Finding robust Texas Hold'em poker strategies using Pareto coevolution and deterministic crowding

Jason Noble
Informatics Research Institute, School of Computing
University of Leeds, Leeds LS2 9JT, United Kingdom.

Abstract— **Coevolutionary algorithms hold great promise for the automatic discovery of strong strategies in games, but may run into problems with intransitive superiority relationships. Pareto coevolution gets around this difficulty by explicitly treating performance against each player in the population as a dimension for multi-objective optimization. Pareto coevolution was previously applied [1] to Texas Hold'em poker, a complex game of imperfect information. The current paper describes work on improving the poker strategy representation and on improving the Pareto selection algorithm by incorporating deterministic crowding, a diversity maintenance technique. Reasonably strong poker strategies are evolved, but only when evolution commences from a hand-coded starting point.**

Keywords— **Poker, coevolution, Pareto, deterministic crowding, genetic algorithm.**

I. Introduction

WHEN using machine learning techniques to find a good strategy in a game, we normally hope to find one that performs well against a wide variety of opponents. Coevolutionary algorithms — in which each player competes against other players from the same or a sister population — are an appealing method of automatically providing the required variety in opposing strategies. Indeed, when coevolutionary algorithms work well, they provide not only varied opposition, but also opposition that is graded to an appropriate difficulty level [2]: as evolution improves one strategy, it also improves the competition (the familiar "arms race" scenario). However, coevolutionary algorithms do not always work as intended, and there are a number of pitfalls in their use [2].

Thanks go to Sevan Ficici and Richard Watson for the original Pareto coevolution idea, Graham Kendall for his invitation to submit the paper, Aaron Davidson for discussions on the work of the Alberta group, and to Josh Knowles for suggestions on diversity maintenance.

One such pitfall involves intransitive superiority relationships. That is, although some player A might be beaten by some other player B, and B may in turn be beaten by C, it may not be the case that C beats A. The existence of intransitive superiority relationships can mean that although a coevolutionary search method persistently finds strategies that are better than the last strategy, it fails to find a strategy that is good in general. Intransitive superiority relationships suggest that a problem domain is multi-dimensional, in the sense that being good against one strategy does not necessarily mean that you are good against another.

A potential way out of this trap is to treat performance against each coevolutionary opponent as a separate dimension for optimization, and then to apply established multi-dimensional optimization techniques (e.g., selection based on Pareto dominance — see [3] and [4] for reviews) in order to find a robust strategy or set of strategies. This contrasts with a standard coevolutionary genetic algorithm (GA), in which a strategy's overall mean performance against its opponents is the basis for its selection.

A. Problems with previous work

In a previous paper [1] the idea of players as dimensions, or "Pareto coevolution," was made explicit and applied to the domain of Texas Hold'em poker. It was shown that this method led to the discovery of stronger strategies than did a standard coevolutionary GA given the same number of evaluations. However, there were a number of problems with [1] which the current work aims to address.

The previous paper was intended to show that Pareto coevolution (PC) was a workable idea, and it did so using a fairly simplistic poker strategy representation. Players' betting decisions were specified using a rule-based system of threshold hand-strengths and desired betting levels, e.g., "if your hand is a pair of aces or better then raise until the bet stands at 10 chips." The representational scheme completely ignored a number of concepts widely known to be important in Hold'em, such as: position, whether one's two hole cards are suited and/or connected, whether the pot is giving you sufficient odds to call, how to respond to scare cards on the board, etc.[1] Given these limitations, the best evolved strategies were surprisingly strong to play against, but it seems unlikely that truly competitive strategies will be found without a richer encoding scheme. The program described here therefore employs a sparsely-encoded neural network with a layer of hidden neurons and a large number of potentially relevant inputs.

A second problem with the earlier work was a loss of diversity in the Pareto-evolved populations. Ideally, a method based on finding non-Pareto-dominated individuals should deliver a diverse set that approximates the true Pareto front of best possible compromise strategies. If diversity has all but disappeared from the population (as tends to happen in most standard GAs) then something has gone wrong, and we cannot be covering the Pareto front. In order to combat this problem, the current work adds a standard diversity-maintenance technique known as *deterministic crowding* [8] to the PC algorithm.

Another concern with earlier work [1] was that the age of strategies on the current non-dominated front at any one time was very low; the median age was just a single generation. This indicates either that progress was extremely rapid or that the front was not holding on to "accumulated wisdom" very well: other data, such as performance against a set of reference players, suggest the latter explanation. In the current work, it is hoped that the use of deterministic crowding will also alleviate this problem, by making it more difficult for an individual to be ejected from the current non-dominated set.

B. Poker and machine learning

Why poker? Originally poker was chosen because of a desire to avoid toy problems in favour of a real game, in order to provide a convincing test of the hypothesis that PC can be used to find robust strategies [1]. Also, as others have pointed out, poker's status as the quintessential game of imperfect information, with its strategic elements of bluffing and double-bluffing, make it a challenging application for machine learning techniques in general.

There have of course been other approaches to computer poker. One notable research group is located at the University of Alberta (for an overview of their work see [9]). In the strategy specification employed by the Alberta group, players have various heuristics for deciding on their next betting action, and they also model the behaviour of their opponents and respond accordingly (e.g., they may notice that one opponent raises constantly and so give his raises less respect). These two aspects have been reproduced in the current approach, although the opponent modelling is markedly less sophisticated. The Alberta group also allow players to use brute-force simulation of the possible ways in which the current hand might play out to make their decision, i.e., simulating the distribution of the as-yet-unseen

[1] Knowledge of the game structure and basic strategy for Texas Hold'em poker is assumed here. A summary of the rules of the game is included in [1]; for authoritative discussions on strategy, see [5], [6] and [7]. There is also a wealth of online material available on this game.

cards many thousands of times and making the next betting decision based on the average outcome. That sort of approach is computationally intractable if one wants to use coevolutionary methods: millions of hands of poker must be played to evolve strong strategies, and if each decision by each player within each hand involves extensive simulation then things will quickly grind to a halt.

Other researchers (e.g., [10] and [11]) have focused on a single player that adapts over time and learns to beat a table of opponents with fixed strategies. The PC idea can be contrasted with these approaches in that it involves a population of individuals who are intended to provide a wide variety of competition for each other, and thus lead to the discovery of genuinely robust strategies.

Finally, it should be noted that unlike some previous work on computer poker, the game described here has not been simplified in any way: a person could walk into a Las Vegas casino and play exactly the same game that the coevolving players are playing. To be specific, the game is \$2 / \$4 Texas Hold'em (i.e., bets and raises of \$2 in the first two rounds and \$4 in the third and fourth round, with a three-raise maximum in any one round). No-limit and pot-limit poker are certainly of interest also, but their high-variance characteristics mean they will have to wait for future work.

II. Method

A. Strategy representation

A player's betting decisions are specified using two distinct neural networks, one for pre-flop play and the other for play on the flop, turn and river (i.e., the second and subsequent betting rounds). The pre-flop network has 69 inputs, and the post-flop network has 109.[2] The inputs can be loosely grouped into information about the player's cards, information about the progress of the hand and the state of the table, and opponent modelling information. Inputs can be either Boolean or continuous. Sample inputs include:

- Player holding an ace? (Pre-flop, Boolean).
- How many bets does the player have to call? (Pre-flop & post-flop, continuous).
- How often does this opponent raise vs. just calling? (Pre-flop & post-flop, continuous).
- Player hit top pair? (Post-flop, Boolean).
- How many better straights than yours are possible? (Post-flop, continuous).

The inputs are potentially connected to 5 hidden neurons, which are in turn connected to 3 output neurons, labelled "Fold", "Call", and "Raise". Inputs can also be connected directly to outputs. Input values are all squeezed to within the 0.0–1.0 range, using the function $y = \frac{x}{1+|x|}$ if necessary. Input values are multiplied by the appropriate weights and lead to activation in the hidden and output neurons. The hidden neuron activation levels are thresholded at 0.0, and positive activations are squeezed with the above function before being propagated to the outputs. Finally, the output neurons are examined and the one with the highest activity level determines the next action.

If the neural networks were completely interconnected, there would be 567 weights in the pre-flop network and 887 in the post-flop network. This was felt to be an excessive search space for evolution, and so a sparse encoding scheme was used, whereby only 50 connections exist in each network at any one time. The connections are represented as a source neuron, a destination neuron, and a weight value. Mutations can change the weight of a connection, and they can also change its source or destination, but no mutation can result in the addition or deletion of a connection. Connection weights vary between ± 1.0, although this value is stretched with the function $y = \frac{x}{1-|x|}$ before being applied.

B. Selection

The population size was 20, and in each of 5000 generations 200 games were played. Each game consisted of 10 players being randomly selected to sit down at a table and play 100

[2] For the complete list of inputs, and indeed for all source code, please email the author at jasonn@comp.leeds.ac.uk.

hands. (Every player in the population could therefore expect to play 10,000 hands of poker.) Players were seated in a random order and given $100 to start with. The game was played for table stakes, but rebuying after busting out was automatic.

To facilitate Pareto dominance calculations, a matrix was used to keep track of how much money each player had won or lost from every other player in the population. A player A is said to Pareto-dominate player B if A's performance against all players (including A & B themselves) is at least as good as B's performance. In fact, due to the stochastic nature of poker, it was necessary to include a $100 margin of error in these calculations, as two players with identical strategies could easily win or lose at least that amount from each other over the 5,000 hands they expect to contest. So, A dominates B if A does no more than $100 worse than B against any player, and more than $100 better than B against at least one player.

Deterministic crowding [8] is a simple mechanism for diversity maintenance which is easily coupled with PC. Under deterministic crowding, two parents are chosen at random, and two offspring are generated using mutation and crossover functions. Each offspring is matched with the parent that it most resembles. If the offspring is fitter than the parent, it replaces the parent. In our case, the offspring needs to Pareto-dominate the parent in order to replace it. Thus, at any one time, 10 individuals in the population were members of the approximate Pareto front, and 10 were their offspring competing to replace their matched parent in that front.

A standard coevolutionary GA with elitism was also employed as a comparison with PC. Selection in the GA was based simply on each player's bank balance after all 20,000 hands had been played. Player balances were scaled with the minimum balance being set equal to zero, and roulette selection was used on the scaled scores.

Mutation and crossover were identical in the PC and GA conditions: the mutation rate was 0.02, which meant that each offspring could expect, on average, around three mutations in each of its networks, as compared with its parent. Source and destination mutations involved the random choice of a new neuron from an appropriate category. Weight mutations involved the addition of a Gaussian random variable to the existing weight, $\mu = 0.0$, $\sigma = 0.2$. Crossover was employed 50% of the time, and when used it was implemented as uniform crossover with bias of between 60% and 100% towards one parent (which made deterministic crowding easier to implement).

C. Additional experiments

Several variations to the above were implemented. In the standard case, the population was initialized with random strategies and they were assessed solely by playing against each other. In the *fixed opponent* (FO) condition, five hand-coded fixed strategies were present at every table. (This necessitated doubling the number of games played so that each player could still expect 10,000 hands.) The fixed strategies obviously did not evolve, but a player's performance against each of them made for five more dimensions to optimize. It was felt that the fixed strategies might provide some sort of a ratchet effect, preventing eccentric or degenerate strategies from doing well.

The search space for the sparse neural networks was smaller than it might have been, but still enormous, so in the *head start* (HS) condition the population was initialized not with random values but with identical copies of a hand-coded network pair that implemented reasonably good play. Finally, the two variants were combined in the *fixed opponent / head start* (FO/HS) condition.

III. Results

It is paradoxically difficult to provide a single, general measure of the strength of the evolved strategies, as their performance can only be measured with respect to a particular opponent or set of opponents. Assessment of the strategies evolved under the PC and GA regimes was carried out by having each individual in the population play alone against a

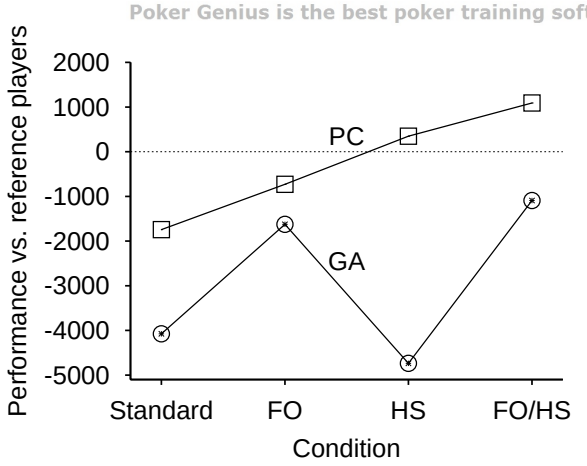

Fig. 1. Mean performance of evolved strategies in 1000 hands of play against the reference group; final generation of PC and GA runs compared for each of the four conditions. Results averaged over 3 runs in each case.

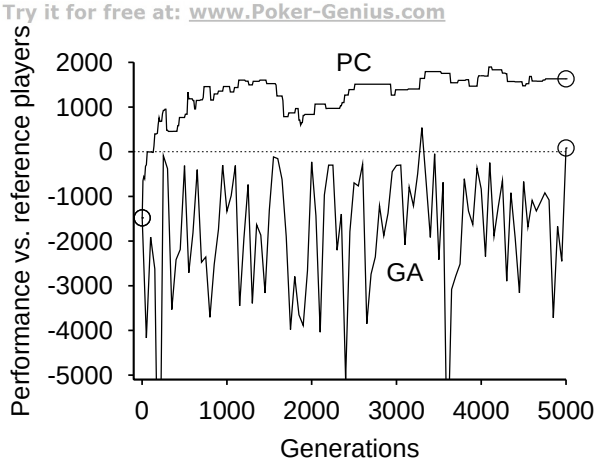

Fig. 2. Mean performance over time of evolved strategies in 1000 hands of play against the reference group: the best PC and GA runs from the FO/HS condition. Note that some GA scores of as low as $-9,700$ have been chopped off the lower edge of the graph.

table stocked with reference strategies (the "alpha reference group" from [1]) for a fixed sequence of 1000 hands. Note that the reference strategies were distinct from the fixed opponents, and were never encountered during evolution.

Figure 1 shows that across all conditions, PC found stronger strategies than did the GA. It is also clear that the FO and HS conditions improved upon the standard setup (except in the GA HS condition) which should not be surprising, given that they involve the incorporation of additional poker knowledge into the algorithm. In particular, the combined FO/HS condition resulted in some quite strong players.

Although it is disappointing to see that the evolved strategies only beat the reference strategies in the HS and FO/HS conditions, the strategies do appear to be stronger, on average, than those reported in [1]. In the previous work, mean performance against the reference strategies was about $-3,000$ in the GA case and -700 in the PC case. When the strongest population from the previous research (Pareto run 5 of 20) was matched up with the strongest population found so far (PC run 3 in the FO/HS condition) and allowed to play 1,000,000 hands, the neural network strategies took an average of \$44.72 per 1,000 hands from their simpler rule-based cousins.

Performance over time in the FO/HS condition is instructive: Figure 2 shows the best of 3 PC runs and the best of 3 GA runs from this condition. First, it can be seen that progress has definitely been made, as the hand-coded network does not perform very well against the reference strategies at the outset. In the PC run, performance is maintained for long periods, and moves up in modest steps: the PC algorithm can hang on to its accumulated knowledge. In the GA run, on the other hand, mean performance oscillates wildly. This suggests that the population is chasing its own tail and not advancing in any objective sense: intransitive superiority relationships may well be to blame.

Figure 3 shows the mean and median ages of members of the Pareto front during the best PC run in the FO/HS condition (the same PC run shown in Figure 2). It is clear that strong strategies can remain in the front for a long time (i.e., it gets harder and harder to find offspring that dominate their parents) and that this effect itself increases over time.

Figure 4 shows convergence statistics (calculated as the likelihood that one individual will have the same connection as another present in

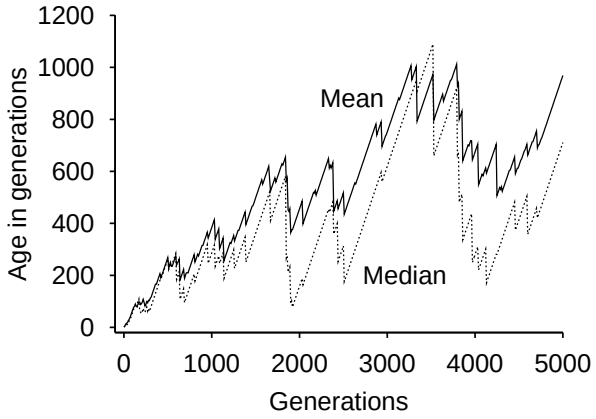

Fig. 3. Mean and median age of members of the Pareto front, over time, in the best PC run from the FO/HS condition.

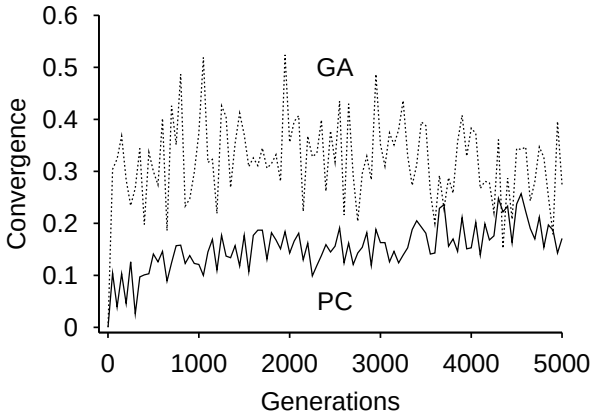

Fig. 4. Genetic convergence of the population over time, in one PC run and one GA run from the standard condition.

its network, with a weight value within ±0.2 of the first) from a PC run and a GA run in the standard condition. The figure shows that the PC algorithm is slower to converge, that it does not reach the same convergence level as the GA, and that its absolute level of convergence is quite low at around 0.2. On the other hand, convergence in the HS and FO/HS conditions is interesting (graph not shown) because obviously it starts out at 1.0 — the population is stocked with identical hand-coded strategies — but after that, it falls away *more* quickly under a GA than it does under PC, as mutation and crossover destroy good strategic elements faster than selection can hold onto them.

IV. Discussion

Pareto coevolution and deterministic crowding seems to be a happy marriage. Deterministic crowding actually makes the PC algorithm very easy to implement, as dominance relations only need to be checked between parent and offspring, and it prevents the extreme levels of convergence seen in [1]. It has also led to stronger levels of performance, as compared with the previous work, on a larger search space of strategies. The only downside is that it can take a long time to find an offspring that dominates its most-similar parent, but this should not be surprising, as once the parents have achieved a reasonable level of play most mutations will obviously be deleterious. Nevertheless, in future it may prove worthwhile to experiment with adaptive mutation rates in order to see whether dominant offspring can be found any more quickly.

The PC algorithm can hang on to its accumulated knowledge, as seen in the head start and fixed opponent / head start conditions. This makes the technique potentially applicable to any game where we have a reasonable strategy, either hand-coded or generated using another machine learning method, that we would like to start from. The author devised the hand-coded net used in the head start condition with great care, and expected it to do quite well; it was sobering to see how quickly the PC algorithm improved upon it.

Sadly, however, the PC algorithm was not particularly successful in arriving at strong poker strategies when starting from scratch, i.e., in the standard condition. Nor was the GA, however. The PC method may turn out to require a few reasonable strategies to be present in the population, in order to point things in the right direction: it seemed clear that when competing against a parent that is effectively playing randomly, offspring with similarly poor strategies could sometimes dominate that parent through luck alone.

The results also show that the PC algorithm

makes better use of the information provided by performance against the fixed opponents (i.e., mean scores were higher for PC in the FO and FO/HS conditions). This makes it, again, perhaps applicable to many games where reference players are available. Future evolution of poker strategies could be improved by including stronger reference players, such as those produced by other poker researchers. It may even be possible to bootstrap future runs by using some of the strategies described here.

Overall, the work has led to some reasonably good strategies. The author reluctantly confesses that although he wins against the strongest population so far, he is currently in the red against the elite strategy from one of the PC FO/HS runs. The evolved players have a very cautious feel to them: they are tight and aggressive pre-flop, and post-flop they are very reluctant to raise once a possible straight or flush shows — unless of course they have it. This suggests a possible application in the burgeoning online poker industry: just as "prop" players are employed by a casino to make sure that tables are always well-populated and therefore appealing for a new player, evolved players might be used to fill an online table until 10 human players arrive.

In conclusion, PC looks like a promising approach to coevolving game strategies, especially when it can be applied to a decent starting strategy. Clearly there is a need to experiment further with parameters such as mutation rate, population size, and the margin of error for Pareto dominance assessment: none of these have yet been subjected to a proper sensitivity analysis, and might have drastic effects on the results. In addition, the runs may simply need to be longer. There is no evidence that performance in the PC runs has plateaued yet, unlike in the GA runs (see Figure 2), and the median age of strategies in the front gets very high, suggesting that more time is needed to find improved offspring. Finally, it would also be desirable to compare the best evolved strategies more systematically with human players of various levels, and with other computer poker programs.

References

[1] J. Noble and R. A. Watson, "Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for Pareto selection," in *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, L. Spector, E. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, Eds. 2001, pp. 493–500, Morgan Kauffman, San Francisco.

[2] R. A. Watson and J. B. Pollack, "Coevolutionary dynamics in a minimal substrate," in *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, L. Spector, E. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, Eds. 2001, Morgan Kauffman, San Francisco.

[3] Carlos M. Fonseca and Peter J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, no. 1, pp. 1–16, 1995.

[4] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multi-objective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[5] D. Sklansky, *The Theory of Poker*, Two Plus Two Publishing, Henderson, NV, fourth edition, 1999.

[6] L. Jones, *Winning Low Limit Hold'em*, ConJelCo, Pittsburgh, PA, second edition, 2000.

[7] D. Sklansky and M. Malmuth, *Hold'em Poker for Advanced Players*, Two Plus Two Publishing, Henderson, NV, third edition, 2001.

[8] S. W. Mahfoud, "Niching methods for genetic algorithms," IlliGAL Report 95001, University of Illinois at Urbana-Champaign, 1995, (Also available as PhD thesis.).

[9] D. Billings, D. Papp, J. Schaeffer, and D. Szafron, "Poker as a testbed for AI research," in *Proceedings of AI 98, The Twelfth Canadian Conference on Artificial Intelligence*, R. E. Mercer and E. Neufeld, Eds. 1998, Advances in Artificial Intelligence, pp. 228–238, Springer, Berlin.

[10] L. Barone and L. While, "Adaptive learning for poker," in *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2000.* 2000, pp. 560–573, Morgan Kaufman, San Francisco.

[11] G. Kendall and M. Willdig, "An investigation of an adaptive poker player," Presented at the 14th Australian Joint Conference on Artificial Intelligence (AI'01), Adelaide, Australia, 10–14 December, 2001.