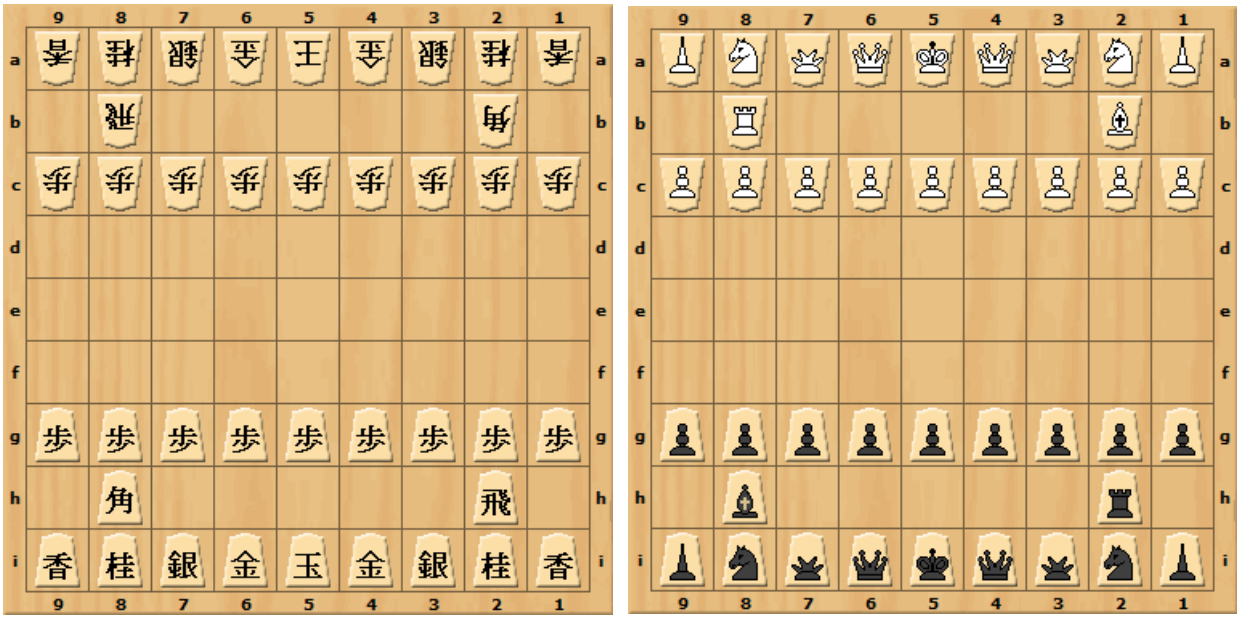
Ryan Cain
CPSC 490
09/22/2020

CPSC 490 Project Proposal: Shogi

Background:

Shogi, also known as Japanese chess, is a two-player strategy board game that sees origins of play as early as 1210. The game is played on a 9x9 grid (as opposed to an 8x8 grid for standard chess) with 8 types of starting pieces (King, Rook, Bishop, Gold General, Silver General, Knight, Lance, and Pawn) as shown below (Westernized pieces on the right).



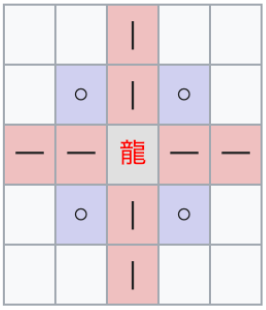
A few of the features that make shogi an interesting variant of chess are both the capture / drop rule and the mechanics of promotion. The drop rule specifies that captured pieces are retained *in hand* (手駒) and can be brought back into play ("dropped" on the bard) under the capture player's control. Thus, piece control and ownership is traditionally indicated by the orientation of the pieces rather than black or white as in western chess. A drop move comes at the cost of a player's ordinary turn and must be played instead of moving a piece already in play. Pieces can be dropped on any open space with the following three restrictions:

1. *Restricted Movement* （行き所のない駒）– Pieces cannot be dropped where they have no squares to move to.

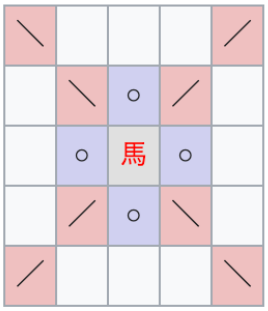
2. *Stacked Pawns*（二歩）– A pawn cannot be dropped in a column with another un-promoted pawn already in it.
3. *Pawn Mate*（打ち歩詰め）– A Pawn cannot be dropped to grant a checkmate. However, other pieces can be dropped for a direct check mate.

The second interesting play mechanic involves promotion. If a player moves any of their pieces into, or completely within, the last one third of the board – rows *a, b, c* for black and *g, h, i* for white – that piece is then eligible for promotion at the end of the turn (excluding turns where a piece is dropped into the promotion zone). Promoted pieces are grated increased range of movement and are indicated by a different character shown on the reverse of the piece. Some examples of promoted pieces are:

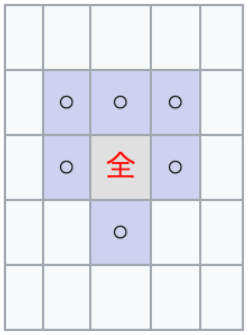
Promoted Rook:



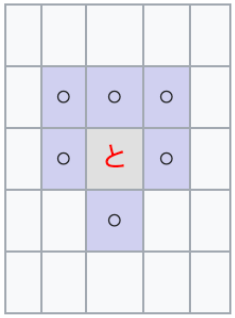
Promoted Bishop



Promoted Silver (moves as gold)



Promoted Pawn



The game ends under the same conditions as regular chess, when a player is unable to move their king. Although the mechanics of the two games are very similar, the larger game board, drop rule, and promotions mean that there is a much larger branching factor in Shogi,

leading to a complexity of around 10^{71} legal positions and an average game length of 140 moves, compared to 10^{47} and 80 for chess respectively[1].

Project:

In the fall of my Junior year I took CPSC 474: *Computational Intelligence for Games* with Professor James Glenn where I learned the fundamentals of game theory and built agents to play various common board games using the techniques discussed in class. In the following spring term, I took the first iteration of Professor Glenn's elective course, CPSC 674: *Advanced Computational Intelligence for Games* where we read and discussed papers in the field and worked on independent projects related to a game of choice.

My game of choice then was in fact Shogi, and I decided to apply the techniques discussed in the paper *Genetic Algorithms for Evolving Computer Chess* to Shogi. This paper utilized a three-step process for creating a competitive chess program capable of performing at or above grandmaster level. In summary, the first step involves evolving a set of parameters for a heuristic function based on samples of grandmaster games. Second, the process is repeated to generate multiple candidates before they compete in a coevolution phase to produce a single best set of parameters. Lastly, they evolve a set of parameters used in common selective search algorithms. Due to time constraints, mainly in that I had to convert my project from Python to C++ for efficiency, I was only able to produce the framework for step 1. For my senior project, I propose to use the framework I have created and continue by implementing the second and third stage required for making a well-playing Shogi agent.

Thus, there will be two primary objectives of this senior project. First, use the genetic algorithm I have written to tune parameters of my heuristic function and produce ten candidate organisms to co-evolve into a final set of weights; then second, incorporate those parameter values into a tree-search algorithm and evolve its search parameters.

From last semester, I already created a set of game features and a heuristic function that utilizes a C++ model of the game from this GitHub repo and interfaced evaluation with my genetic algorithm code written in Python. However, my heuristic evaluation is less than optimal due to my hasty porting into C++ and as a first step will need to be improved in

[1] For more information on complexity see the wiki here.

order to produce 10 offspring in a reasonable amount of time. Then, in Python I will need to implement a new co-evolution algorithm for the 10 candidate solutions to run and produce a single set of optimized heuristic weights. Lastly, I will need to implement a selective tree search algorithm in C++ that uses this heuristic function and then interfaces with one final genetic algorithm driven again in Python. The reason for using two languages is that Python provides efficient packages for genetic algorithms but is orders of magnitude too slow to handle repeated heuristic evaluation and tree search for Shogi.

Deliverables:

1. Finalizing the feature set of the heuristic function and improving the efficiency of calculating each individual feature value in C++. An objective way to measure this will be to compare evaluation times across a large sample of positions between the new and old features.
2. 10 (this number is not arbitrarily chosen, it is the same number used in the chess paper) individually evolved sets of weights for my heuristic function. Note that this is not the same as taking the top 10 individuals from a population since the parameters are initialized randomly. Due to how long the evolution takes, this is more complicated than just running Step 1 ten times. In order to be able to do so I have to deliver on step one because in the algorithm's current state it is too slow to be run ten times in one semester. Also, I will have to write scripts to keep track of the 10 instances, save their progress, and restart from a saved point in order to not hog Zoo resources or be booted partway through evolution.
3. A selective search algorithm written in C++ that utilizes my heuristic function and takes additional parameters for tuning.
4. A genetic algorithm written in Python to interface with and evolve the C++ search algorithm.
5. A final set of tuned search parameters, which in turn means all the pieces required for a complete, decently performing Shogi agent.
6. ** A final project report as outlined by the CPSC 490 guidelines.

** Ideally for my final report I will be able to produce an Elo score for the agent I create by playing it against other computer programs online via a service called Floodgate. However, I am not sure that I will have the time to adapt my code to the Universal Shogi Interface (USI) and attach it to the Windows Shogi GUI, Shogidokoro, in order to connect to the Floodgate servers.