Theo Bisdikian, Jeff Holm
Sebastian Garcia, Ryan Masson

CoupBot

Project Overview - All members

For our project, we implemented a genetic algorithm that produced an optimal bot to play the board game Coup. To do this, we first programmed a command line implementation of the game with gameplay functionally identical to the physical version. We then implemented a genetic algorithm that could randomly generate a population of bots and simulate games to train. The bots could train on the other bots in the generation, and then test on the baseline bot. We also included the option to do crossover and mutation (which are defined below) within each generation of the algorithm.

Bot Design - All members

We distilled the decision making process of gameplay to a large tree that would accurately model gameplay (see coupbot.github.io for the full tree as it is too large to include in this pdf). Each decision is represented as a node in the tree, and each edge weight between nodes is represented as a probability that the bot will choose the specific action. We represented the bot as a vector of 172 probabilities, one for each possible decision the bot could make. We traverse the tree using dictionaries; at each decision layer, there exists more sub-dictionaries which hold all of the possibilities for a given action. We prune out many of the probabilities using game state information (i.e. actions which are impossible given the game state), so the bot is only directly comparing between 2 and 7 actions at a given time. These bots are then stored as vectors of numbers in .csv files.

The Algorithm - Ryan, Theo, Jeff

For our first version of the algorithm, we created a population of 100 randomly generated bots. We designated one of our random bots as a test bot, which served as the baseline. We played each bot against each other 100 times to train, and then we used the win rates of each bot to perform selection for successive generations.

In our second iteration of our algorithm, we created a three step process: selection, crossover, and mutation. The selection step chooses two bots at a time, based on probabilities proportional to their fitness scores (win percentages during training). For each selected pair, crossover might occur at a static probability. We implemented two different crossover

techniques, but we ended up settling on single-point crossover (in which two bots exchange all chromosomes after a randomly chosen point). We implemented three different types of mutation: full random mutation, bounded mutation, and point mutation.

Parameters to assign:
CHROMOSOME_SIZE: Number of edge weights in the game tree
POPULATION_SIZE: Number of bots in each generation
NUM_GENERATIONS: Number of generations to run
UNIFORM_CROSSOVER_RATE: Swap an entire sub-dictionary of probabilities
SP_CROSSOVER_RATE: Swap one probability value
MUTATION_RATE: Random reassignment of a whole gene
BOUNDED_MUTATION_RATE: Random reassignment of a gene within bounds of old values
POINT_MUTATION_RATE: Random reassignment of an individual value in the vector
MUTATION_BOUND: Bound on mutations when using bounded mutation
NUMPLAYS: Number of runs of the game while training each bot

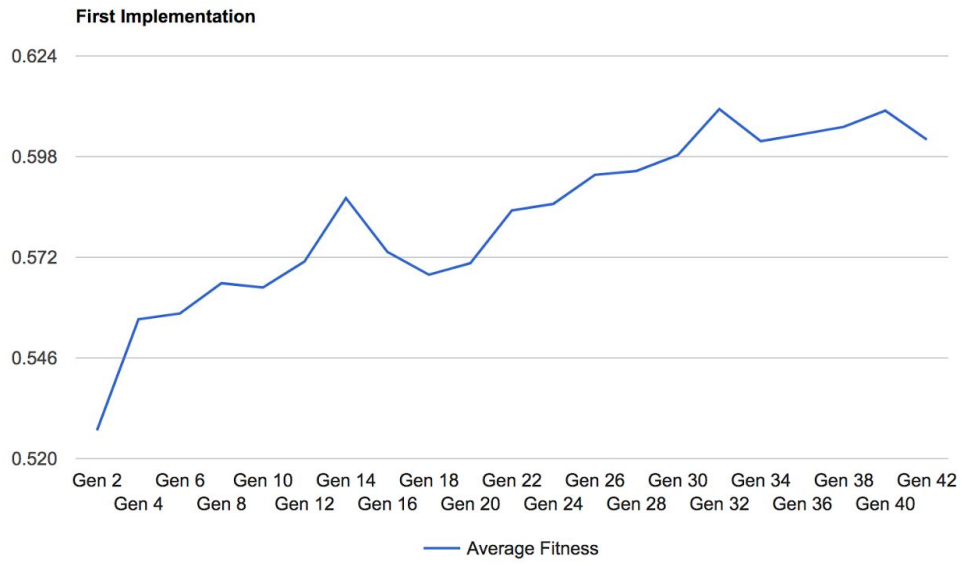
Overview

- Initialize population of random bots (feature vectors of game tree probabilities)
- Run NUM_GENERATIONS times:
    - Test all bots in the population and assign their fitness values (fitness values are win percentages)
    - Run until new population is full:
        - Select two bots at random, with probabilities proportional to fitness
        - Perform crossover some of the time
        - Place the two bots in the new population
    - Mutate some values, based on mutation rates
    - Write metadata to file

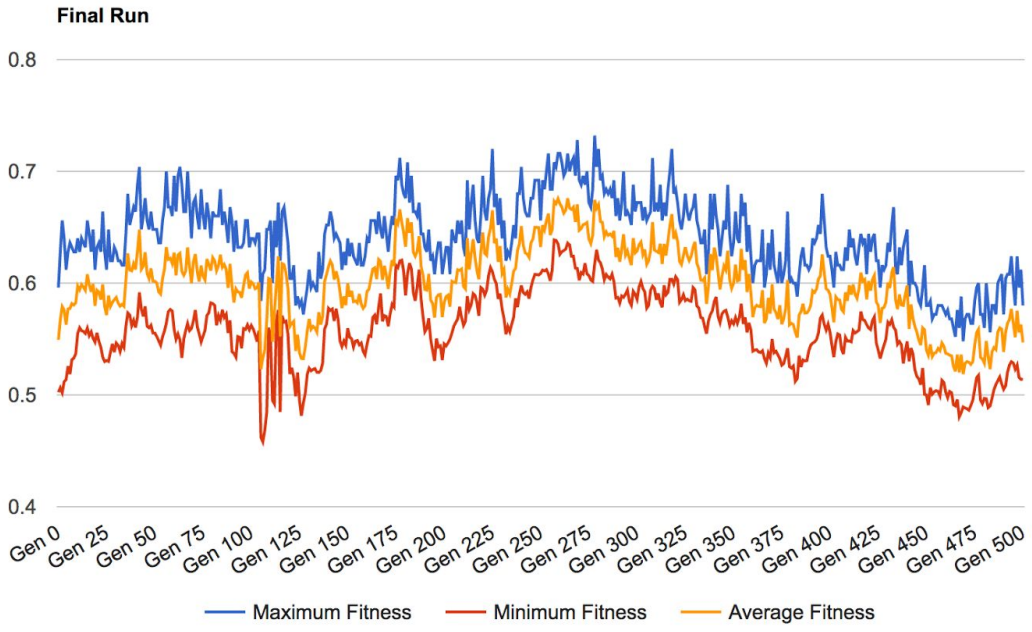
Results and Analysis - Sebastian, Theo, Ryan

Parameter	Run 1	Run 2
CHROMOSOME_SIZE	172	172
POPULATION_SIZE	100	100
NUM_GENERATIONS	100	500
UNIFORM_CROSSOVER_RATE	0.0	0.0
SP_CROSSOVER_RATE	0.0	0.01
MUTATION_RATE	0.0	0.0
BOUNDED_MUTATION_RATE	0.0	0.02
POINT_MUTATION_RATE	0.0	0.0
MUTATION_BOUND	0.0	5.0
NUMPLAYS	100	250

In our first iteration, our initial population of bots had a win rate of about 52%, and after 100 generations we achieved a peak win rate of about 61%. This showed us that, even without any of the crossover or mutations implemented, the bots were able to learn a strategy that was about 10% better than random. We were on the right track .



In our second iteration, we used crossover and mutation. Though we implemented all the possibilities for crossover and mutation explained above, we found the best results with single point crossover and bounded mutation, as you can see from the settings in the above table. The minimum, average, and maximum fitness values of bots in each generation are plotted against generation in the plot below. We see several instances of overall upward trends, with the most notable being from Generation 200 to Generation 275. The generation with the highest average fitness of its bots had an average of 0.675. Also, the best individual bot we achieved had a win percentage of 73.2%. This was an improvement over the genetic algorithm without crossover and mutation. In this second iteration, we showed that we could train a Coup bot that performed well against a specific opponent.



Further Work - you?

This project was a fascinating way to approach machine learning in a controlled environment, but one of its biggest shortcomings is that it only learned what strategies could beat randomly generated bots. The next obvious step would be to teach it how to play against humans. However, the dataset required to be substantial enough would be incredibly difficult to build. If datasets are acquired that accurately modeled enough (over 100) successful players, a bot that could mimic human strategy could possibly be built. Another shortcoming of our bot is that it doesn't take into account past actions other players have taken. Implementing a form of

short-term memory as well as some game-state heuristics to determine the favorability of a given board-state are more steps we can take to produce a bot that can stand against humans.