

# An Adaptive Learning Model for Simplified Poker Using Evolutionary Algorithms

**Luigi Barone**

Department of Computer Science,  
The University of Western Australia,  
Western Australia, 6907  
luigi@cs.uwa.edu.au

**Lyndon While**

Department of Computer Science,  
The University of Western Australia,  
Western Australia, 6907  
lyndon@cs.uwa.edu.au

**Abstract-** Evolution is the process of adapting to a potentially dynamic environment. By utilising the implicit learning characteristic of evolution in our algorithms, we can create computer programs that learn, and evolve, in uncertain environments. We propose to use evolutionary algorithms to learn to play games of imperfect information – in particular, the game of poker.

We describe a new adaptive learning model using evolutionary algorithms that is suitable for designing adaptive computer poker players. We identify several important principles of poker play and use these as the basis for a hypercube of evolving populations in our model. We report experiments using this model to learn a simplified version of poker; results indicate that our new approach demonstrates emergent adaptive behaviour in evolving computer poker players. In particular, we show that our evolving poker players develop different techniques to counteract the variety of strategies employed by their opponents in order to maximise winnings. We compare the strategies evolved by our evolved poker players with a competent static player to demonstrate the importance of adaptation to achieve this end. Comparison with our existing evolutionary poker model highlights the improved performance of this approach.

## 1 Introduction

Genetic algorithms, first introduced by Holland [9], employ the principles of evolution by natural selection to solve problems in computers. A population of candidate solutions evolves under the influence of some selection mechanism until a solution emerges that satisfies the criteria of the problem. Beyond the genetic level of modelling evolution lies the field of evolutionary algorithms, where evolution is modelled as behavioural linkages between parents and offspring. Encompassing evolutionary programming [8] and evolutionary strategies [13], 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.

Evolutionary algorithms can model both sexual and asexual reproduction [15]: a  $(\mu + \lambda)$  evolutionary strategy uses  $\mu$

parents to generate  $\lambda$  offspring, with the next generation chosen from the combined population of parents and offspring. Other strategies exist where the next generation is chosen only from the offspring population: these are termed  $(\mu, \lambda)$  evolutionary strategies.

Poker is a card game in which the players are dealt cards and then take turns to bet money into a communal pot. At each turn, a player may either:

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

The aim of the game is to win as much money as possible from opponent players. Different poker variants use various numbers of betting rounds, interspersed with the replacement, receipt, or revelation of cards.

Poker is a game of imperfect information. In this type of game, players do not have complete knowledge about the state of the game and must make value decisions about their relative strength using only the public information available to them – the handling of this incomplete information is essential for optimal performance. The most fundamental ability of a successful poker player is deducing opponents' playing styles in order to exploit their weaknesses. To do so requires adaptive learning and is essential in order to maximise winnings against different playing styles.

In this paper, we introduce a new framework for designing adaptive poker players. We embed the implicit learning characteristics of evolutionary algorithms in our model to evolve poker players that can exploit weaknesses in opponents' playing styles. In Section 2 we see that most previous approaches to designing computer poker players were based on the idea of static models and hence cannot be adaptive. Section 3 introduces our new model for adaptive poker players and Section 4 describes experiments that demonstrate the adaptive behaviour of this approach. In particular, we demonstrate that our model evolves different poker players that are capable of winning against vastly different playing styles – the specialisation allowing them to win more than the generalised strategy they were based on. Section 5 concludes the work.

## 2 Previous Models to Play the Game of Poker

The earliest research in applying game theory to poker was conducted by von Neumann and Morgenstern in the 1940s [19]. Using a very simplified game of poker, one of their most important conclusions was the need for bluffing for competent play. Some attempts were made to adapt this approach to more realistic games of poker [1, 14], but with only limited success. Common failings of these game-theoretical approaches include their inadequate handling of several fundamental poker principles (e.g. betting position) and the difficulty of applying them to different poker variants. A model with a dynamic learning ability should be applicable (with re-training) to any variant of the game.

Many books have been written on how to play poker, including some by poker experts of today's standard [10, 16]. Often these books target human psychology or preach a simple mechanical approach to the game, and while generally statistically inclined, they are usually not very systematic in their approach. As these approaches are non-adaptive, considering only the probability of a given hand winning, these types of models can never be optimal.

Virtually all previous learning-based research into designing computer-based poker players was by Findler, and is based on a series of simple heuristics and statistical models [4, 6, 7]. He considered a simplified version of the game – a form of draw poker with no ante, in which the computer always played last. In Findler's early work, he devised a number of algorithms that bet in a well-defined, deterministic manner. This approach reveals far too much information to an intelligent opponent – it would be defeated easily by competent human players.

Findler concluded that static mathematical models were unsuccessful and that dynamic, adaptive algorithms are required for successful play [5]. He performed a number of experiments to estimate certain commonly-arising probabilities and used these results to define a number of game-specific heuristics for his poker variant. 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 no conclusive results were obtained – he labelled most of his learning algorithms as “under construction”.

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.

Also of note is recent work using Bayesian networks [12]. A Bayesian network is a hierarchically-structured program that uses conditional probabilities combined with facts believed to be true to formulate the probability of an event occurring. Published results using this approach are very lim-

ited: their experiments are restricted to two player games, using only knowledge of revealed cards, making judgments about the success of this method difficult at this time.

## 3 Designing An Evolvable Poker Player

We propose using evolutionary algorithms to design a poker player capable of learning the playing style of its opponents and exploiting their weaknesses to maximise its winnings. As a side-effect of the competition for limited resources (space in the population), the members of an evolving population need to learn which strategies are successful and which are unsuccessful in order to survive into future generations. We utilise this implicit learning process of evolutionary algorithms to make our poker player adaptive. We structure our poker player as a hypercube of populations of different betting suggestions and employ a  $(1 + 1)$  evolutionary strategy system as our learning mechanism.

### 3.1 Poker Principles

We incorporate three important principles of the game of poker into our model. The first principle is *hand strength analysis*. Hand strength analysis is concerned with determining the strength of the current hand and suggesting an action based on this information. With a strong hand a player should raise, trying to maximise the stakes when there is a high probability of winning. Correspondingly, with a weak hand, a player should fold, minimising the amount of money lost. With a medium-strength hand with a moderate chance of winning, a player could call, minimising risk while still being eligible to win the entire pot.

The second principle is *position*. Being first or second to act (called early position) places a player at a disadvantage – a player in early position has no idea how players in later betting positions will act. The early-position player is unsure how the other players value their hands, and may find that after a call or a raise, an opponent may re-raise, causing the player to re-assess their probability of winning. However, a player who is last or second last to act (called late position) has a large amount of information about how the other players value their hands. This player has seen how the other players have acted, and already has enough information to start inferring possible hand strengths of opposing players. A player in late position is often able to play hands more aggressively than a player in early position.

The third principle is *risk management*. Risk management is responsible for examining how much money is required to remain in contention to win the pot, making a betting suggestion depending on this amount. When considering risk management, a player determines if a call or raise is justified in terms of the expected value of the hand, 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.

### 3.2 Determining a Game Action

As a hand in a poker game progresses, each player in turn is asked to make a decision whether to fold, call, or raise – a decision being made based on their inferences of their relative strength in the game. For good human players, their decision is based on their ability to estimate (either by calculation or experience) their 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. The use of this information is essential in order to maximise winnings. For example, to simply ensure the expectation is positive often gives up potential winnings, especially against players who can be bluffed out of a hand.

As the probability of winning increases, we assume that the likelihood of a player raising will monotonically increase. Correspondingly, we assume the likelihood of a player folding will decrease monotonically as the probability of winning increases. We also assume that the probability of calling follows the shape of an augmented Gaussian, modelling the region between folding and raising. This allows us to represent the likelihood of following the three betting actions with the functions:

$$\begin{aligned} eval(c) &= \frac{c}{1-c} \\ fold(x) &= \exp(-eval(b) \times (x - a)) \\ call(x) &= eval(c) \times \exp(-eval(b)^2 \times (x - a)^2) \\ raise(x) &= \exp(eval(b) \times (x + a - 1.0)) \end{aligned}$$

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 point at which the function intersects  $f(x) = 1$ , 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  $c$  governs the maximum allowable value. The  $eval(c)$  function maps a number in the range  $[0..1]$  to the range  $[0..\infty)$  with  $eval(0.5) = 1$ . Thus all constants are constrained to the range  $[0..1]$ .

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

### 3.3 Evolutionary Structure

We propose using an hierarchical structure for our evolving poker player: the poker player consisting of a hypercube of populations of candidates. We segment our hypercube into two dimensions: one representing position and one representing risk management. The position dimension is sub-divided into three divisions: early, middle, and late position, while 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. This corresponds to twelve elements in the hypercube. Each element of the hypercube consists of a population of  $N$  candidates representing the functions defined above. We note that we can expand our model to include more poker principles by adding extra dimensions to the hypercube.

Each candidate of a population in a hypercube element consists of seven real 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. At any point in the game in which it's our poker player's turn to act, the poker player first selects the appropriate population in the hypercube by determining the poker player's current position and the number of bets to call. The selected population is consulted, with one candidate chosen to determine the action of the player. Choice of candidate, in turn, cycles through all population members of the selected hypercube element.

Feedback about the poker player's success in the current hand (how much was won or lost) is reported directly back to the chosen candidate. When multiple candidates are used (due to the poker player being asked to make multiple actions), feedback about poker playing success is reported to all used candidates. Evolution of the population occurs when each candidate has been used a fixed number of times. Evolution occurs via a  $(1 + 1)$  evolutionary strategy, in which the best  $N/2$  candidates are kept as parents for the next generation. The remaining  $N/2$  population members are discarded and replaced with a mutated copy of each of the parents; 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 most previous research into computer poker. We follow the same experimental structure as Barone and While [2]. We study a simple poker variant in which each player makes their best five card hand from two private cards and five community (shared by all) cards. We restrict the game to only one betting round, thus removing the need for partial hand evaluation. As with most poker variants, a maximum of three raises is allowed during the betting round.

## 4.1 Opponents

Research by Sun and Wu [17, 18] 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, but general, player. However, in the game of poker, opponent styles are often significantly different, requiring vastly different strategies in order to maximise winnings against each style – a general strategy will not win as much against an opponent as a specialised strategy evolved from play against that opponent. We take the view that our poker player should start from a pre-built general strategy and adapt to the weaknesses of that opponent’s playing style – the specialisation is essential to discover weaknesses in the playing style of the opponent.

It is commonly thought that there are four types of poker players, as shown in Table 1.

	Loose	Tight
Passive	Players who over-value their hands, but rarely raise, being fearful of large pots.	Players who play few hands, usually with a high probability of winning, but rarely raise, being fearful of large pots.
Aggressive	Players who over-value their hands, and who raise often to increase potential winnings.	Players who play few hands, usually with a high probability of winning, and who raise often to increase potential winnings.

Table 1: A Classification of Poker Players

See Barone and While [3] for a complete description of the simulated players used in our experiments.

Using these poker playing types, we can simulate a number of interesting poker table configurations. We concentrate on two table configurations for our experiments. The *loose table* has one evolving poker player and nine loose-aggressive players. When playing in this configuration, the evolving poker player will need to counteract the high variance play of the loose-aggressive players. The *tight table* has one evolving poker player and nine tight-passive players. When playing in this configuration, the evolving poker player will need to overcome the timidity of the tight-passive players.

Each player type has a particular weakness in their strategy. The loose-aggressive players have a tendency to play too many hands, losing many contested pots. Alternatively, the tight-passive players tend not to raise the bet often enough, failing to maximise their winnings when holding the best

hand. Our evolving poker player will need to exploit these weaknesses in order to be successful at each table.

## 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. The graphs below plot the stack size of the evolving poker player at the end of each epoch. An epoch comprises 500 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 shows the fitness profile for the loose table. The graph shows an upward trend in the fitness of the evolving poker player, corresponding to a fall in the fitness of the loose-aggressive players. Initially, the evolving randomly-defined poker player plays rather badly and loses a large proportion of its stack. However, at the 33rd epoch, the return of the evolving poker player is positive and a rapid increase in profit is realised. After approximately 250 epochs, the rate of increase levels off, but the number of losing sessions decreases over time.

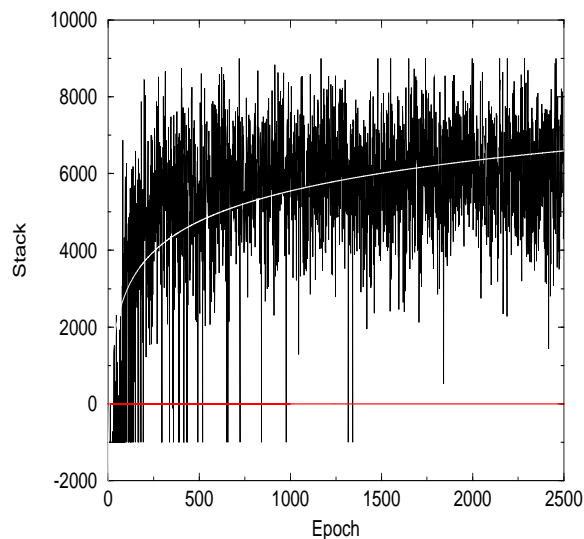


Figure 1: The Fitness Profile for the Loose Table

Figure 2 shows the fitness profile for the tight table. The result is somewhat similar to the loose table, with the performance of the evolving poker player starting badly and improving over time. However, it is evident that the winnings at this table are less than the winnings of the evolving poker player at the loose table. This is what we would intuitively expect at a table containing tighter players.

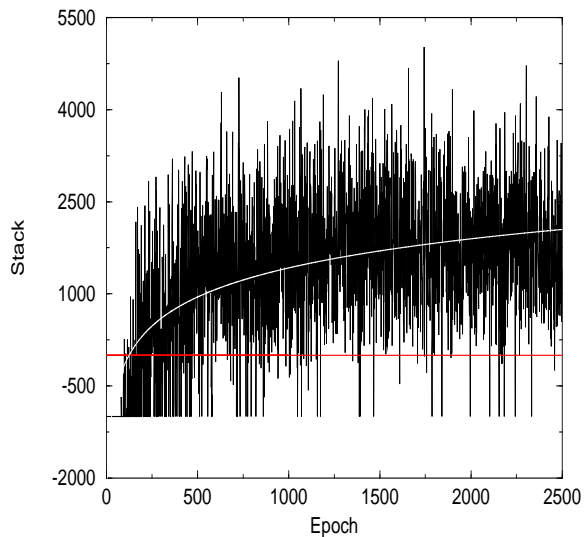


Figure 2: The Fitness Profile for the Tight Table

### 4.3 Experiment 2

Experiment 2 demonstrates adaptive behaviour from a fixed playing style. Adaptation is essential to maximise winnings – in order to exploit the weaknesses in an opponent’s playing style, a player needs to start with a reasonable playing strategy and then modify the strategy to the competition at hand. We used the ideas of Findler’s static poker players [5] to design a simple, but competent, tight-aggressive player as our starting point. We name this player  $A$ . See Barone and While [3] for a complete description of this player.

We evolved  $A$  for 2500 epochs at the tight table, naming the player at the end of the last epoch  $A^{tight}$ . Figure 3 shows the 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).

Note that both the leftmost and rightmost columns of Figure 3 contain only two probability distributions. When there are zero bets to call (the leftmost column), the player cannot fold, instead choosing the “free” option of calling the zero unit bet. The rightmost column corresponds to the situations when the evolving poker player is being asked to make an action when there are already three bets to call. With a maximum of three raises allowed in our poker variant, a raise suggestion in this situation is interpreted as a call action.

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 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 even when holding the best possible hand. However, in late position and zero bets to call, the controller suggests raising even when only having a 7% chance of winning against random opponent hands. 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 tight play can be bluffed out of the pot by raising. However, in early position, bluffing with a weak hand is fraught with danger as even a very tight player knows to raise when holding a good hand. When holding a reasonable hand in this position, the evolving poker player has learnt that the timid play of the tight-passive players means raising does not maximise winnings – tight-passive players will fold to a raise unless holding a good hand. Instead, the player has evolved a strategy of calling the zero unit bet, hoping to trap a later position player into raising on a weaker hand. Only in the cases of holding a very good hand will the evolved controller suggest not folding when there are more than zero bets to call.

We also evolved  $A$  for 2500 epochs at the loose table, naming the player at the end of the last epoch  $A^{loose}$ . Figure 4 shows the evolved controller of  $A^{loose}$ .

The evolved controller for  $A^{loose}$  suggests a significantly different strategy than  $A^{tight}$ , with almost 21% 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.

The different strategies evolved by  $A^{tight}$  and  $A^{loose}$  again demonstrate the adaptive process of evolutionary algorithms in action. We expect this learning process will allow us to build a poker player capable of strong play at the complete game in the near future.

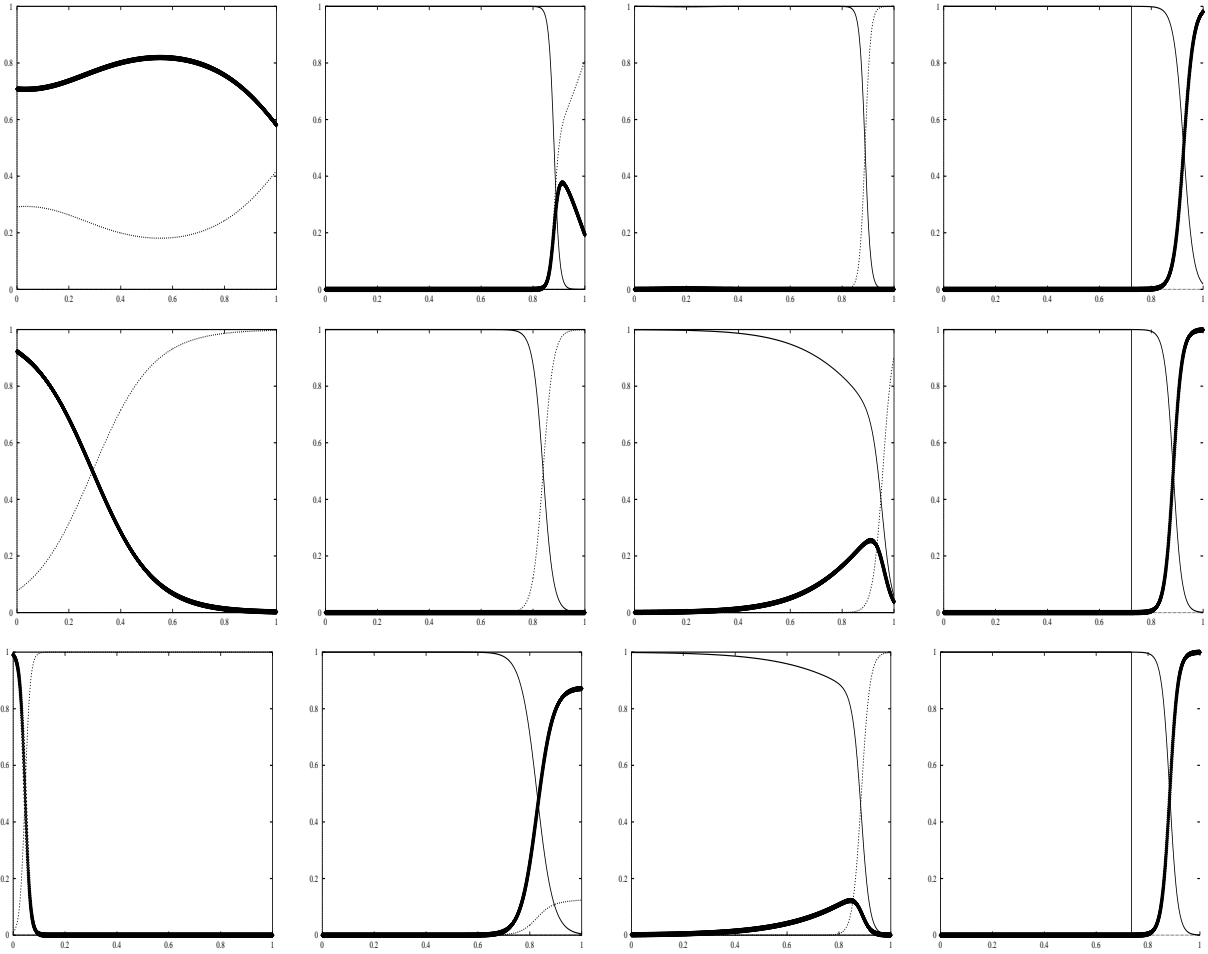


Figure 3: The Evolved Controller of  $A^{tight}$

Position gets later top-bottom (early to late) as the number of bets to call increases left-right (zero to three).

Each graph plots the probability of folding (thin solid line), calling (thick solid line), and raising (thin broken line) for the average candidate of each element in the hypercube. The x-axis of each plot represents the probability of winning from a given game state and is the independent variable in the betting functions of the candidate.

#### 4.4 Experiment 3

Experiment 3 demonstrates that the adapted players from Experiment 2 out-perform both  $A$  and each other in their own “environments”. Defining  $W_t(X)$  to be the average return of a non-evolving player  $X$  at table  $t$ , we expect the four inequalities of Table 2 to hold.

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

Table 2: Inequalities We Expect to Hold for Our Poker Model

The first pair of inequalities state that both evolved players perform better than  $A$  at their respective tables. The second pair of inequalities state that players evolved at a given table  $t$  perform better than players evolved at another table  $t'$  when playing at  $t$ .

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 3. Each figure is the average of 2500 epochs' play at the appropriate table.

The results of Table 3 support our claim. Statistical 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 dis-

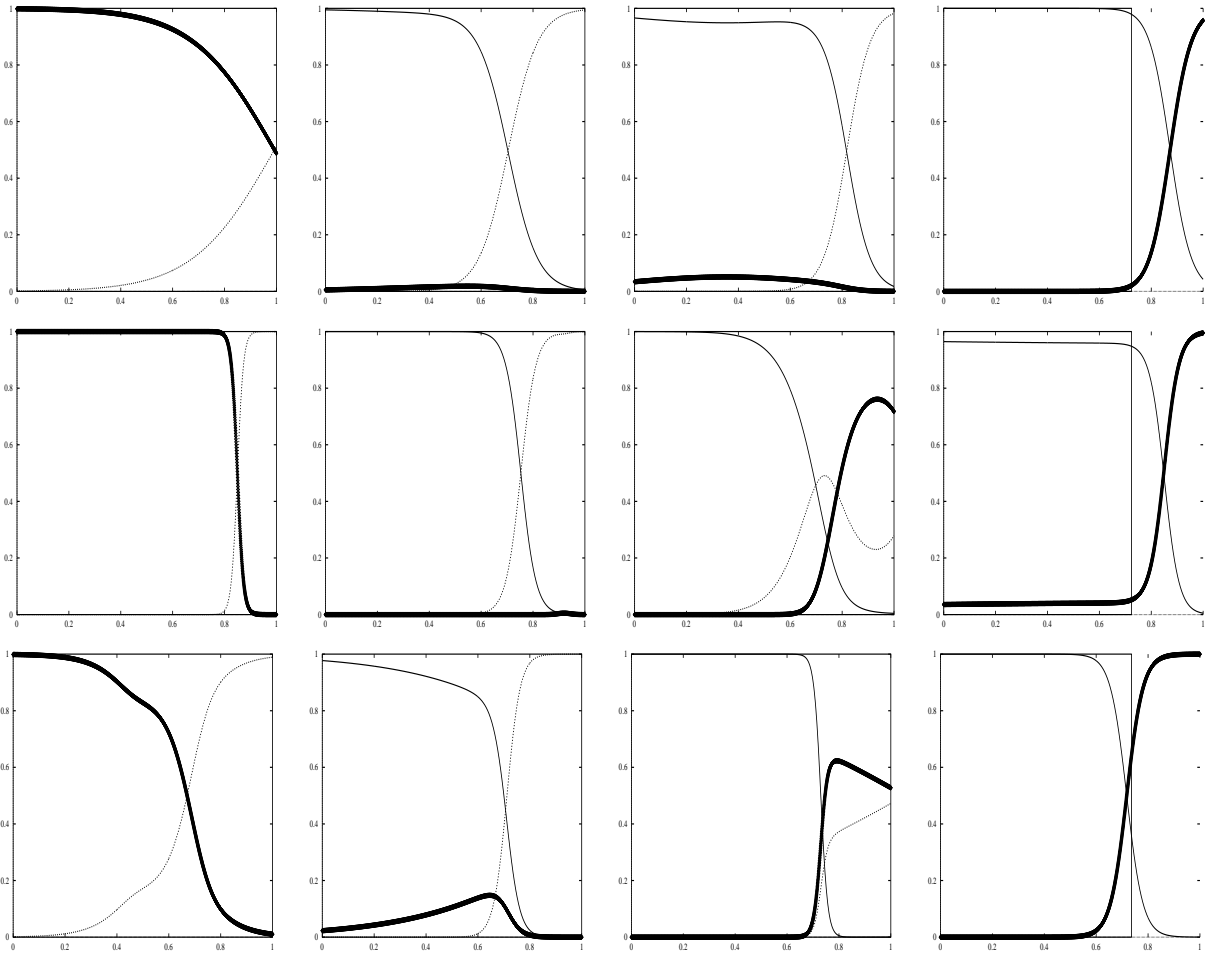


Figure 4: The Evolved Controller of  $A^{loose}$

Position gets later top-bottom (early to late) as the number of bets to call increases left-right (zero to three).

Each graph plots the probability of folding (thin solid line), calling (thick solid line), and raising (thin broken line) for the average candidate of each element in the hypercube. The x-axis of each plot represents the probability of winning from a given game state and is the independent variable in the betting functions of the candidate.

$W_{tight}(A^{tight})$	2113
$W_{tight}(A)$	-730
$W_{tight}(A^{loose})$	-195
$W_{loose}(A^{loose})$	6255
$W_{loose}(A)$	5400
$W_{loose}(A^{tight})$	4397

Table 3: Results of Players  $A$ ,  $A^{tight}$  and  $A^{loose}$  at the Loose and Tight Tables

covered weaknesses in their opponents' strategies in order to maximise their winnings against each strategy.

#### 4.5 Experiment 4

Our last experiment compares the results of this model with that of our previous poker model [2]. The results are sum-

marised in Table 4. For our new model, each figure is the average of 2500 epochs' play at the appropriate table. For our old model, each reported number is the average of 2000 generations of play with 50 poker agents used in the simulation.

Winnings	New Model	Old Model
$W_{tight}(A^{tight})$	2113	1194
$W_{loose}(A^{loose})$	6255	4147

Table 4: Comparison of our Two Poker Models

Statistical analysis of the results of Experiment 4 shows a statistical difference between the two models at the 5% level of significance. These results show that our new model out-performs our previous model in terms of the money won against each different playing style.

Along with better performance, our new model has other

advantages over our existing model. The previous model required a population of poker playing agents, each occupying an actual position in a game. Play under the new model requires only one poker game – evolution occurring inside the elements of the hypercube. Hence, the new model runs faster and can be more easily applied to real games against human opponents. However, when including new poker principles in the new model, care must be taken to equalise evolutionary pressure in all hypercube elements to ensure the development of good sub-controllers for all poker situations. Less care is required in the existing model.

## 5 Conclusions

Poker is more than a game of chance – the need for bluffing, and most importantly, the necessity of deducing other opponents' playing styles in order to exploit their weaknesses is essential to maximise winnings. While many statistical approaches exist to play the game of poker, due to their inability to adapt to individual playing styles, these strategies are not optimal. Dynamic algorithms that learn and adapt to opponents' playing styles are necessary to maximise winnings.

As a precursor to modelling adaptive behaviour in the full game of poker, we have introduced a new model of adaptive learning using evolutionary algorithms to learn a simplified version of the game. Our results indicate that this evolutionary approach demonstrates adaptive behaviour when applied to different poker configurations. In particular, our experiments show how evolving poker players can develop different techniques to counteract the strategies employed by different opponents. The evolved poker players have discovered weaknesses in their opponents' strategies and were able to exploit them to maximise winnings. We have also demonstrated that our evolving players out-perform a competent static player when played in the same configuration. Lastly, we showed that this new model out-performs our existing evolutionary model against the same opposition.

## Bibliography

- [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, 1998.
- [3] L. Barone and L. While. An adaptive learning model for simplified poker using evolutionary algorithms. Technical Report 99/1, Department of Computer Science, The University of Western Australia, 1999.
- [4] N. V. Findler. Computer model of gambling and bluffing. *IRE Transactions on Electronic Computers*, pages 97–98, 1961.
- [5] N. V. Findler. Studies in machine cognition using the game of poker. *Communications of the ACM*, 23(4):230–245, 1977.
- [6] N. V. Findler. Computer poker. *Scientific American*, 239(1):112–119, July 1978.
- [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] K. Korb, A. Nicholson, T. T. Lien, and N. Jitnah. Bayesian poker player. Poster presentation. Email [annn@cs.monash.edu.au](mailto:annn@cs.monash.edu.au) for details.
- [13] I. Rechenberg. Cybernetic solution path of an experimental problem. Royal Aircraft Establishment, Library translation No. 1122, 1965.
- [14] M. Sakaguchi and S. Sakai. Solutions of some three person stud and draw poker. *Math Japonica* 37, pages 1147–1160, 1992.
- [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] C.-T. Sun and M.-D. Wu. Self-adaptive genetic algorithm learning in game playing. In *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'95)*, pages 814–818, 1995.
- [19] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behaviour*. Princeton University Press, 1944.