Coevolving game strategies: How to win and how to lose

Dr Evan J. Hughes
Senior Lecturer
Radar Systems Group
Department of Aerospace, Power, and Sensors,
Cranfield University,
Royal Military College of Science,
ej.hughes@ieee.org
Abstract

• This tutorial describes the application of co-evolution to two player games in order to discover effective value functions.
• The tutorial will describe the requirements for co-evolving value functions and typical co-evolutionary algorithm structures, with an emphasis on implementation issues.
• The main case study will look at the effects of different value function structures for the game of checkers, and give examples of structures that have been used for other games.
• The tutorial will examine value function structures that look very promising initially, but are extremely difficult to evolve - how to lose.
• We will then focus on structures that work, and describe why they work - how to evolve to win.
Outline

- Classic Player Algorithms
- Value Functions
- Typical Evolutionary Algorithm Structures
- Value functions - Winners and Losers
- Case study of Checkers
Player Algorithms
Direct game playing

- A function is used to decide which move should be made next, given the current board state, and possibly some history of the play-to-date.

- Genetic programming has been used to create functions (e.g. Ahlschwede with Mancala), but they would have to be incredibly complex to achieve strong play.
Minimax game playing

- Minimax is an exhaustive search strategy that expands the game tree from the current board position for a fixed number of piece moves (ply).
- The strategy assumes the opponent is going to make the worst possible move for the player and is therefore pessimistic.
- As the game tree can rarely be expanded to an actual win/draw/lose decision, a function has to be used to approximate the anticipated payoff.
Monte-Carlo Methods

- For complex games such as Go, the game tree is too vast and minimax is less practical, so the tree is just sampled.
- The Monte-Carlo approach is to make many thousands of random move sequences, starting with each of the possible moves from the current board position.
- The expected value of each of the current moves is then approximated and the highest apparent payoff taken.
Player algorithms - Summary

- Direct game playing is only really suitable for the simplest of games (tic-tac-toe etc.) and where high speed decisions are required.
- Minimiax is the workhorse of the games industry. The play performance is governed by the quality of the evaluation function, and the depth of tree that is searched.
- Monte-Carlo methods are very processor intensive, but can create good play on complex games.
- All methods can be enhanced with opening book move and end-game databases.
Value Functions for MiniMax
Hand-Coded Heuristic functions

- Classically the payoff for each board configuration has been decided using hand-tuned heuristics.
- Structure is often a weighted sum with one player positive, the opponent negative and a ‘draw’ represented by zero.
- Key strategies for many games are mobility and piece differential i.e. who has the largest quantity of useful pieces that can be utilised at short notice.
- Classic examples:
  - Chess - Deep blue
  - Checkers - Chinook
Genetic Programmes

- Genetic programming has been used to evolve fast evaluation functions.
- The chromosomes are often tree structures comprised of function nodes and information leaves, e.g. nodes such as +*=/= If_then_else and leaves being constant values, or states of given board locations.
- Large trees are often generated and careful choice of node and leaf types is important.
- Poor choice of leaf representations can make it difficult to recognise complex piece relationships.
Piece Counters

- Based partly on heuristic knowledge that the piece difference is fundamental.
- Weighted piece counters give each board location a weight.
- The weights may be time dependent, or change with who is predicted to win (e.g., aggressive play if winning, defensive if losing).
- Evaluation is often fast, but play can be ‘mechanical’.
Artificial Neural Networks

- ANNs allow non-linear piece-counter value functions to be constructed.
- Both Evolutionary and Reinforcement learning methods have been used to successfully train networks.
- The non-linearity often allows more ‘inspiring’ play to be expressed, when compared to linear piece counters.
- For large networks, the processing burden can be very high.
Board Representations

- In many games, different piece types exist with different characteristics.
- There are many different ways to represent the relative merit of the different pieces within the evaluation, for example, in checkers, a king is often included as equivalent to 1.3 men. A Queen in chess is often represented as worth 9 pawns.
- In essence, the evaluation is a multi-objective problem!
Mathematical Description

- The **Utility function** is often defined as 1 for a win, 0 for a draw and -1 for a loss.
- At each board configuration, the **Expected Value** of the utility function is the probability weighted average of the possible utility values,
  - eg. \( E = 0.6(1) + 0.3(0) + 0.1(-1) = 0.5 \)
  - Estimating probabilities is difficult (eg with Monte Carlo)
- **Material Value** is often what is calculated
  - \( M = 0.7(\text{sum(b)} - \text{sum(w)}) + 0.2(\text{sum(back_rank)}) \)
Evolutionary Structures
Nature of Evaluat or Evolution

- The objective is often calculated through coevolution as no specific training data is available.
- The evaluation is usually the aggregate of a small number of games and is therefore slow.
- The coevolutionary process leads to a noisy objective function.
- Often parallel EA algorithms are required.
- Generalisation of players is desired, rather than learning to defeat a single opponent.
- A small amount of randomness is usually added to the evaluator to prevent stagnation through excessive numbers of draws.
EAs for Noisy Co-evolution

- Simple Evolutionary Programmes are useful as they work well on rugged/noisy fitness landscapes. The added advantage is they are simple.

- Evolutionary strategies have also been used, although the self-adaptation of the mutation sizes can be upset by the noise.

- Generally, steady state with binary tournament or populations that replace only a small proportion are more consistent.
Objective Functions

- Examples of training evaluators to re-create decisions based on opening book moves often lead to non-generalised players - be careful.
- Often say 5 games are played and an aggregate score given - more games = less noise = slower evolution.
- Often win=1, draw =0, loss = -1 or similar
- Alternatives based on number of moves to win/loss can be effective at improving gradient, but can potentially modify play behaviour.
Value Functions - How to Create Winners and Losers
Value function Requirements - Winners

• The faster it can be processed, the deeper the minimax search can be
• The more non-linear the representation, the more scope there is for interesting play.
• **BUT**
  – The function must be evolvable ...
Heuristics - Winners

- The more heuristic and expert knowledge that can be incorporated indirectly, the better the play is likely to be.
- Prime example of key heuristic knowledge is adding piece difference to the final output through a single evolved weight.
- Piece difference can be evolved directly in a weighted piece counter, but it is not easy.
Observability - Losers

- The evaluator can only be trained to elicit an appropriate response to a given state if it has either:
  - visited the state regularly during evolution
  - visited a similar state and both states elicit a very dominant response (e.g., Definite win/loss)

- Any deficiencies in the training states will lead to some parameters having a very low evolutionary drive to settle at specific values.

- These parameters may drift erratically, or converge to an arbitrary value.
Objective Gradient - Losers

- If minor changes in the evaluator do not elicit changes in the objective function that stand proud of the noise often enough, the evolutionary process will be very slow and erratic, if it ever starts at all.

- Often the problem can be overcome by ‘bootstrapping’ the evaluator with some heuristic knowledge - eg. 1 gene to control the degree of piece difference added in.

- If the gradient is low, all initial random evaluators are awful and most games are very short - this compounds the startup problem as many gene values then have low observability.
Examples of Winners

- Genetic programmes which have piece difference and other key heuristics available directly as leaf nodes.
- Neural networks with heuristic values (e.g. piece difference) fed to output neuron directly through a single weight.
- Generally, the shorter the chromosome, the smaller the decision space, the more successful the evolution can be.
- Pattern matching structures such as convolutional neural networks can be effective at covering large boards with few genes.
Examples of Losers

- Genetic programming where the only leaf nodes that refer to the board are the value of individual tiles: to re-create piece difference requires a large complex chromosome.

- ANNs without any heuristic advantages: even though each hidden neuron can be evolved to be a piece counter, it is not easy. A second hidden layer will often stop evolution unless heuristic knowledge is applied.
Case Study - Checkers
Case Study - Checkers

- Two trials:
  - Simple weighted piece counter strategy evolution problem
  - Complex artificial neural network
Strategy evolution problem

- Many attempts at evolving players for games have used piece difference directly, or enforced symmetry in the evaluation function.
- How simple is it to evolve the anti-symmetry required in a simple weighted piece counter process?
- How well can a simple weighted piece counter play?
Board structure for checkers

Black moves first, only diagonal moves. If a jump available, it must be taken. Draw called after 100 moves each player.
Checker s and minimax

• A minimax strategy with $\alpha$-$\beta$ pruning was used, and run to 4 ply in the EA. No ply extension was used if the search tree ended on a forced move.

• The evaluation function consisted of four weights for each board location, depending on whether a black or white man or king was present. The sum of the active weights was passed through $tanh()$ to limit to $\pm 1$.

• A small amount of randomness was added to help prevent play becoming stuck in repetitive loops.
EA structure - piece counter

- A parallel ES(100+100) was used with each set of weights playing a total of 5 games against random players. A farming model was used with the population spread across 6 DEC Alpha processors.
- The chromosome consisted of 120 weights (28+32+32+28 allowing for the far row where men are promoted to kings)
- A win scored 1, draw scored 0 and loss scored -2.
- Games were played as black or white at random.
- A total of 7388 generations were used to evolve the set of weights.
Weight Distributions after 100 Generations
Rotated reversed weight distributions after 100 generations
Rotated reversed weight distributions after 7388 generations
Summary of weight evolution experiment

• After 100 generations, A rotational anti-symmetry is evolving with the white weights becoming negative.
• After 7388 generations, the magnitude of the equivalent weighting of black and white squares is very close.
• The weights of the kings have not evolved as well as the men, probably as they are seldom in play.
• The back rank of each player is favoured, with play quite uniform across other areas.
• Try evolution with rotational anti-symmetry enforced.
• A second EA that enforced rotational anti-symmetry was used with 60 weights and ran for 16593 generations.
Weights with enforced symmetry

weights, view from black squares - men

weights, view from black squares - kings

weight value

square number

weight value

square number
King Weight's Relative to Men

King/Man, view from black squares

weight ratio

square number
Piece location for black men

Piece location map for black men
Piece location for black kings
Comparison with original trial

Dashed line is maximum of black/white for original trial.
Summary of 2nd weight evolution experiment

- With anti-symmetry enforced, the playing ability evolved much faster.
- The weights for the kings still seem far less evolved than for the men.
- The average king weight with respect to a man is 1.23, although it appears to increase as play moves towards the opponent’s area.
- The values of the weights from both experiments have a similar profile and amplitude.
Playing Ability

- Played 1000 games to 8 ply against simple piece difference
  - We won 68%, drew 30%, lost 2%

- Against Xcheckers
  - We won 22%, drew 66%, lost 12%
  - Our play appears to improve with increasing ply.

- Played on Microsoft Gaming Zone, 30 games to 10 ply and reached a ranking of 2108. Player also competed in Evolutionary Checkers competition at WCCI 2002 and declared winner.
Initial Conclusions

- Piece difference can be evolved, but it may take a while to get good results.
- It is clear that the back-rank is favoured in the weights, but other strategies are difficult to see.
- If rotational anti-symmetry is enforced, play is simpler to evolve, and capable of significant results for such a simple algorithm.
Complex Artificial Network

- Fogel’s Blondie24 - large artificial neural network, 5048 real-valued genes to evolve.
- Used ES with population of 15 and 840 generations to evolve a player.
- Has been compared favourably against Chinook and performs well on Microsoft Gaming Zone, reaching Expert level.
Network Structure

Reproduced from: “Anaconda Defeats Hoyle 6-0: A Case Study Competing an Evolved Checkers Program against Commercially Available Software”
Kumar Chellapilla and David B. Fogel, IEEE 2000
Our trials - Brunette24

- Similar network structure to Blondie24 evolved for same 840 generations with population of 15.
- Small differences to Blondie:
  - Play at black/white was random, rather than biased to black,
  - Small amount of randomness was added to evaluation output.
  - No ply extension used when final move is a capture.
Playability of Brunette24

- Played 6 ply successfully on MSN gaming zone, but not enough games to generate a reliable ranking.
- Played twice against Blondie24:
  - First game Blondie24 black, Brunette24 white. Blondie scored narrow win.
  - Second game colours reversed, played to draw, Brunette24 a king up.
- Results inconclusive, but promising given that Blondie24 effectively plays deeper search.
Evolutionary state of Brunette24

- After 840 generations, with a population of 15, many of the games were playing to a draw, so evolution was slowing rapidly.
- The mean weight for the piece difference and the random number after 840 generations were 0.83 and 0.17 respectively. A king was worth 1.20 men.
- After 1882 generations, the weights were 0.86 and 0.01 respectively. A king was worth 1.36 men.
- The extra 1042 generations appear to have a significant fine-tuning effect on the weights.
- The 1882 player plays better than the 840 player, although more trials are required to see if the difference is significant.
- ‘bootstrapping’ effect of piece difference is significant - if the gene is set to zero, the gradient is poor and evolution never starts.
Key Conclusions

• Evolution is very capable of discovering primary strategies to games - as long as the evolving structure is simple enough, or has simple pathways that can act as a ‘bootstrap’ to provide sufficient initial gradient to the objective surface, when compared to the noise induced by coevolution.

• Genes related to regions/pieces not often in play are less observable and therefore difficult to evolve.

• Many evolved players do incorporate human expertise that was critical to their achieved success.
Any Questions?