***INFO6205 Final Project-Genetic Algorithms***

*Robot Controller*

*Group 525*

***Group Members:***

*Tse-Hsi Lin*

*Yu-Chiao Huang*

***Instructor:***

*Prof. Robin Hillyard*

***Implements***

*Problem Statement:*

The problem we are going to solve is designing a robotic controller that can use the sensors to walk through the maze and arrive the destination without crashing into the wall. The robot has six different direction sensors, three on the front, one on the left, one on the right and one on the back. The robot can take four actions: move one-step forward, turn left, turn right, or stay. Each route can be one solution and can be scored, we can find the fittest solution by comparing the scores we get. The purpose of this project is not to train a robot to solve the maze, our purpose is to automatically program a robot controller with six sensors, so the robot won't crash into the wall, and the maze in this case represent the complicated environment in the world.

*Phenotype:*

We dedicate Individual class as Phenotype, which contains one genotype.

*Genotype:*

In individual class (Phenotype), we have one chromosome and its type is int[] with

binary value 0 and 1. We call 0 and 1 value as gene.

*Fitness:*

The fitness score is a number that represents how good a solution to the problem this individual is. The fitness of each solution is calculated by the route. We initially set the route we want and calculate the route provided by chromosomes. In each route, we calculate whether this route passes through the route we set, and it will be scored.

We choose the highest fitness solution by tournament selection, which is the higher an individual’s fitness the greater chance that the individual will be chosen for crossover and mutate for the next generation. The final solution will be found until the number of generations reach the maximum we set.

*Initialization:*

The maze object we’ve created uses integers to represent different terrain types: 1 defines a wall; 2 is the starting position, 3 traces the best route through the maze, 4 is the goal position and 0 is an empty position that the robot can travel over but isn’t on the route to the goal.

This code contains a public method to get the start position, check a position’s value and score a route through the maze.

The scoreRoute method is the most significant method in the Maze class; it evaluates a route taken by the robot and returns a fitness score based on the number of correct tiles it stepped on. The score returned by this scoreRoute method is what we’ll use as the individual’s fitness score in the GeneticAlgorithm class’ calcFitness method.

*Expression:*

We define one chromosome as one solution. Each solution indicates a series of possible actions by binary code, such as 10011101110001001110100101101010000001001111001101010110010101111100011010110101001000010101011101000101101001110010110001100100.

Our robot will have four actions: do nothing, move forward one step, turn left and turn right. These can be represented in binary as:

* “00”: do nothing
* “01”: move forward
* “10”: turn left
* “11”: turn right

***Execution***

We use *IntelliJ* as our IDE for this project, and the framework we use is *Maven*, we have five different packages: *controller, helper, impl, repository and robot\_main*. We put the robot controller under *controller* package, put maze generator under *helper* package, put our Interfaces under *repository* package, put our implements for Interfaces under *impl* package and put our Main class under *robot\_main* package, we also have a maze folder to store our maze and a test package to store our tests.

**To execute our project**, you need to run the Main class under *robot\_main* package, then it will call the controller class to run our program, you also need to input the Maze name in the console, for example, input “*Maze1”*, we have 4 different Mazes, if you want to run the default Maze, you can just press *Enter*.

***Code***

***Experiment***

***Conclusion***