Program Structure and Algorithm

Final Project: Genetic Algorithm.
Author: Zheng Liu 001836884, Yang Li 001811572
Director: Robin Hillyard
Repo: *https://github.com/FlyFlyZheng/INFO6250_235.git*

Problem Statment

In a specific gridding map, there are some cups in fix number that are randomly scattered across the whole map. Meanwhile, a cleaner robot can take some actions (up, down, left, right, pick up, random move or do nothing) in a specific sequences. For each action, if a cup is picked successfully, the robot will get score accordingly. However, if some useless action happened, the robot will lose marks as well. Finally, the mission of cleaner robot is to clean those cups of entire environment in a specific sequence with highest score.

Rules for score:

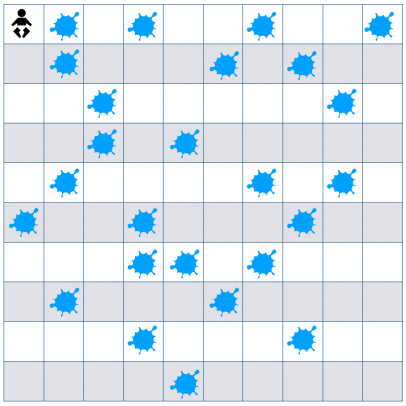
1. Clean a gird with a cup -> +10 points
2. Clean a cup without a cup -> -1 points
3. Hit the wall -> -5 points
4. Normal move -> 0 points

To resolve this problem, we propose an evolutionary approach based on Genetic Algorithm (GA) which consisted of several steps to get final solutions.

Implementation Design

Environment&Problem

Environment is just a map with fixed length and width. Before the Genetic Algorithm (GA) begin to evolve, a map along with cups with fix number will be generated randomly. Meanwhile, each time when the robot successfully pick up a cup, it will generated a cup in diagonal grid of that map.
Length and width of the map, and cup number can be defined in Config file.
A example of generated map is list below:



Genotype & Phenotype

Because there are only seven different kinds action that robot can perform for each step, we use seven integer to represent different states.

```
public static final int GO_UP = 0;
public static final int GO_DOWN = 1;
public static final int GO_LEFT = 2;
public static final int GO_RIGHT = 3;
public static final int GO_RANDOM = 4;
public static final int PICK = 5;
public static final int DO_NOTHING = 6;
```

In genetic algorithm, the gene is nothing but a set of reliable information which is a property of candidate solution. For this project, the genotype is just a collection of simple actions with a fix number for a robot. That is implemented as a fields below Robot class in an array of Integer.

```
Genotype: private int[] steps;
```

With the a specific order of motions, it is easily get the total score when of that clean robot accordingly, which is the phenotype of that genotype. A method named ‘calScore’ provides the mapping relationship between Genotype and Phenotype.

```
Phenotype: private int score;
```

Fitness

The fitness of one robot is a relevant performance crossing all populations in one generation. In detail, if we want to find the best solution, then it means we are going to find an individual with highest score. The fitness of one robot is *f(Robot1)/f(Robot1) + f(Robot2) + …… f(RobotN).* f() is a method to calculate the total score a a particular path. T*his kind of representation will give us the relevant performance of each robot.* And we can use this fitness method to filter out those path with low score and select better parent for each generation.

Selection

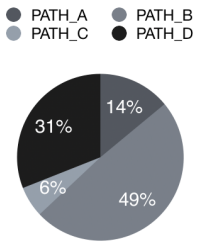
For selection implementation, we use **Roulette Wheel Selection** to randomly selected parent of next generation. We chose roulette wheel selection is because even though the probability that the weaker solutions will survive is low, it is not zero which means it is still possible they will survive. Based on this kind of selection, there is a chance that even weak solutions may have some features or characteristics which could prove useful following the recombination process.

Roulette Wheel Selection is used to implement the as selection mechanism in this project, which is very helpful to select potentially useful solution for recombination in Genetic algorithm. As we mentioned before, the fitness function has already assigned a fitness to each possible solution path. Accordingly, this fitness level can be used to associate a probability of selection with each individual. For example, if f(i) is the fitness level of individual I of that population, the probability of being selected is:

P = f(i) / f(1) + f(2) +……+ f(N). N is the number of each population

During this each selection process, a random number will be generated from [0, 1]. From first robot, if the accumulated probability larger than the generated random number, it will be selected.

Take population number=4 as example



Suppose we generated 4 random number from [0, 1] as:
R1 = -0.450126 R2 = -0.110347
R3 = 0.572496 R4 = 0.98503
The selected time for the following for paths will be 1, 2, 0, 1

	Fitness score	Probability of being selected	Accumulated probability	Selected tims
PATH_A	169	0.14	0.14	1
PATH_B	575	0.49	0.63	2
PATH_C	64	0.06	0.69	0
PATH_D	361	0.31	1	1

Crossover

After selecting two parents, we randomly choose the starting point of point of crossover and the ending point of crossover, and then perform a swap operation between start point and end point.

Parent1

2 3 1 5 2 4 3 1 2 5 4 6 1 6 3 2 2 … … … 5 5 3 4 2 6 6 1 2 4 3 2 5 1 5 3

Parent2

1 3 1 4 2 3 4 1 1 5 3 6 4 5 1 2 3 … … … 2 2 1 2 1 2 1 3 6 4 5 4 5 1 5 3

Child1

2 3 1 5 2 4 3 1 2 5 4 6 1 5 1 2 3 … … … 2 2 1 4 2 6 6 1 2 4 3 2 5 1 5 3

Child2

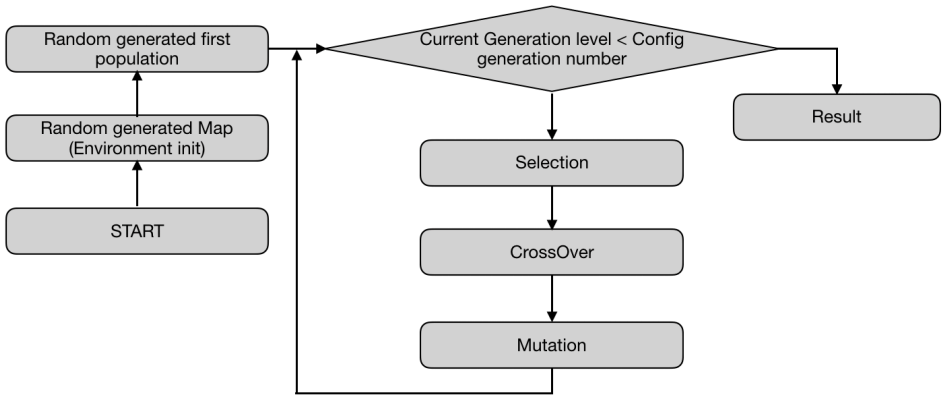
1 3 1 4 2 3 4 1 1 5 3 6 4 6 3 2 2 … … … 5 5 3 2 1 2 1 3 6 4 5 4 5 1 5 3

Mutation

For each step sequence of one generation, a random number between [0, 1] will be generated. If this number less than variable percentage (set in config file), mutation operation will perform on this entity. The the position of mutation is randomly selected.

Evaluation

A single evolution includes process of selection, crossover and mutation. The flow of whole evaluation process is like below:

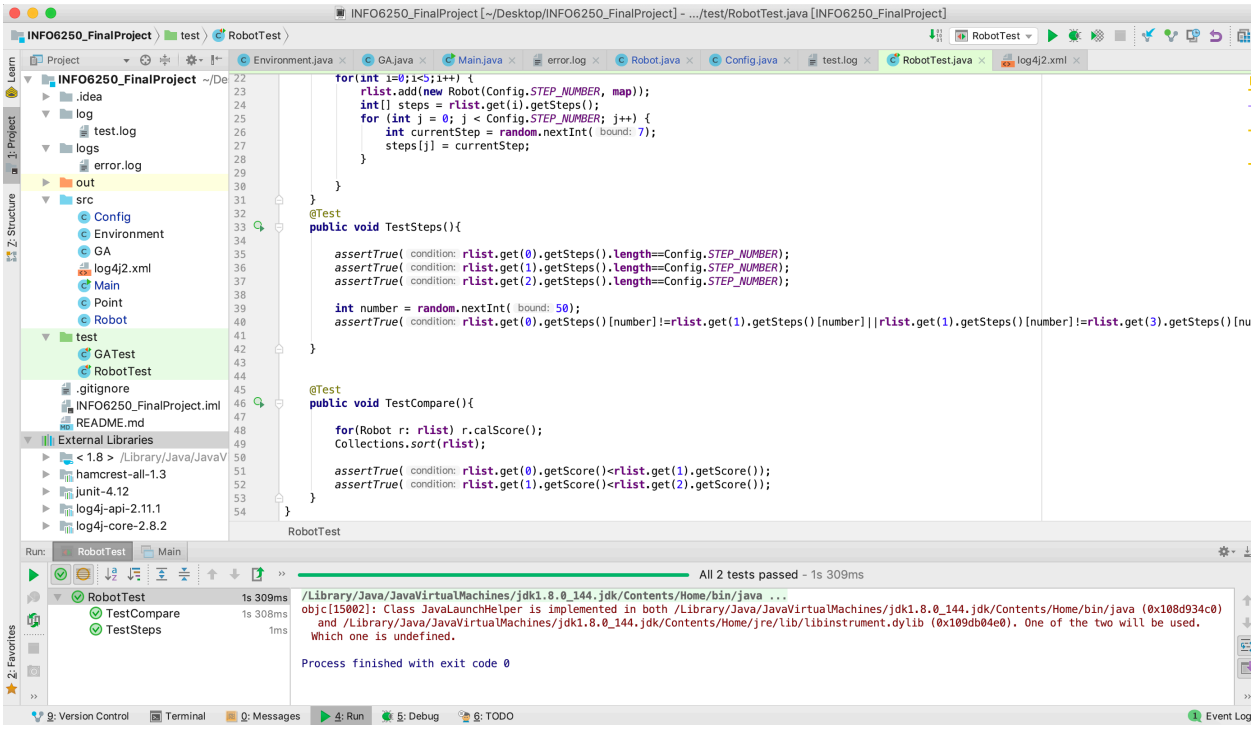


Result

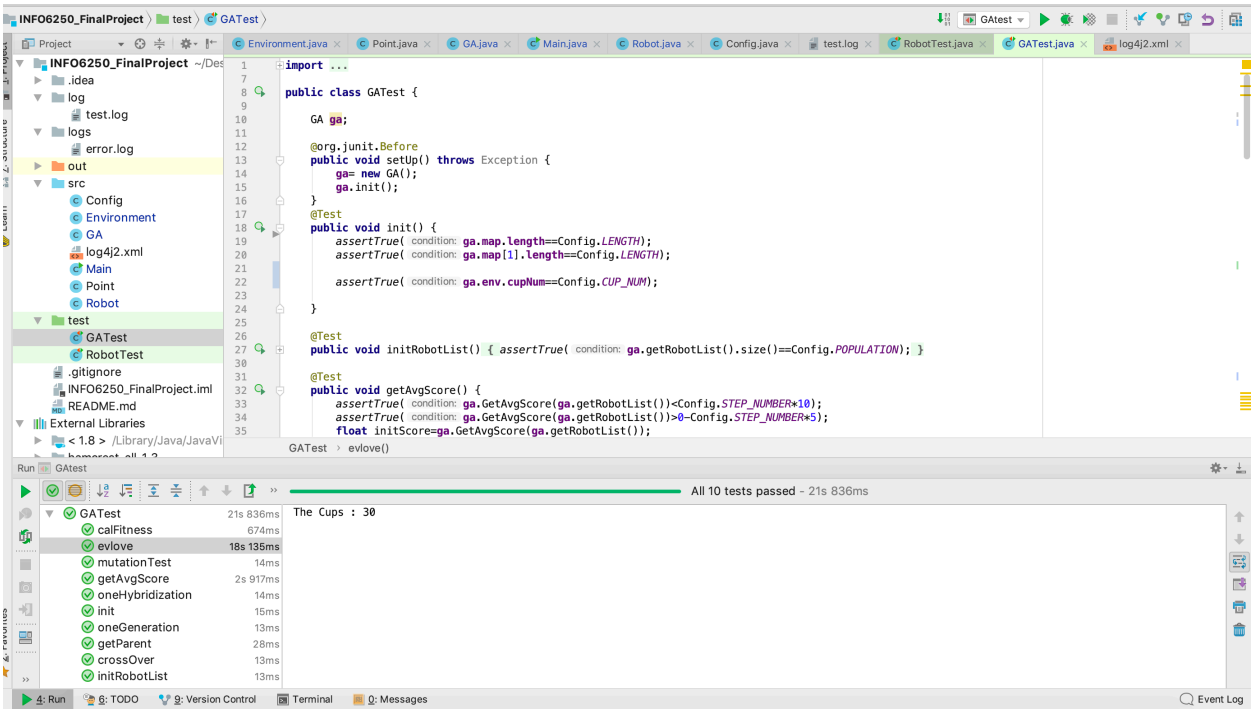
All Unit tests pass

Our project builds and passes all unit test

1. passed all unit tests for robot method

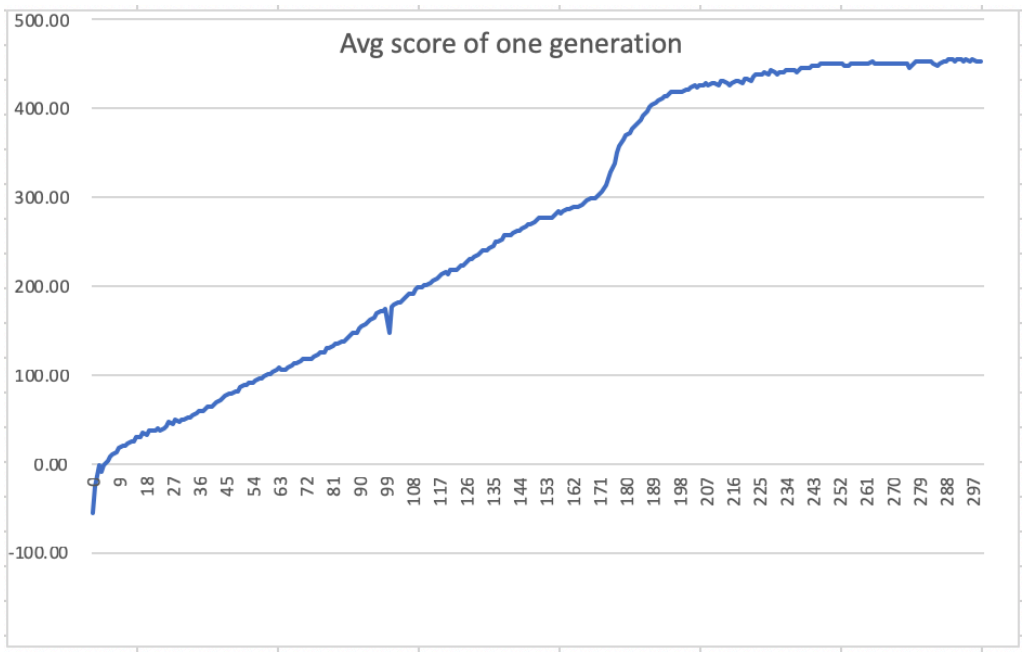


2. Passed all unit tests for fundamental evaluation method



Evaluation Result

We have initialize map with 10*10, and randomly put 50 cups in that map. Robot willl start it's clean at (0,0) and the step number is 200. As a result, The average score of the first 300 generation is showed in Figure1. After several times generation, the score is close to 500.



After 10000 generation, the best track is as follow with best score 480. Detailed in formation can be found in log file at the top level of repo.

3->5->2->5->0->5->3->5->6->3->1->5->6->0->5->1->3->6->3->6->2->0->5->6->3->6->2->0->0->1->0->5->3->2->6->1->5->3->6->3->6->0->2->1->3->0->5->3->6->0->5->6->0->0->5->3->0->5->2->5->0->0->5->3->1->0->5->1->5->3->6->0->1->0->5->3->5->2->3->1->6->5->0->1->2->5->2->2->5->1->2->5->2->5->1->2->1->1->2->6->1->2->1->6->1->3->5->2->3->6->6->1->6->5->6->3->6->5->0->2->0->3->0->2->1->6->5->3->5->0->3->0->0->0->6->5->2->5->0->5->2->1->6->6->1->5->3->5->3->6->1->0->6->3->3->6->6->6->1->3->5->2->1->5->1->6->5->2->6->5->2->5->0->6->3->5->6->0->5->0->2->5->2->6->0->3->5->6->3->3->0->5->0->3->5->0->5->3->1->2->