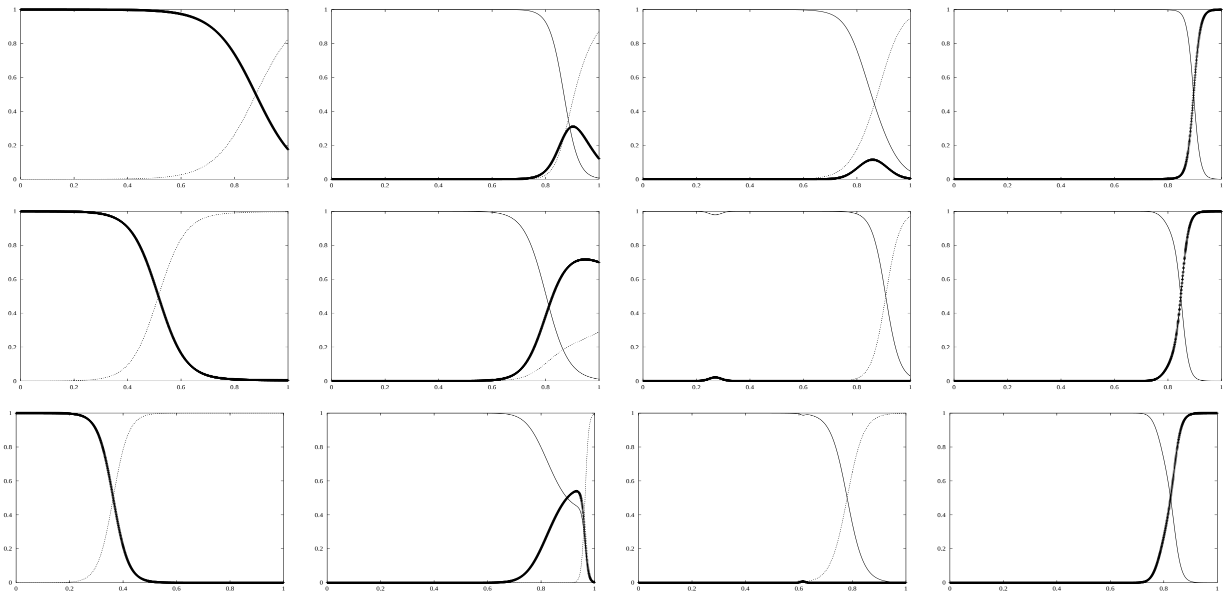

Figure 2: The Evolved Controller of A^{tight}

Position gets later top-bottom (early to late) and bets-to-call increases left-right (0–3).

position of the evolving poker player becomes earlier (bottom to top). This is most evident in the zero bets to call elements of the hypercube. In early position, with zero bets to call, the evolved controller of A^{tight} predominately suggests calling. However, when in late position and zero bets to call, the controller suggests raising a lot more often. These two situations are remarkably different and are a consequence of the tight-passive player's fear of raises.

In late position, the evolving player can be confident that all previously acted players do not value their chances of winning highly (they would have raised otherwise) and because of their tightness can be bluffed out of the pot by raising. However, in early position, bluffing with a weak hand is dangerous as even a very tight player knows to raise when holding a good hand. Only in the cases of holding a very good hand will the controller suggest not folding when there are more than zero bets to call.

Figure 3 shows the post-river evolved controller of A^{loose}. The evolved controller for A^{loose} suggests a significantly different strategy than A^{tight}, with over 15% of hands being played differently. This difference reflects the tendency of the loose-aggressive players both to over-value their own hands (so you can't win by bluffing with a mediocre hand at the loose table) and to stay in even against an opponent's raise (so you can raise to maximise winnings without losing action). When there is one or more bets to call, the evolved controller of A^{loose} suggests a significantly looser strategy than that of A^{tight}, choosing to call or raise more often than that of A^{tight}. The evolving poker player has learnt to respect the raises of the tight-passive players much more than those of the loose-aggressive players. With zero bets to call, the evolved controller of A^{loose} suggests not bluffing as often as the controller of A^{tight}, since loose-aggressive players are less likely to be forced out of a hand. The evolving poker player has learnt that tight-passive players respect the raises of opponent players much more than loose-aggressive players.

4.4 Experiment 3

Experiment 3 demonstrates that the evolved strategies from Experiment 2 out-perform both A and each other in their own "environments". We play A^{tight} and A^{loose} at tables containing only tight-passive and loose-aggressive players and demonstrate that the evolved strategies outperform A. Defining $W_t(X)$ to be the average return of a non-evolving player X at a table t, we expect the four inequalities of Table 1 to hold. The first two inequalities state that both evolved strategies perform better than A at tables containing only the opponent playing style used in their training. The other two inequalities state that the evolved strategies suited for play at table t perform better than players evolved at another table t' when playing at t.

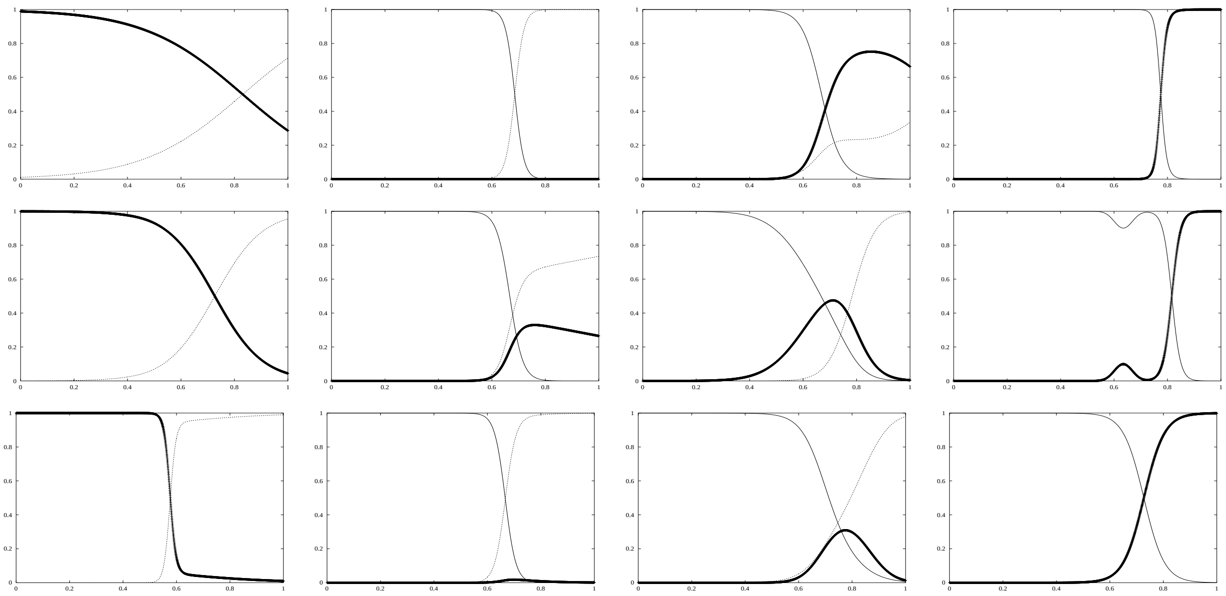

Figure 3: The Evolved Controller of A^{loose}

Position gets later top-bottom (early to late) and bets-to-call increases left-right (0–3).

$$\begin{array}{lcl} W_{tight}(A^{tight}) & > & W_{tight}(A) \\ W_{loose}(A^{loose}) & > & W_{loose}(A) \\ W_{tight}(A^{tight}) & > & W_{tight}(A^{loose}) \\ W_{loose}(A^{loose}) & > & W_{loose}(A^{tight}) \end{array}$$

Table 1: Inequalities for Our Poker Model

We fixed (no evolution) the strategies of A^{tight} and A^{loose} and played these players along with A against the loose-aggressive and tight-passive players of the loose and tight tables respectively. The results are summarised in Table 2. Each figure is the average of 1000 epochs' play at the appropriate table.

$W_{tight}(A^{tight})$	1978
$W_{tight}(A)$	-528
$W_{tight}(A^{loose})$	-361
$W_{loose}(A^{loose})$	6119
$W_{loose}(A)$	5294
$W_{loose}(A^{tight})$	4222

Table 2: Results of Players A, A^{tight} and A^{loose} at the Loose and Tight Tables

The results of Table 2 support our claim. Analysis shows a statistical difference between the two means of each inequality at the 5% level of significance.

These results again highlight the fact that loose players are easier to exploit than tight players. They show that the evolving poker players have specialised to the playing styles of the competition at hand and are able to exploit the discovered weaknesses in their opponents' strategies in order to maximise their winnings against each strategy.

4.5 Real World Testing

To gain some real-world experience, we entered our poker player in the annual, play-by-email no-limit poker tournament collectively hosted by the Usenet group rec.gambling.poker. The tournament started with 987 competitors, including expert poker players and authors of best-selling poker books. The tournament presented a new challenge to our poker player — our player had never competed in a no-limit tournament structure before, having previously only played in structured "ring" (continuous) games.

We took the evolved strategy A^{tight} and used this as the base strategy for unknown competitors. We approximated unrestricted opponent bets to a number of raises, dependent on the absolute size of the bet and the size of the bet relative to their stack. When suggesting a raise, we instructed our poker player to double the size of the previous bet, but non-deterministically allowed for greater sized bets.

Our poker player played a very conservative game, playing few hands, but generally very solidly. Our poker player was recently eliminated from the tournament, finishing within the top 22% of players — even managing to outlast one of the authors! We consider this a very reasonable effort, especially considering it had never competed in a fast-paced tournament before.

5 Conclusions

Poker, a game of simple rules, often requires complex strategies for competent play. One of the most important qualities of a good poker player is the ability to deduce opponent playing styles in order to exploit individual weaknesses in each strategy. Dynamic algorithms that learn and adapt to opponents' playing styles are essential to maximise winnings.

In this paper, we discussed our model of adaptive learning using evolutionary algorithms to learn the game of poker. We used several important poker principles to create a hypercube of populations of candidates, using a discretisation of the game state to select a representative candidate. Our experimental results show that our evolving poker player can develop different techniques to counteract the different strategies employed by opponents, exploiting weaknesses to maximise personal winnings. We have demonstrated adaptive learning by showing that our poker player out-performs a competent static player when played against the same opposition.

References

[1] N. C. Ankeny. *Poker Strategy: Winning with Game Theory.* Basic Books Inc., New York, 1981.

[2] L. Barone and L. While. Evolving adaptive play for simplified poker. In *Proceedings of the 1998 International Conference on Evolutionary Computation (ICEC '98)*, pages 108–113. IEEE Publications, 1998.

[3] L. Barone and L. While. An adaptive learning model for simplified poker using evolutionary algorithms. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC '99).* IEEE Publications, 1999.

[4] D. Billings, D. Papp, J. Schaeffer, and D. Szafron. Opponent modelling in poker. In *Proceedings of the Fifteenth National Conference of the American Association for Artificial Intelligence*, 1998.

[5] N. V. Findler. Computer model of gambling and bluffing. *IRE Transactions on Electronic Computers*, pages 97–98, 1961.

[6] N. V. Findler. Studies in machine cognition using the game of poker. *Communications of the ACM*, 23(4):230–245, 1977.

[7] N. V. Findler, H. Klein, and Z. Levine. Experiments with inductive discovery processes leading to heuristics in a poker program. In *Cognitive Processes and Systems*, pages 257–266, 1973.

[8] L. J. Fogel. Autonomous automata. *Industrial Research*, 4:14–19, 1962.

[9] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Cambridge, MA, 1975.

[10] L. Jones. *Winning Low Limit Hold'em.* ConJelCo, Pittsburgh, PA, 1994.

[11] D. Koller and A. Pfeffer. Generating and solving imperfect information games. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1185–1193, 1995.

[12] I. Rechenberg. Cybernetic solution path of an experimental problem. Royal Aircraft Establishment, Library translation No. 1122, 1965.

[13] M. Sakaguchi and S. Sakai. Solutions of some three person stud and draw poker. *Math Japonica 37*, pages 1147–1160, 1992.

[14] J. Schaeffer, D. Billings, L. Pea, and D. Szafron. Learning to play strong poker. In *Proceedings of the Sixteenth International Conference on Machine Learning*, 1999.

[15] H.-P. Schwefel. *Numerical Optimization of Computer Models.* John Wiley and Sons, England, 1981.

[16] D. Sklansky. *The Theory of Poker.* Two Plus Two Publishing, Las Vegas, NV, 1987. Formerly titled "Winning Poker".

[17] C.-T. Sun, Y. H. Liao, J. Y. Lu, and F. M. Zheng. Genetic algorithm learning in game playing with multiple coaches. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 239–243, 1994.

[18] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behaviour.* Princeton University Press, 1944.