Investigation of an Adaptive Cribbage Player

Graham Kendall and Stephen Shaw

School of Computer Science & IT, The University of Nottingham, United Kingdom
{gxk,sds98c}@cs.nott.ac.uk

Abstract. Cribbage is (normally) a two-player card game where the aim is to score 121 points before your opponent. The game has four stages, one of which involves discarding two cards from the six cards you are dealt. A later stage scores the four cards in your hand together with a card cut randomly from the deck after the discards have been made. The two cards that were discarded are used to form another hand, when combined with the two discards from your opponent. This additional hand is referred to as the crib or box and is scored alternatively by you and your opponent. In this work, we investigate how a strategy can be evolved that decides which cards should be discarded into the crib. Several methods are investigated with the best one being compared against a commercially available program.

1 Introduction

Game playing has a long research history. Chess has received particular interest culminating in Deep Blue beating Kasparov in 1997, albeit with specialized hardware [1] and brute-force search. Chess is, arguably, a solved game but it is still of interest as researchers turn to adaptive learning techniques which allow computers to 'learn' to play chess, rather than being 'told' how it should play [2]. Adaptive learning was being used for checkers as far back as the 1950's with Samuel's seminal work ([3], re-produced in [4]). Checkers research would lead to Jonathan Schaeffer developing Chinook, which claimed the world title in 1994 [5]. Like Deep Blue, it is arguable as to whether or not Chinook used AI techniques. Chinook had an opening and ending database. In certain games it was able to play the entire game from these two databases. If this could not be achieved, a form of mini-max search, with alpha-beta pruning was used. Despite Chinook becoming the world champion, the search has continued for an adaptive checkers player. Chellapilla and Fogel's [6] Anaconda was named due to the stranglehold it placed on its opponent. It is also named Blondie24, this being the name it used when competing in Internet games [7]. Anaconda uses an artificial neural network (ANN), with approximately 5000 weights, which are evolved by an evolutionary strategy. The inputs to the ANN are the current board position and it outputs a value which is used in a mini-max search. During the training period, using co-evolution, the program is given no information other than a point score over a series of games. Once Anaconda is able to play at a suitable level, it often searches to a depth of 10, but depths of 6 and 8 are also common in play.

J. Schaeffer et al. (Eds.): CG 2002, LNCS 2883, pp. 29–41, 2003.

Poker also has an equally long research history with von Neumann and Morgenstern [8] experimenting with a simplified, two-player version of poker. Findler [9] studied poker over a 20 year period. He also worked on a simplified game, based on 5-card draw poker with no ante and no consideration of betting position due to the computer always playing last. He concluded that dynamic and adaptive algorithms are required for successful play and static mathematical models were unsuccessful and easily beaten. In more recent times three research groups have been researching poker. Jonathan Schaeffer (of CHINOOK fame) and a number of his students have developed ideas which have led to POKI, which is, arguably, the strongest poker playing program to date [10]. It is still a long way from being able to compete in the World Seies of Poker (WSOP), an annual event held in Las Vegas, but initial results are promising. Schaeffer's work concentrates on two main areas. The first research theme makes betting decisions using probabilistic knowledge [11] to determine which action to take (fold, call, or raise) given the current game state. Billings et al. also uses real time simulation of the remainder of the game that allows the program to determine a statistically significant result in the program's decision making process. Schaeffer's group also uses opponent modeling [12]. This allows POKI to maintain a model of an opponent and use this information to decide what betting decisions to make. See [13] and [14] for good discussions on automated poker.

Koller and Pfeffer [15], using their GALA system, allow games of imperfect information to be specified and solved, using a tree-based approach. However, due to the size of the trees they state "...we are nowhere close to being able to solve huge games such as full-scale poker, and it is unlikely that we will ever be able to do so." Luigi Barone and Lyndon While recognize four main types of poker player; Loose, Tight, Passive, and Aggressive [16, 17]. These characteristics are combined to create the four common types of poker players: Loose Passive, Loose Aggressive, Tight Passive and Tight Aggressive players. A Loose Aggressive player will overestimate their hand, raising frequently, and their aggressive nature will drive the pot higher, increasing their potential winnings. A Loose Passive player will overestimate their hand, but due to their passive nature will rarely raise, preferring to call and allow other players to increase the pot. A Tight Aggressive player will play to close constraints, participating in only a few hands which they have a high probability of winning. The hands they do play, they will raise frequently to increase the size of the pot. A Tight Passive player will participate in few hands, only considering playing those that they have a high probability of winning. The passive nature implies that they allow other players to drive the pot, raising infrequently themselves. In their first paper [18] they suggest evolutionary strategies as a way of modeling an adaptive poker player. They use a simple poker variant where each player has two private cards, there are five community cards and one round of betting. This initial work incorporates three main areas of analysis: hand strength, position and risk management. Two types of tables are used, a loose table and a tight table. The work demonstrates how a player that has evolved using evolutionary strategies can adapt its style to the two types of table. In [16] they develop their

work by introducing a hypercube, an n-dimensional vector, used to store candidate solutions. The hypercube has one dimension for the betting position (early, middle and late) and another dimension for the risk management (selected from the interval 0..3). At each stage of the game the relevant candidate solutions are selected from the hypercube (e.g., middle betting position and risk management) and the decision is made whether to fold, call or raise. They extend the dimensions of the hypercube to include four betting rounds (pre-flop, post-flop, post-turn and post-river) and an opponent dimension so that the evolved player can choose which type of player it is up against [17]. The authors report that this player out performs a competent static player.

Cribbage is a two player card game where the aim is to get 121 points before your opponent. One of the distinguishing features of the game is the board that is used to "peg" the points. The board has sixty holes for each player, arranged in two rows. A player has to complete two circuits of the board to make 121 points (the final point comes from "pegging off"). Like Bridge, cribbage is a multi-stage game. Unlike Bridge, which has two stages, cribbage has four stages.

The first stage is concerned with discarding some of the cards you are dealt into a "crib" (or box). In the second stage, players alternate playing cards onto the table, trying to earn points which are determined by the rules of the game (for example, playing sequences of cards which total fifteen or 31, playing pairs of cards etc.). The third stage allows the players to use their cards (which they retrieve from the table) to make various card combinations (cards which total fifteen, pairs of cards, runs of cards etc.). In playing these cards a community card, which is cut from the deck after the discards have been made, is also used (see below for an example of the play and scoring). The final stage allows one of the players to play the cards in the crib, creating card combinations as they did in the previous stage for their own hand. Playing of the crib alternates between the players. For a two player game (by far the most common) the players are dealt six cards and have to discard two. This leaves each player with four cards and also gives the crib the same number of cards. As an example of the play and scoring, consider the hand {9♣, 9♢, 9♠, 6♡, 5♣, 5♠}. The player holding this hand might decide to discard the 5♣ and 5♠ into the crib. If the cut card came as 6♢ the player would score two points for each total of fifteen they could achieve; in this case twelve points as each nine can be paired with each six. They would also score two for the pair of sixes and six for the three nines (i.e. three ways to make a pair of nines). Therefore this hand would score (12+2+6) = 20, a very good score. However, if it were not your turn to play the crib, you may not want to discard the two fives into it as you immediately give your opponent two points and if they also discard cards valued at ten or five (or one of these cards is cut) then their points start to accumulate rapidly. It can be appreciated that there is a certain amount of luck involved in the game, a certain amount of skill, and some strategy as you need to be aware who will play the crib and play accordingly. For a more complete description of the rules see any book on cards games (e.g., [19]). The rules are readily available on the Internet (e.g., `http://www.pagat.com/adders/crib6.html`).

The academic literature for cribbage is limited. O'Connor applied temporal difference reinforcement learning to teach a multi-layer perceptron to play cribbage through self-play [20]. A technique that has also been successfully applied to backgammon is temporal difference learning [21]. Martin considers the optimal hand values [22] and states that there has been little previous work carried out for cribbage (his thesis contains no references!). This paper aims to create a cribbage player that evolves a good discard strategy (stage one from above) and to maximize the score in the players hand (stage three). In conducting these experiments we will ignore the suit of the cards. That is, the deck is made up from cards of ace through king but the cards have no suit. This decision was made as the number of possible hand combinations is 18,395 by not including suits but prohibitively large if we include suits. In fact, for cribbage, the suits play a minor part in the game so not including them does not prevent the proposed technique from being investigated. Assuming the technique is successful, the method could be extended to include suits for a future implementation.

2 Evolutionary Strategies

Evolutionary strategies (ES) are closely related to genetic algorithms and evolutionary programming. Originally they used only mutation, only used a population of size one, and were used to optimize real-valued variables. More recently, ES's have used a population size greater then one, they have used crossover and have also been applied to discrete variables [23, 24]. However, their main use is still in finding values for real variables by a process of mutation, rather than crossover.

An individual in an ES is represented as a pair of real vectors, $v = (x, s)$. The first vector, x, represents a point in the search space and consists of a number of real valued variables. The second vector, s, represents a vector of standard deviations.

Mutation is performed by replacing x by

$$x^{t+1} = x^t + N(0, \sigma)$$

where $N(0, \sigma)$ is a random Gaussian number with a mean of zero and a standard deviation of s. This mimics the evolutionary process that small changes occur more often than larger ones. In evolutionary computation there are two variations with regard to how the new generation is formed. The first, termed $(\mu + \lambda)$, uses μ parents and creates λ offspring. Therefore, after mutation, there will be $\mu + \lambda$ members in the population. All these solutions compete for survival, with the μ best selected as parents for the next generation. An alternative scheme, termed (μ, λ), works by the μ parents producing λ offspring (where $\lambda > \mu$). Only the λ compete for survival. Thus, the parents are completely replaced at each new generation. Or, to put it another way, a single solution only has a life span of a single generation. In this initial investigation, we use a 1+1 ($\mu = 1$ and $\lambda = 1$) strategy but plan to investigate other strategies in the future. Good introductions to evolutionary strategies can be found in [25, 26, 27, 28, 29].

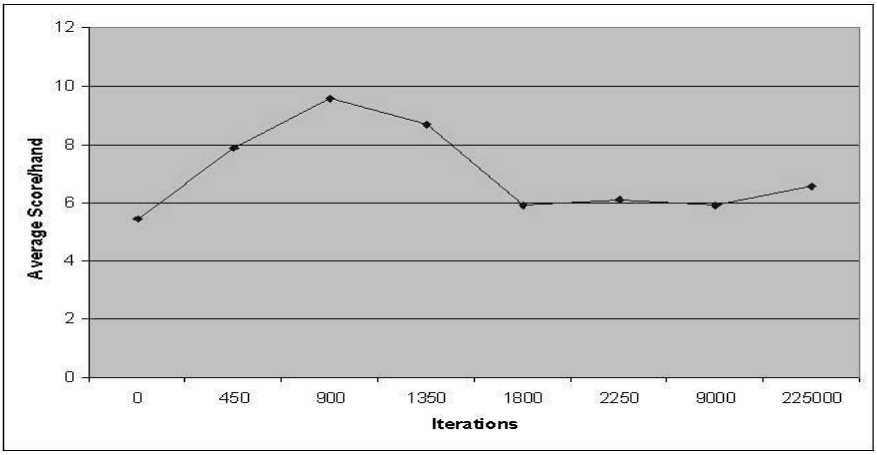

Fig. 1. Learning which cards to discard and constraining variables to 0.1..0.9.

3 Experiments

Each of the 18,395 possible six card hands was represented by the cards that make up that hand along with a real-valued variable for each card in that hand. These variables are used in the decision as to which cards to discard. The aim of an initial experiment was to ascertain if any learning could take place. A small set, n, of training hands was used. One iteration consisted of playing a single hand. It therefore takes n iterations to play the entire training sample. In this case $n = 9$ (although n is relatively small the hands were carefully chosen to represent both potentially low and high scoring hands). The values associated with each card for each hand were randomly set. The values were constrained between 0.1 and 0.9. In order to decide on the discard, the two lowest values from the hand were chosen. If the discard choice was deemed unsuccessful (see below), the values relating to the two cards that were discarded are updated by adding a random Gaussian number (standard deviation = 0.1). The discard was deemed unsuccessful if the point count was less than 75% of the optimal point count by considering all possible discards. The results arising from this experiment are revealing (see Fig. 1).

At the start of learning the average score per hand was approximately 5.5. After each hand has been played 50 times (450 iterations), there was an improvement with the average score being about 7. This increases further until the average score peaks at 9.56. After this the average decreases until it almost returns to the initial value.

Examination of one hand reveals why the rise and subsequent drop occurs. Hand seven, for example, consisted of a king, two queens, a ten, a five and a four. With no consideration for crib scoring, the optimum cards to discard are the ten or king and the four. This leaves a guaranteed 8 points (three 15s and

a pair), with a good chance of more if a ten or five is cut. After 900 iterations, examination of the hand array reveals the following (to 2 decimal places):

Card	K	Q	Q	10	5	4
Value	0.90	0.90	0.88	0.79	0.80	0.11

Choosing the two lowest values we correctly discarded the 10 and the 4, although most of the values are not especially far apart from one another. Examination of the array after 9000 iterations reveals the following:

Card	K	Q	Q	10	5	4
Value	0.90	0.90	0.49	0.90	0.90	0.90

Almost all the values have converged to 0.9. This results in the second Q being thrown, along with one other card at random. This convergence to the upper values explains why the system plays randomly after learning to play relatively well.

In an attempt to overcome the problem of the values converging to the upper bound the same experiment was conducted but without placing bounds on the variables. The results are shown in Fig. 2. They appear to support the argument that constraining the values held in the statistical array had a detrimental effect on the performance of the player after a certain period of learning time.

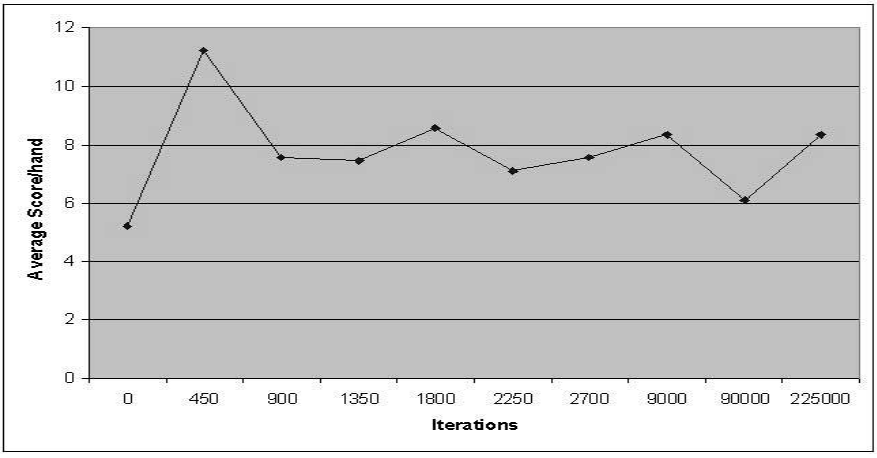

Fig. 2. Learning which cards to discard with unconstrained variable.

Discards are still at a reasonable level even after 225,000 iterations, with an average score of between 6 and 8. The peaks and troughs in the graph are explainable by the fact that cribbage relies on a certain degree of luck. The cut card plays a large role in determining the final score for each hand, with good discards sometimes not rewarded, and bad ones "getting a lucky cut." The figure

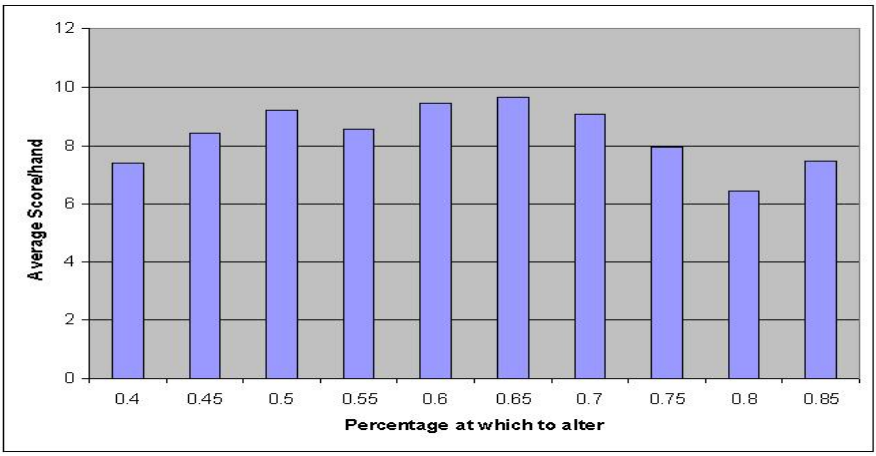

Fig. 3. Testing for the best cut off point p.

of 75% as being the cut off point, p, between a good discard and a bad discard was arbitrarily chosen. Intuitively, the value of p is important, as it determines how often the value relating to the discarded cards are altered, and therefore how often the discard pattern for that hand are changed. It seems likely that if p is set too low, then non-optimal discards will allow low scores to be seen as acceptable. If p is too high, then even good discards may be altered, if the cut card is an adverse one. Therefore, if p is set too high, the statistical array is too volatile. This means that a "final" discard pattern is never reached, as the player constantly evolves, trying to find a non-existent discard pattern that satisfies the over-stringent evaluation function.

In order to try and find a good quality value for p, a series of experiments were run that varied p and recorded the average score. The same set of training hands were used for each value of p, 250,000 iterations were executed. The results obtained are shown in Fig. 3. The best value for p appears to be 0.65. As suspected, if the value is too high or too low the scoring, in general, tails off. $p = 0.65$ can be further verified as a reasonable value by comparing the number of hands over which it appears to make good quality discards. With $p = 0.65$, 6 of the 9 test hands are played perfectly, resulting in an average score of 9.63 points per hand. Although this figure is a lot higher than the average score when the player has not been evolved, it is not clear why some hands have been learned and others have not.

Further analysis, over a greater number of test hands, showed that 0.65 is not optimal for every hand and it would appear that each hand has its own optimal value. It would appear that 0.65 is the value at which the largest number of hands can be learned optimally. Some hands will need values lower than this and some higher. One suggestion is to evolve the percentage value, as well as the card values, for each hand. However, this was discounted, preferring instead to investigate an approach, which did not rely on percentage values.

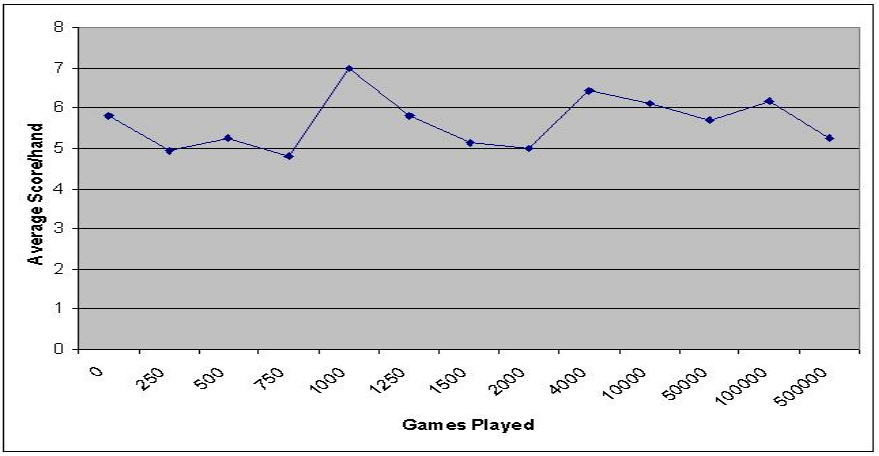

Fig. 4. Discard strategy using co-evolution.

A co-evolutionary approach was introduced by dealing one of the test hands to two players, so that they receive (normally) different hands from the training set. The players were then allowed to discard and the cut was made to reveal the community card before calculating the scores. The loser then altered their statistical array (i.e. added a Gaussian random number with a standard deviation of 0.1 to the variable associated with the cards that were discarded). The results can be seen in Fig. 4. This data was collected by dealing, at random, one of the test hands to two players and executing 500,000 iterations.

These results are disappointing as the player never gets close to the average previously recorded (about 9.63 using the strategy based around Fig. 2 with $p = 0.65$) and there exists no general upward trend. This would appear to indicate that learning is not taking place, although this is not entirely true.

When explaining why the mechanism for learning employed here is not successful, we need to examine in more depth the game of cribbage. As explained by Dan Barlow on his website (`http://zone.msn.com/cribbage/tips.asp`), cribbage involves a high degree of luck. This luck is mostly present in the form of the cards you are dealt, and the cut received. In other words, given better cards, the authors are capable of beating the best cribbage player in the world, although, given the same cards as an expert, the authors would be defeated. This leads to the conclusion that to be successful at cribbage depends not only on playing the hand you are dealt well, but also on getting good hands in the first place.

The nine test hands are all capable of achieving different scores. Consequently, the hands with the least potential are forced to constantly evolve, as they inevitably lose to better hands, no matter how well they are played. Throughout the learning process, we can see that hand four (which consists of 9,9,9,6,Ace,2) quickly converges to optimal values (that is discarding the Ace and 2). All the other hands, when dealt against this hand, will be forced to evolve, no matter

how well they are played. Therefore, it seems sensible to deal both players the same hand (even though this is sometimes not normally possible if playing with a physical deck of cards, for example when a player is dealt three nines) and let the player that plays it worse evolve. However, to stop the players converging to the same values and then playing the same way, but maybe sub-optimally, one of the players decides which discards to make based on the values stored in its array and the other player plays randomly. Only the player playing from the array will update its values should it lose (which is certainly possible especially given that the cut card adds a high degree of luck), on the basis that if it plays well it will beat the random player but should the random player win, then the player should evolve.

As the array will be altered less frequently, this means that the training period for each hand will be longer. These tests were conducted on the same set of training hands as previous experiments, and the results are summarized in Fig. 5.

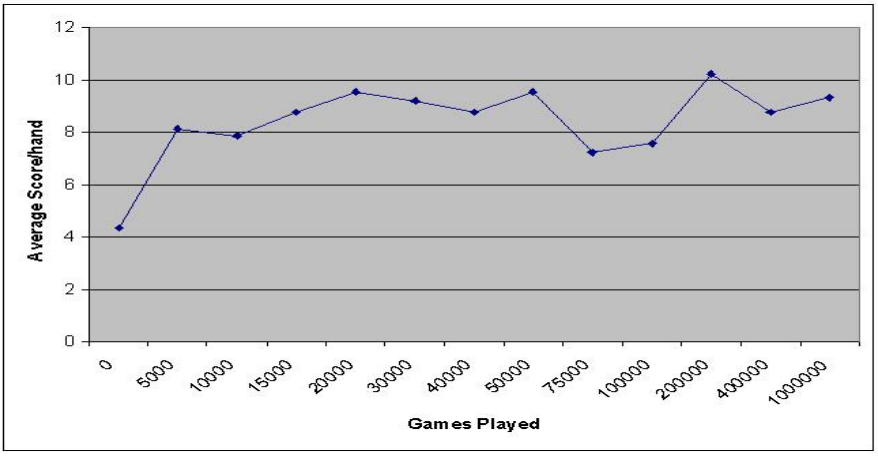

Fig. 5. Discard strategy using co-evolution (same hands dealt to both players).

The results for this experiment are comparable to the strategy based around Fig. 2 and with $p = 0.65$. There is an upward trend with peaks and troughs corresponding to the element of luck introduced by the cut card. Undeveloped, the player scored an average of 4.33 points per hand which increased to 9.33 after the training period.

In cribbage, the cards you discard are not just based on maximizing your own hand score but some attention must be paid to the crib. If it is your crib you also want to maximize that score but if it is your opponent's crib you will want to minimize the crib score which may involve reducing the score in your own hand. In order to model this, each hand now requires two sets of values, one

representing your turn to play the crib, the other set of values being used when it is your opponent's crib.

To test this idea, two random cards were placed into the crib to simulate the cards discarded by the opponent. If it is your crib, then the crib score is added to the hand score, but the crib score is deducted from your score if it is the opponents crib. As before, the best overall score is calculated, and if the score is not within 65% of the best score, then the array is altered. One test hand was created, which has different discard requirements depending on whose box it is. This hand consisted of 10, 9, 8, 7, 6, 5. If it is your crib then the best discard is likely to be 10/5. If it is the opponent's crib the best discard is probably 6/10 or 6/7, which preserves some points in your hand: and avoids giving your opponent a 5 (notionally the best card in cribbage as it allows scores of 15 to be easily made). In this initial investigation we only tested with one hand. However, if the adaptive player can be shown to adapt to different discards depending on whose crib it is for this single hand, then we believe it will be possible to adapt across a wide range of hands.

After playing the hand for 5000 iterations, the player was discarding the 10/5 when it was its own crib and discarding the 6/7 when it was its opponent's crib. The contents of the array were as follows:

Crib	10	9	8	7	6	5
Player	3.55	3.58	3.58	-1.83	-3.50	5.56
Opponent	0.73	2.54	2.60	2.50	2.52	-3.50

It is gratifying that the five card is at different ends of the scale for the different types of crib. We intend to investigate this area of crib strategy in later work but, as a final experiment, we decided to test one of our players against a commercially available program. To decide which evolved player to use we took the player represented in Fig. 2, with $p = 0.65$ and the player represented in Fig. 5, and played them against one another (after a 24 hour training period over all hands). The first of these players (based on Fig. 2) won a five game series 5-0 and was chosen as our champion. The game scores were as follows:

121-98	121-100	121-89	121-109	121-92

There are a number of versions of cribbage available, all of which are capable of playing at quite a high level. The version we decided to use was Ultimate Cribbage (UC) by Keith Westley. This decision was taken as Keith was the only person who replied to our help for assistance [30]. UC uses statistical methods and heuristics in order to calculate which cards to discard into the crib. The 'easy' level discards the first two cards from the hand, while the 'medium' level applies a sorting heuristic to find the best discards. The 'hard' level introduces additional rules to make the card selection even better. The 'harder' level reviews all possible card combinations and applies probabilities and observed card discard frequencies to calculate the best discards. In order to play against UC, a pegging routine was implemented for the evolved player. This algorithm was of poor quality with regards to playing cribbage, as we are really only interested in

the discard strategy and just needed a pegging algorithm to allow us to play the game against an opponent. It plays the first available card supplemented with simple heuristics (make 15 if possible, and lead from a pair).

3.1 Evolved Player versus UC (Easy)

Playing Ultimate Cribbage at its 'easy' level resulted in an easy win for the evolved player. This is not surprising as the commercial program simply takes the first two cards from its hand, therefore playing randomly. The results were as follows (evolved player first).

121-78	121-50	121-91	121-84	121-78

3.2 Evolved Player versus UC (Medium)

During this test, it was noticeable that the discards made by the evolved player were usually better than those made by UC, but the weakness of the pegging algorithm was laid bare, resulting in a tight game. However, the evolved player still won by three games to two. The scores for this match were as follows (evolved player first).

121-110	112-121	121-119	90-121	121-115

3.3 Evolved Player versus UC (Hard)

During this match it became apparent how important pegging is. The discards were of a comparable quality each time, but UC would usually score around 4-6 points more than the evolved player in each hand. This inevitably led to the evolved player being defeated; 5-0 (evolved player first).

100-121	98-121	110-121	96-121	115-121

It was suspected that this defeat was due to the evolved player's poor pegging so a rematch was held, this time discounting the pegging points. This resulted in a much tighter game with the evolved player winning 3-2 (evolved player first).

121-120	121-118	111-121	121-116	109-121

3.4 Evolved Player versus UC (Harder)

This match was one step too far for the evolved player and it lost the match, even based solely on discards, 5-0.

4 Discussion

It is gratifying to see that a player that evolves its discard strategy is able to compete with a commercial application and it demonstrates that players that have no strategy programmed into them are able to evolve strong playing styles that are able to compete with players that have been explicitly programmed with game strategy.

Now that this technique has shown promise in this game of incomplete information we plan to apply the same techniques (and developments of this technique) to other games of incomplete information (such as poker). In addition, we would hope (in the longer term) to use the same techniques in real-world "games" such as stock market prediction. However, we recognize this is a long term aim and one which is far from guaranteed to work. With regards to cribbage we also plan to develop this work further so that we are able to compete with the highest levels of play with regards to the discards and we are also planning to develop these techniques so that we can play the other phases of cribbage and thus compete against other programs and humans in all aspects of the game.

Acknowledgments

The authors would like to thank the referees for the helpful comments and also to the editors and their staff for their help in improving this paper.

References

1. Hamilton, S., Garber, L.: Deep Blue's hardware-software synergy. IEEE Computer **30** (1997) 29–35
2. Kendall, G., Whitwell, G.: An evolutionary approach for the tuning of a chess evaluation function using population dynamics. In: Congress of Evolutionary Computation. (2001) 995–1002
3. Samuel, A.: Some studies in machine learning using the game of checkers. IBM Journal of Research and Development **3** (1959) 210–229
4. Samuel, A.: Some studies in machine learning using the game of checkers. IBM Journal of Research and Development **44** (2000) 207–226
5. Schaeffer, J.: One Jump Ahead: Challenging Human Supremacy in Checkers. Springer-Verlag (1997)
6. Chellapilla, K., Fogel, D.: Anaconda defeats Hoyle 6-0: A case study competing an evolved checkers program against commercially available software. In: Congress on Evolutionary Computation. (2000) 857–863
7. Fogel, D.: Blondie24: Playing at the Edge of AI. Morgan Kaufmann (2001)
8. von Neumann, J., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press (1944)
9. Findler, N.: Studies in machine cognition using the game of poker. Communications of the ACM **20** (1977) 230–245
10. Billings, D., Davidson, A., Schaeffer, J., Szafron, D.: The challenge of poker. Artificial Intelligence **134** (2002) 201–240

11. Billings, D., Peña, L., Schaeffer, J., Szafron, D.: Using probabilistic knowledge and simulation to play poker. In: Sixteenth National Conference of the American Association for Artificial Intelligence (AAAI-99), AAAI Press (1999) 697–703
12. Billings, D., Papp, D., Schaeffer, J., Szafron, D.: Opponent modelling in poker. In: Fifteenth National Conference of the American Association for Artificial Intelligence (AAAI-98), AAAI Press (1998) 493–499
13. Billings, D., Papp, D., Schaeffer, J., Szafron, D.: Poker as a testbed for AI research. In Mercer, R., Neufeld, E., eds.: Advances in Artificial Intelligence, Springer-Verlag (1998) 228–238
14. Schaeffer, J., Billings, D., Papp, D., , Szafron, D.: Learning to play strong poker. In Fürnkranz, J., Kubat, M., eds.: Machines That Learn To Play Games, Nova Science Publishers (2001) 225–242
15. Koller, D., Pfeffer, A.: Representations and solutions for game-theoretic problems. Artificial Intelligence **94** (1997) 167–215
16. Barone, L., While, L.: An adaptive learning model for simplified poker using evolutionary algorithms. In: Congress of Evolutionary Computation. (1999) 153–160
17. Barone, L., While, L.: Adaptive learning for poker. In: Genetic and Evolutionary Computation Conference. (2000) 560–573
18. Barone, L., While, L.: Evolving adaptive play for simplified poker. In: IEE International Conference on Computational Intelligence. (1998) 108–113
19. Buttler, F., Buttler, S.: Cribbage: How to Play and Win. Cassell Illustrated (2000)
20. O'Connor, R.: Temporal difference reinforcement learning applied to cribbage (2000) `http://www.math.berkeley.edu/~roconnor/cs486`.
21. Tesauro, G.: Temporal difference learning and TD-Gammon. Communications of the ACM **38** (1995) 58–68
22. Martin, P.: Optimal expected hand values for cribbage. Technical report, Department of Mathematics, Harvey Mudd College (2000) (`http://www.math.hmc.edu/seniortheses/00/philip-martin-00.pdf`).
23. Bäck, T., Hoffmeister, F., Schwefel, H.: A survey of evolution strategies. In: International Conference on Genetic Algorithms, Morgan Kaufmann Publishers (1991) 2–9
24. Herdy, M.: Application of the evolution strategy to discrete optimization problems. In Schwefel, H., Männer, R., eds.: International Conference on Parallel Problem Solving from Nature, Springer-Verlag (1991) 188–192
25. Bäck, T., Fogel, D., Michalewicz, Z.: Handbook of Evolutionary Computation. Oxford University Press (1997)
26. Fogel, D.: Evolutionary Computation: The Fossil Record. IEEE Press (1998)
27. Fogel, D.: Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press (2000)
28. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag (1996)
29. Michalewicz, Z., Fogel, D.: How to Solve It. Springer-Verlag (2000)
30. Westley, K.: (2000) Personal communication.