Genetic Chess

Mark Harrison

May 2, 2017

Abstract

This work is a program for evolving chess-playing AIs. Through putting a population of AIs into chess matches, killing off the losers, and breeding the winners, it is hoped that one specimen will be able to stand up against a more traditionally developed engine (if only on the easiest difficulty setting). Though it is written in C++, it is the hope of the author that the style and architecture are comprehensible.

Contents

1 Building

Run `make` to create release and debug executables in a `bin/` subfolder that will be created if it does not already exist. If, in the course of working on this project, new files are created or the `#include` files are changed within a file, run `python create_Makefile.py` to regenerate the Makefile. This python script assumes all `*.cpp` and `*.h` files in the current directory and all subfolders are part of the project and need compiling.

2 Running

genetic_chess -genepool [file_name] This will start up a gene pool with Genetic_AIs playing against each other–mating, killing, mutating–all that good Darwinian stuff. The required file name parameter will cause the program to load a gene pool and other settings from a configuration file. A record of every genome and game played will be written to text files.

genetic_chess (-human|-genetic|-random) (-human|-genetic|-random) Starts a local game played in the terminal with an ASCII art board. The first parameter is the white player, the second is black.

- **-human:** a human player entering moves on the command line and seeing the results on a text-based drawing of the board. Moves are specified in standard algebraic notation (SAN) or in coordinates that indicate the starting and ending square.
- **-genetic:** a Genetic AI player. If a file name follows, load the genes from that file. If there are several genomes in a file, the file name can be followed by a number to load the genome with that ID. If no number is specified, the last genome in the file is loaded (presumably this one is the most evolved). If there is another file containing records of games played during a gene pool run that has the same name as the genome file name with an extra suffix "_games.txt", then the last genome with at least 3 wins will be selected.

-random: an AI player that chooses a random legal move at each turn.

Genetic Chess can communicate with GUI chess programs through the Chess Engine Communication Protocol, including xboard, PyChess, Cute Chess, and others. When using Genetic Chess this way, only specify the arguments for a single player (-genetic or -random). The program will then wait for communication from the GUI.

3 Non-evolutionary aspects

These sections describe the aspects of the chess AI that are not genetically modifiable, usually because of at least one of the following reasons:

- There is no sense in which the aspect of play is improvable. Any modification would be detrimental to the chess playing;
- It would take far too much time to evolve that aspect of play from a random starting configuration, or it would interfere too much with the evolution of other aspects;
- I cannot conceive of how to represent the state space of that particular play strategy so that it may be genetically encoded.

On the last point, it has been suggested to me that, instead of the specific genes listed in Section **??**, the genes should encode more abstract and generic strategies and heuristics for evaluating a board state. While this would probably better mimic biological evolution (wherein adenosine and thymine are rather neutral as to their teleolgy), I have no idea how to program such an abstract representation and how to translate the actional of such genes into chess moves. So, what results from all this programming is a glorified tuning algorithm for parameters in pre-defined genes with hard-coded meanings.

On the other hand, so is every other genetic algorithm (see [1], [2], and others). Plus, these genes can be deactivated through mutation, so these AIs are perfectly capable of telling me exactly what they think of my painstakingly crafted genes.[1]

3.1 Endgame Scoring

Winning gives an infinite score. Losing gives a negative infinite score. Draw gives zero.

Why not evolve these numbers? While the priorities of various genes can be varied to yield different playing styles, the only reasonable score to assign to a win is one that is larger than any other score. It can only be a disadvantage to prefer anything to a winning move. While this would result in upward evolutionary pressure on the score assigned to winning, it would stall the evolution

[1] The little ingrates!

of all other genes while the score assigned to winning was pushed high enough to always be preferred.

The specific values were chosen to make the scoring symmetrical between the two players, in that the score for one side is the negative of the score as seen from the other side (assuming the same player does the scoring). What is good for one player is bad for the other player by the same amount.

3.2 Mini-maxing

The principle behind the minimax algorithm is that the quality of a move is measured by the quality of moves it allows the opponent to make. Of course, the quality of those moves is measured by the quality of the moves that follow. Ideally, the only required board evaluation scores would be ∞ for a win, 0 for a draw, and $-\infty$ for a loss. Unfortunately, the number of positions to examine in a typical game is far too large to examine in a few minutes, so the search has to be cut off at some point and a heuristic evaluation employed to estimate the probability of winning from that stopping point. In this program, the decision to stop and the heuristic is genetically determined and evolved over many games.

3.3 Alpha-Beta Pruning

Alpha-beta pruning is based upon keeping a record of two game state evaluations:

Alpha: is the highest score that cannot be avoided by the opponent making different moves. That is, the player whose turn it is can force the game into a state with at least this score.

Beta: is the lowest score that the opponent can limit the current player to. If the current player finds a move with a higher score, then the opponent can make an earlier move that makes this game state impossible to reach (refuting it). Once such a move is found, the examination of moves from the current game state is abandoned, as the opponent will not allow this state to be reached.

Each time a move examination reaches a greater depth in the game tree, these values switch roles to represent the view of the board from the opponent's perspective.

There is one additional optimization implemented here in which the search is cut off if alpha represents a win at a shallower depth than is possible in the current game tree branch. If a checkmate can be forced in fewer moves, there's no point in looking for a win in more moves.

3.4 Principal Variation Recall

If the best move is chosen base upon the probable resulting future board state, and if the opponent makes the predicted move in that variation, then the moves

leading to that board state are examined first during the next move. This high-scoring board state should lead to early cutoffs from alpha-beta pruning, especially if that board state was a game-ending state. If that board state is actually avoidable by the opponent, then the shallower depth of that state during the next turn should lead to a faster refutation, leaving time to examine alternate variations.

4 The Genome

4.1 Regulatory Genes

4.1.1 Piece Strength Gene

This gene specifies the importance or strength of each different type of chess piece. It does not provide any direct action, but other genes reference this one for their own evaluation purposes–the Total Force Gene, for example.

4.1.2 Look Ahead Gene

This gene determines all aspects of time control. It does this by determining how much time to spend choosing a move, including how much time to spend on future moves. When choosing a move from the current board, the amount of time to examine each move and its consequences is divided equally among every legal move. This naturally limits the depth of search while allowing deeper searches for positions with fewer legal moves. If a move examination is cut off early for whatever reason (e.g., a game-ending move is found), then the remaining time for that move is available for as yet unexamined moves.

The amount of time to use in examining moves is determined by genetic factors indicating an average number of moves per game. The decision to look at future moves is determined by a genetically estimated rate at which positions can be examined and the amount of time left for this move. The distribution of moves per game is modeled with a Poisson distribution with a genetically variable mean parameter.

4.1.3 Branch Pruning Gene

This gene cuts off searching the game tree to greater depths if the current move lowers the score of a board state to less than the amount specified within the gene. If a move is sufficiently bad in the immediate state, then there's probably not going to be a dramatic recovery.

4.2 Board-Scoring Genes

4.2.1 Total Force Gene

This gene sums the strength (according to the Piece Strength Gene) of all the player's pieces on the board.

4.2.2 Freedom to Move Gene

This gene counts the number of legal moves available in the current position.

4.2.3 Pawn Advancement Gene

This gene measures the progress of all pawns towards the opposite side of the board.

4.2.4 Opponent Pieces Targeted Gene

This gene sums the total strength (as determined by the Piece Strength Gene) of the opponent's pieces currently under attack.

4.2.5 Sphere of Influence Gene

This gene counts the number of squares attacked by all pieces. Bonus points are awarded if the square can be attacked with a legal move. That is, if a piece cannot reach a square in one move (perhaps because such a move is blocked by another piece), then that square is still counted as falling under the influence of the side owning that piece.

4.2.6 King Confinement Gene

This gene counts the squares the king can reach given unlimited legal moves. This is a measure of how much room the king has to maneuver and escape. It is somewhat in opposition to the King Protection Gene.

4.2.7 King Protection Gene

This gene counts the squares that have access to the king by any valid piece movement that are unguarded by that king's other pieces. In other words, it measures how exposed the king is to hypothetical attacks. A higher score means a less exposed king.

4.2.8 Castling Possible Gene

This gene returns a positive score to indicate that castling is possible or has already happened. Score can vary based on the preference for kingside or queenside castling.

5 The Gene Pool: On the care and feeding of chess AIs

In each generation, the players in a gene pool are randomly matched up with each other to play a single game of chess. If the game ends with a winner, whether through checkmate or time violation, then the two players mate to

produce an offspring by picking each gene randomly from either parent with equal probability. The offspring is then subject to a a single extra mutation procedure wherein two genes on average are individually mutated. Finally, the offspring replaces the loser in the gene pool. This way, some the genes of losing players are passed on and only slowly weeded out since a single game does not actually provide much information about the fitness of any gene.

If the game ends in a draw, then one of two things happens. With high probability–currently 95%–the players are left as they are and will participate in the next round. The other possibility is that one of the players is randomly picked to be replaced by the result of mating the other player with either a randomly generated Genetic_AI, or any past Genetic_AI, even long dead ones. This happens with low probability because it destroys genetic information. However, it is necessary to keep a pool from stagnating due to never-ending drawn games. Bringing in a randomly generated AI injects new genetic information into the pool (though with low probability of high-quality information). Bringing back dead Genetic_AIs injects good information from the past and should help to keep a gene pool from becoming trapped in a self-reinforcing, pathological playing style that only works against similar players. In effect, both of these strategies are meant to kick a gene pool off of a local maximum of genetic fitness.

One final means of preventing gene pool stagnation and preserving genetic diversity is the use of multiple gene pools. Each gene pool evolves separately for a long time, allowing each to genetically diverge. Then, every once in a long while (the time being user-specified), the best player from each pool is transferred to the next pool over. Thus, the best genes are further spread afield so that they can be tested against a wide range of opponents. A useful measure of a pool's strength is how long its best player survives when it enters a new pool.

An example of typical output during a gene pool run is shown below:

```
Gene pool ID: 0  Gene pool size: 16  New blood introduced: 126 (*)
Games: 22024  White wins: 10469  Black wins: 9562  Draws: 2737
Time: 38.538 sec   Gene pool file name: pool.txt
    ID   Wins  Streak  Draws  Streak
 59014      9       7      1       0 T
 59031      3       3      0       0
 59054      1       0      2       2
 59055      2       2      0       0
 59074      1       1      0       0
 59077      1       1      0       0
 59078      1       1      0       0
 59081      1       1      0       0
 59095      0       0      0       0
 59096      0       0      0       0
 59097      0       0      0       0
 59098      0       0      0       0
```

```
 59099      0       0      0       0
 59100      0       0      0       0 *
 59101      0       0      0       0
 59102      0       0      0       0

59055 vs 59097: White!
59074 vs 59101: Black!
59102 vs 59100: White!
59096 vs 59014: None! 59014 mates with random / 59096 dies
59031 vs 59078: None!
59054 vs 59095: Black!
59098 vs 59099: Black!
59081 vs 59077: White!

Most wins:     18 by ID 20968
Longest lived: 27 by ID 45394
```

The `Streak` column indicates the current number of consecutive wins or draws a player has attained in the last few games. In the example above, player 59014 is on a seven-game winning streak following its last draw. Player 59054 has drawn its last two games.

The asterisk (`*`) indicates an offspring of the result of a drawn match. Note that the outcome of the game between 59096 and 59014 means that another such offspring will be brought into this pool for the next round. The (`T`) indicates that it is the best AI from another gene pool that has been copied to this pool.

6 Some Consistent Results

Here are a few results that are reliably reproduced in multiple simulations.

Piece values are rated in near-standard order

In descending order of valuation by a Genetic_AI:

1. Queen
2. Rook
3. Bishop and Knight nearly equal
4. Pawn

White has a slight advantage

Of the games ending in checkmate, white wins about 10% more often than black. Wins by time are shared by black and white equally.

The Total Force gene and the Pawn Advancement gene typically dominate.

The Pawn Advancement gene usually gains higher priority first, probably because it is the simplest gene that makes an immediate difference in the game. Push the pawns forward both threatens the opponent's pieces with low-risk attacks and increases the chances of promotion.

The Queen is the most popular piece for promotion.

Even when the Piece Strength gene has not been tuned at all, the queen is the overwhelming favorite, followed by the rook, then bishop, and finally the knight. In human games, only the queen and knight are chosen since they have different move patterns. If you need at least a rook or bishop, you might as well take a queen since that piece provides both. Only the knight provides a viable alternative.

As an example, the following is a count of all promotions in a gene pool run after more than 300,000 gaes.

Piece	Promotions
Bishop	2232
Knight	1648
Rook	7215
Queen	146664

Threefold repetition is the most common stalemate

Just like real games.

The Sphere of Influence gene typically counts legal moves as just as valuable as any other move.

This was unexpected. I thought that the legal moves would count more since they present a greater threat to the opponent. You cannot capture your opponent's Queen if your own King is in check. Perhaps Genetic_AIs find this gene more useful as a forward-looking view of the game.

7 Public Methods of Classes - Programmer's API

7.1 Piece

7.2 Move

7.3 Complete_Move

7.4 Board

7.4.1 Important Notes

The Complete_Moves returned by Board::get_complete_move() and Board::all_legal_moves(), as well as the const Piece* returned by Board::view_piece_on_square(), should be handled carefully as these structures are rendered invalid by the following two operations:

1. The originating Board is modified via Board::submit_move().
2. The originating Board is destructed.

Furthermore, a Complete_Move generated by one Board is only valid for another Board if it is an unmodified copy of the originating Board.

When calling Board::view_piece_on_square(), an empty square is represented by a nullptr.

7.5 Player

7.6 Clock

References

[1] G.S. Hornby, A. Globus, D.S. Linden, J.D. Lohn, "Automated Antenna Design with Evolutionary Algorithms." AIAA

[2] W.H. Miner, Jr., P.M. Valanju, S.P. Hirshman, A. Brooks, N. Pomphrey, "Use of a genetic algorithm for compact stellarator coil design." IAEA Nuclear Fusion, Vol. 41, No. 9. 1185-1195