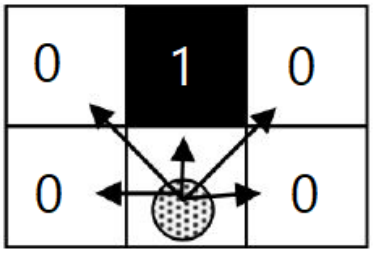
Syed Hasan Amin Mahmood
20100249

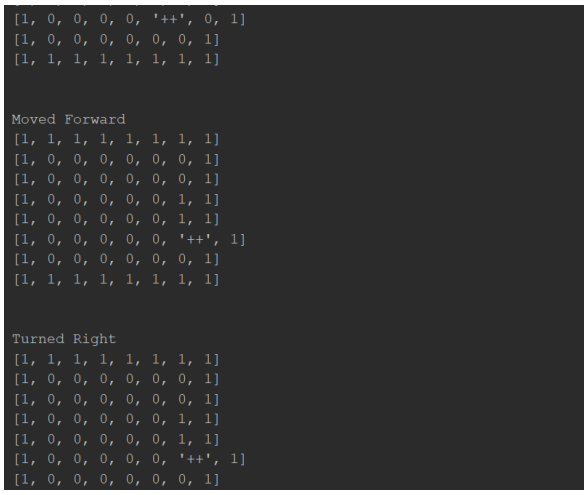
Evolutionary Intelligence Based Wall-Following Robots

In this assignment, we were asked to solve a simple wall-following problem using evolutionary algorithms. With genetic algorithms learnt in class as the foundation, I began implementing my solution, adding to what was learnt in class as deemed necessary for optimization *while also adhering to practical/ realistic constraints* (such as not forming 90% of the next generation from the fittest robot because *natural* evolution does not occur like this over subsequent generations).

Solution Overview

First of all, the environment/ arena was defined (as a list of lists), and a robot class implemented. To keep track of the robot's position a coordinate system was defined with top left cell representing (0,0), while the robot's direction was maintained using a variable based on the angle of the robot (North corresponds to 0°, East to 90° and so on). Having numeric representation of position and direction made manipulation of robot's direction and position convenient through basic arithmetic operations. The main problem was then solved drawing inspiration from Finite-State Automaton and Genetic Algorithms.

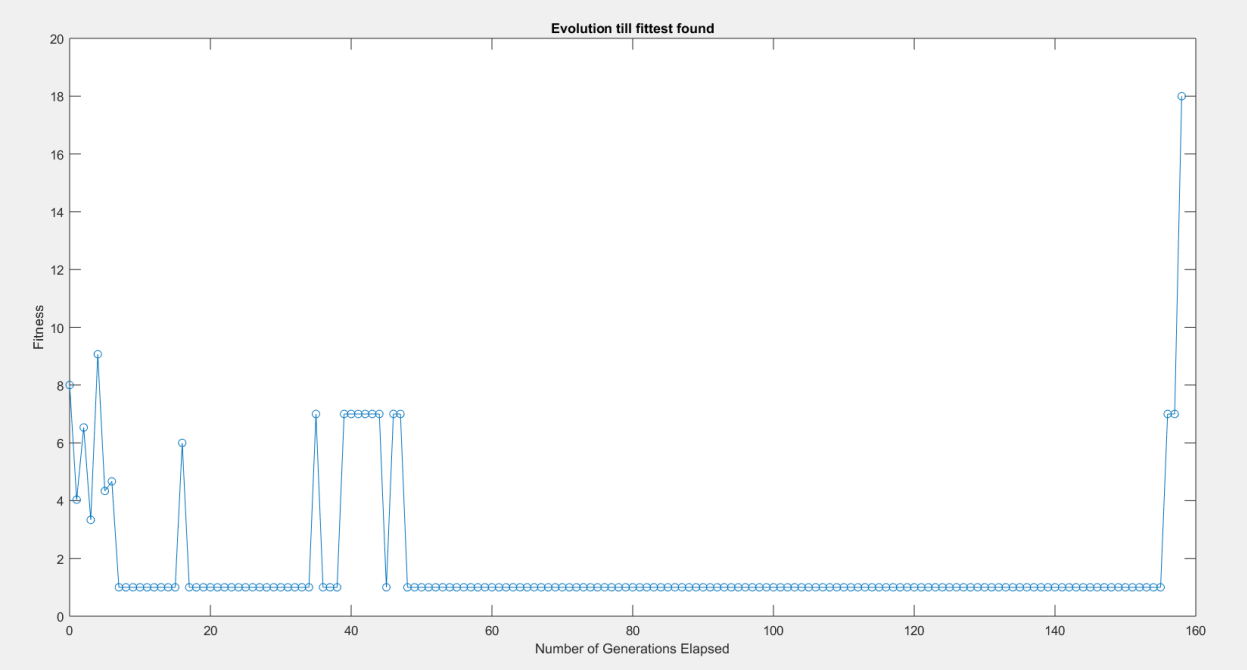




Each robot has five distinct sensors, detecting the presence (1) or absence (0) of wall. The output of the five unique (front, front-left, front-right, left, right) sensors was associated with one *state*. The robot thus became capable of identifying its state (00100 or 10111 and so on), and it was now a challenge to decide how to act in each state. With 5 sensors, and 2^5 = 32 states, a response list (of length 32) was prepared, each of its element corresponding to how the robot responded/ acted, having 4 possibilities (do nothing, move forward, *turn* right, *turn* left), when in one particular state (out of the 32 possible states). The response list was initialized randomly. At each generation/ stage, this response list was updated through a process that mimicked recombination and mutation. Robots with a better response list – having a higher *fitness* – i.e. having responses such that a better (wall-following) path was followed, have a higher likelihood of making it to the upcoming generation based on selection algorithm used. Therefore, more higher fitness robots get included in upcoming generations, until the whole population becomes really fit. One of these fit robots is then graphically put to test, as shown above.

Algorithm

```
Create list of n robots called current_gen with randomly filled response list
While(fitness < desired_fitness):
      Shuffle list of robots # to randomize
      Traverse this list choosing a pair (parents) at a time:
            Choose random start and end locations. Flip 2 response lists
            between those point // mimick recombination
            With probability prob_mutation, carry out mutation at each
            element of the response list
            Check fitness of offsprings and create copies equal to fitness^2
            Add these copies to temporary new_generation list
      Randomly select n individuals from new_generation and make this
      current_gen
Test and display robot following wall in test arena
```



Fitness with passing generation is shown above, where randomness is clearly visible due to a small population. The fitness noticeably falls to minimum from as high as 9, which is primarily depictive of mutation's damage, and usually occurs when the starting state is mapped to "do nothing" response.

The approach of solving the problem, modelling FSM and using GA, is quite novel to my knowledge and quite generic too – not map-specific unlike other easier less intelligent implementations. Computation time is primarily dependent on number of states (2^num_sensors) so it is an extremely efficient algorithm including for different larger maps. Survival in upcoming generations is dependent on fitness^2, and uncertain nonetheless even for the most and least fit, mimicking reality quite closely. The graphical representation helps visualize and adequately appreciate the solution.