Sam Orfield && Jack Wines

Minesweeper Evolution

Introduction:

Problem Definition:

We're playing 8 by 8 minesweeper on a beginner board with 10 mines on it. This is the same to how minesweeper would play if you looked it up right now. We have one significant deviation from this. The game starts with a random zero on the board already clicked. This ensures at least 4 nonzero squares are presented to the player on first play.

Infrastructure:

We wrote the Minesweeper core ourselves. We also wrote the fitness function ourselves. The only outside code we're using is the Jenetics[1] library. This is a library that lets you choose a genome structure and selection method. We opted to use a genome of 250 patterns, each 11 integers long. Ultimately, that made for a genome of a list of 2,750 integers.

Approach:

As we mentioned before, we used a representation of a minesweeper board to model the problem. We used a single objective fitness, with the objective being the number of mines the algorithm correctly identified, minus 1.5*the number of spaces the algorithm incorrectly flagged. We did not use coevolution.

Our genotype consisted of a long string of ints containing 250 "patterns" of length 11. The first nine integers of each pattern corresponded a value on a hypothetical 3*3 piece of the board. The 10th index contained an action 0 is click, 1 is flag. The last index contained information for which square in the 3*3 to click or flag.

To play the game we first gave the individual a square that was not a mine. Then we told it iterate through all the frontier squares. Frontier squares are defined as clicked squares that have at least one adjacent space. At each frontier square, the individual would compare the frontier square, and the squares immediately adjacent to it, to each of the 250 patterns. If the pattern had the same center value as the frontier square, and

[1]http://jenetics.io

the pattern had the same number of adjacent spaces and adjacent flagged squares, then the pattern would be assigned a fitness such that patternFitness = center value - number of adjacent flags / number of adjacent spaces. We kept a running total of the lowest fitness on the board. When all frontier squares had been iterated through, the individual would find the frontier square with the lowest fitness pattern, and use the values at index 9 and 10 of the pattern to choose which adjacent space to click or flag, and then choose whether to click or flag that space.

This pattern would continue until the individual accidentally clicked on a mine. At this point, we would evaluate the fitness of the player. Game fitness is proportional to the number of mines that a player had correctly flagged, minus 1.5 times the number of spaces a player had incorrectly flagged. Our reasoning for the 1.5 is because otherwise, a player still has an incentive to be somewhat overzealous with flagging many spaces.

Baseline Tests:
We had two baselines. Our upper bound was a human solution, which involves playing the game as one normally would. It's figured that a person, in the long run, will correctly identify every mine. That gives them a per-game fitness of 74, which is the base 64 plus 1 point for every mine, minus 1.5 for every incorrectly flagged space. Our minimal solution was random, with half chance of clicking a random unmodified square and a half chance of flagging a random unmodified square. Once it hit a mine or ran out of squares to click and flag, the game would stop. Over a hundred trials, this led to a fitness of 59.65.

Figure 1: Results of Trials

Trial number	1	2	3	4	5	6	7	8	9	10	Mean
Tournament size of 12	65.16	64.85	64.93	65.11	64.87	65.08	65.17	64.92	65.07	65.07	64.999
Tournament size of 8	64.96	64.99	65.11	64.87	64.96	65.04	65.13	64.96	65.05	64.95	65.055
Tournament size of 4	64.94	65.1	64.87	64.92	65.19	65.06	65.15	65.24	65.12	64.96	65.003
Stochastic Universal selection	64.76	64.93	64.79	64.77	64.86	64.85	64.82	64.89	64.86	64.86	64.8367

Results and Analysis:

Overall, we had a moderate success. The data shows that any tournament selection is better than Stochastic Universal, and that a tournament size of 8 is the best tournament size, although not by a significant margin. The parameters for evolution on all trials were as follows. Every individual had its fitness determined by its average fitness per game over the course of 100 games on the standard beginner board. This went on for 250 generations. Variation in population size and in number of generations didn't help, giving the impression that this is the best result obtainable from this method.

To put those numbers in context, a board with no moves on it has a fitness of 64. A wrong flag has a penalty of 1.5, while a correct flag has a reward of 1. So a game played completely correctly has a fitness of 74, with every flag correctly labeled and no false flags.

So that means that our best individuals performed at one correct flagging per game on average. When watching the individuals play, they usually performed 2 or 3 flaggings before hitting a mine. Results varied strongly from game to game, depending on how much of the board the initial zero click would reveal.

Recursive Reflection:

One detour that we took was deciding how to evaluate fitness. At first, we'd implemented a fitness method that calculated fitness proportional to the amount of the board that had been revealed by a player that could only click. We decided, while this wasn't inaccurate, it did represent very well how a minesweeper player played the game. We abandoned this fitness evaluation method in favor of a method that looked at all shown squares on the board that are adjacent to spaces, determined, by examining the neighbors of these "frontier" squares, which would be the best to evaluate, chose which adjacent space to perform an action (click or flag) on, and performed the action until the algorithm hit a mine. The fitness was then calculated proportional to the number of mines correctly flagged, minus 1.5(the number of spaces incorrectly flagged). This produced much better results.

A second detour that we took had to do with the way in which we used pattern matching to determine which frontier square to evaluate. Originally, the genome for the

3*3 square representing each frontier square and the neighbors around it, generated random numbers between -2 and 8, where -1 is a flagged square and -1 is a mine and the rest are values. The fitness of the pattern was the sum of the differences between the values in the pattern and the values on the board. However, we realized that it very rarely matters what the values of the adjacent squares are. For example, when a 1 is on a corner, and there are no adjacent flags, that corner spaces is *always* a mine. This problem meant that our fitness gradient for determining patterns wasn't very good. Some patterns could have very small differences in value from the board, but could still be very useless when informing a click or flag decision. Therefore we reduced our patterns to only hold values for whether the adjacent square was a space, was shown (already clicked) or was flagged. We only evaluated patterns where the center square matched, and the number of adjacent flagged squares and spaces were the same. Then we assigned the fitness of that pattern at that spot to be the value of the (center square - the number of adjacent flagged squares) / number of adjacent spaces, and evaluated the frontier square on the board with the lowest overall fitness in a matching pattern. While most of the patterns in the genome are still never used, this dramatically improved the performance of individuals.

Toward the end of our project, we realized that our minesweeper individuals would not be able to solve minesweeper. As we stated in the previous paragraph, most of the randomly generated patterns in the genome will not or cannot ever match to a spot on the board. Currently, our selection only selects a very long string containing 250 of these patterns, and mutates and implements crossover to the entire string. In any given game, at least 200 of the patterns are never used. If we had more time, we would implement a type of mutation such that these vestigial patterns get replaced by copies, or mutated copies of patterns that do get used. We think this change would give each individual more close choices, and therefore the algorithm would be more likely to evolve a pattern that, for instance, can recognize what to do with a 1 on a corner.