



















# INFO6205 Final Project Report

## Problem Description

A robot named "Robby" who lives in a (computer simulated, but messy) two-dimensional world that is strewn with empty soda cans. Robby's job is to clean up his world by collecting the empty soda cans. Robby's world, consists of 100 squares (sites) laid out in a 10×10 grid, and there is a wall around the boundary of the entire grid. Various sites have been littered with soda cans (but with no more than one can per site). From wherever he is, he can see the contents of one adjacent site in the north, south, east, and west directions, as well as the contents of the site he occupies. A site can be empty, contain a can, or be a wall.

For each cleaning session, Robby can perform exactly 100 operations. Each operation

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5										
6										
7										
8										
9										

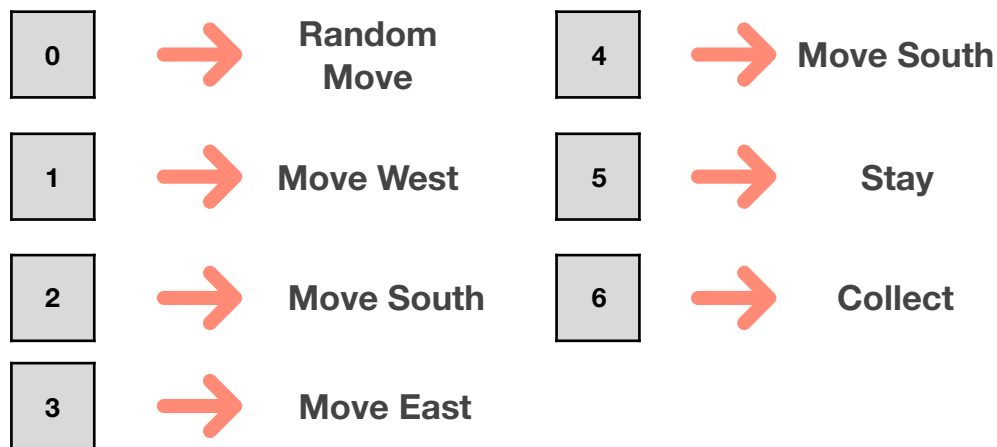
consists of one of the following seven choices: move to the north, move to the south, move to the east, move to the west, choose a random direction to move in, stay put, or bend down to pick up a can.

# Genetic Algorithm

## 1.Genotype

Because the robot can scan 5 sites around it. And every site have 3 situations, there are totally  $243(3_{\text{situations}}^{5_{\text{sites}}})$  different possible situations. So the length of genotype can be 243, and for each gene in genotype, it can be 0 to 6(corresponding to 7 operations).

0	1	2	3	4	5	6		242	243
3	1	0	6	2	3	2	.....	6	2



## 2.Fitness calculation

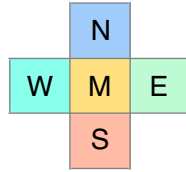
Each operation may generate a reward or a punishment. If Robby is in the same site as a can and picks it up, he gets a reward of 10 points. However, if he bends down to pick up a can in a site where there is no can, he is fined 1 point. If he crashes into a wall, he is fined 5 points and bounces back into the current site. Clearly, Robby's reward is maximized when he picks up as many cans as possible, without crashing into any walls or bending down to pick up a can if no can is there. After 100 operations, Robby can get his score which is the fitness of this problem.

## 3.Phenotype

The genotype of this problem is how Robby finish his 100 operations according to his genotype.

## 4.Expression(need a graph)

For a given map and start position, a unique genotype defines a unique phenotype. That's say, the position of a gene means that Robby is in the situation of this gene position and the value of gene means what operation strategy he will take. And in this Figure,Robby's position is 68.

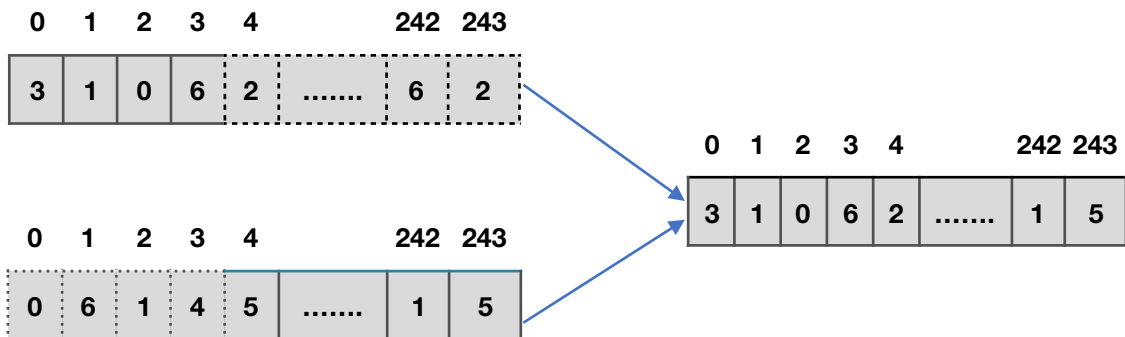


M	W	S	E	N
0	2	1	1	2

$$3^4 \cdot 0 + 3^3 \cdot 2 + 3^2 \cdot 1 + 3^1 \cdot 0 + 3^0 \cdot 2 = 68$$

### Evolution Process

1. Firstly, initialize the size of the map as 10x10, and each grid has 50% possibility having a can. Then randomly initialize the start position of Robby in the map. Thirdly, set the population of each generation equal to 200 and generate 243 genes randomly from 0 to 6 for the first generation.



2. Initialize the first generation.  
According to his genotype, each Robby in the population choose the operation after knowing which situation he is in. After 100 operations, calculate the score they get. Then using Roulette Wheel Selection Algorithm to calculate the rate for each Robby, the larger rate they have, the larger possibility they have to reproduce. During reproduction, using crossover to generate new generation. The process of crossover generates a new population which is twice larger as current population, calculation their fitness to choose the best half of the population.  
For each generation, every gene of each individual has the same possibility to mutate to a random new gene, set possibility of mutation as 0.5%.
3. New generation replace the old generation, then does Process 2.
4. Looping Progress 3 until reach the maximum generation number.

# Conclusion and analysis

This is the best Gentye of final generation

042043511223243110406346261113142316111230145102410111335006166404114060  
43130342266663646166266010266166411366663615616602400256263534066626303  
06634226202465602446505123561625222324022122520250142365050244201216120  
5412554340530031456510450056;

## 1.strategy analysis

We find there are 23 “5”s in the best Gentye. It means in these 23 situations, the robot will stay in the grid. When we initialize the original Genetic. There are 243 situation. 16.7% of each operation is likely to occur in the genetic sequence. This means that there are about 40 of each operation in the initial Genetic Sequence.

Decimal	7	45	57	116	127	132	157	164	166	170	175
Ternary	00021	01200	02010	11022	11201	11220	12211	20002	20011	20022	20111
Decimal	188	192	199	201	215	219	220	225	232	234	238
Ternary	20222	21010	21101	21110	21222	22010	22011	22100	22121	22200	22211

We find after 1500 generation of hybridization, there are 23 ‘5’s in the genetic. It means in these 23 situation, the robot will stay in the grid. And the first bit represents the middle position. And 2XXXX is not exist this situation. So only 7 situation the robot will not move. It proves our training is work.

## 2.Result and analysis

This is our original condition. We want to explore this GA best performance. At the beginning, we assume the population larger, the score higher. And after much generations, the best score will continues to rise. So we initialize the population at the up bound 1000.

```
public static int generationNumber = 2500;
public static int ROW = 10;
public static int COL = 10;
public static double CAN = 0.5;
public static int POPULATION = 1000;
public static int operation = 100;
public static double MUTATIONRATE = 0.005;
```

And after 2500 generations, we print the best score in each generation. So we find that before the 500 generation. The best score has increased linearly. And after 500 generation, the best score increase very slow. After 1000 generation, the best score will not change much more. It just swing near the 400 score.

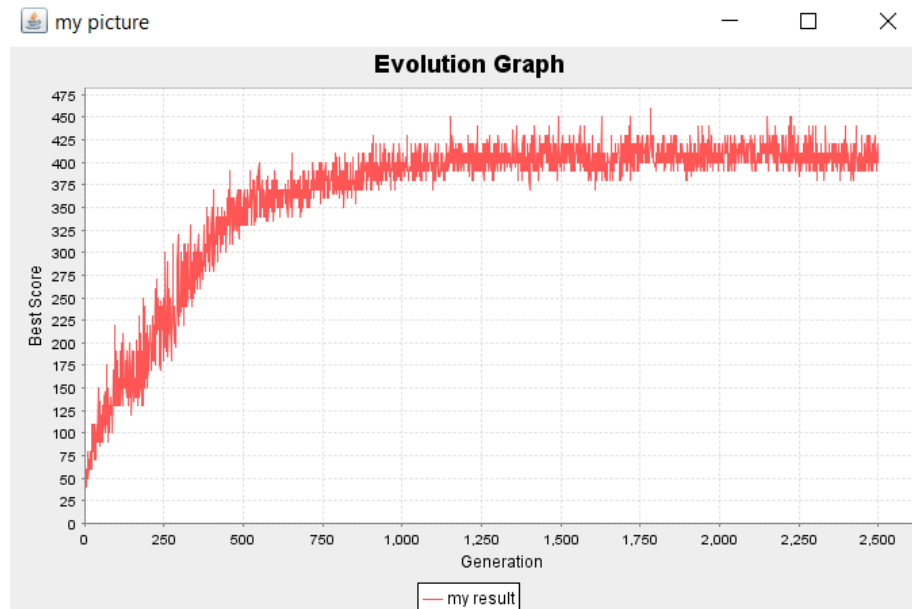


Fig.1 population 1000

So we test the population 800, 600, 400.

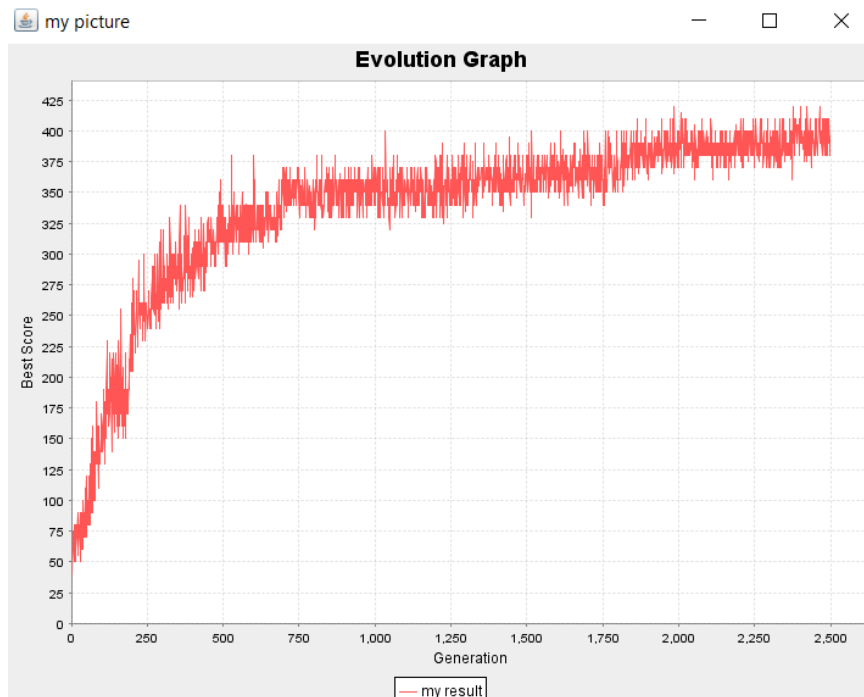


Fig.2 population 800

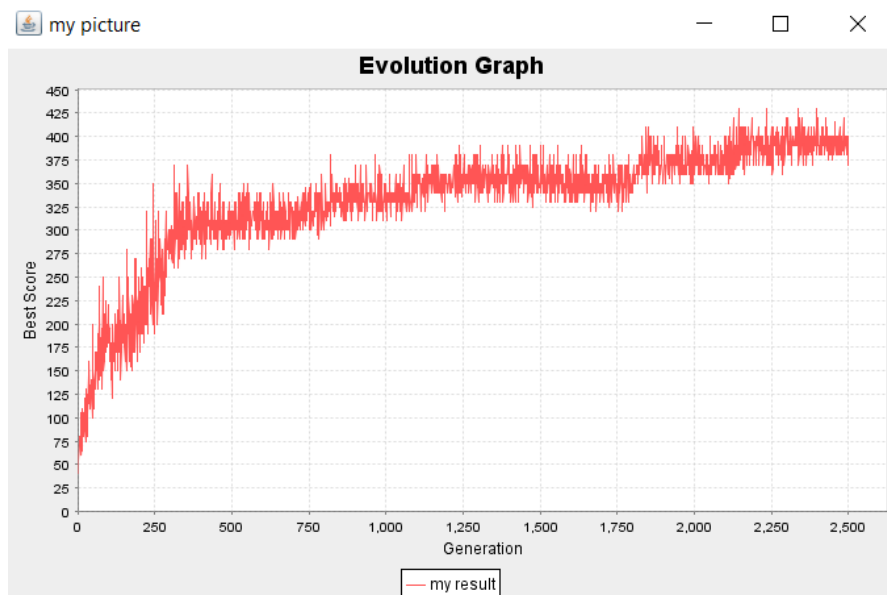


Fig.3 population 600

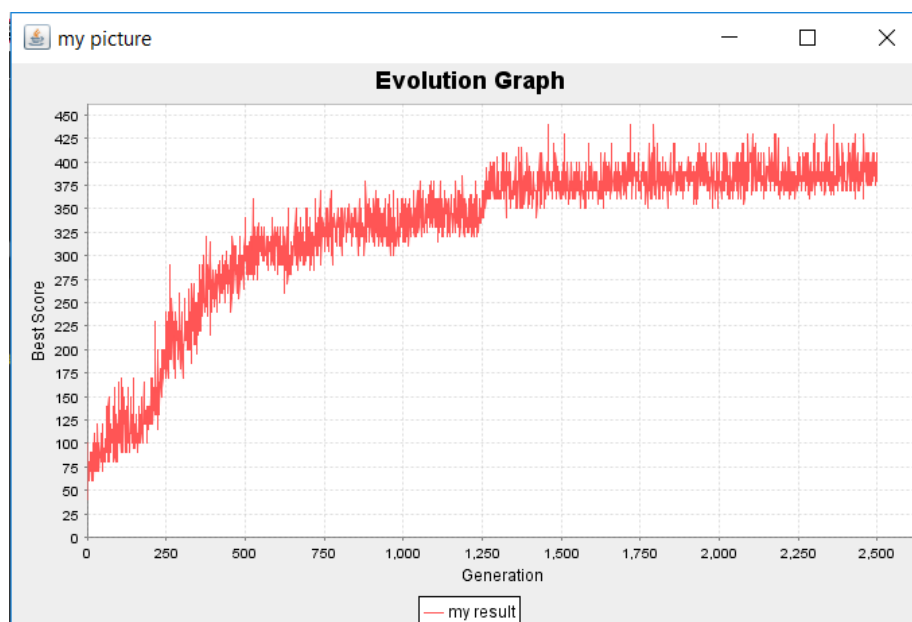


Fig.4 population 400

We find in these situations, the best scores of the first 500 generations grew very fast. But bigger population very get higher score at 500th generation. And finally the best score will have higher low swing point.

Population	1000	800	600	400
Low swing point	410	390	375	360

So I think the population bigger, the best score will location at the higher low swing point. At the end, the value of low swing point will approach 500 score. Then I test 10000 population.

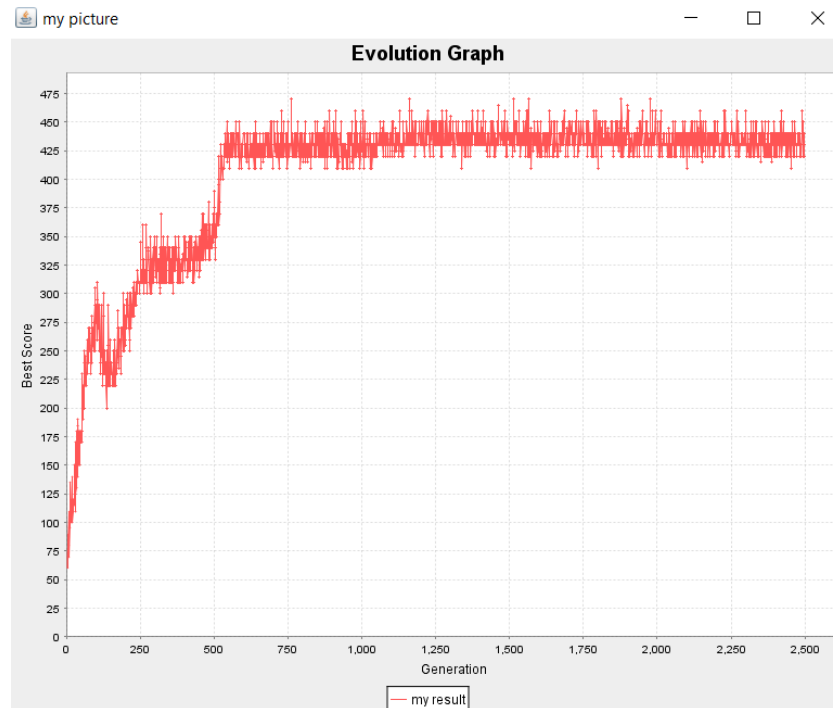
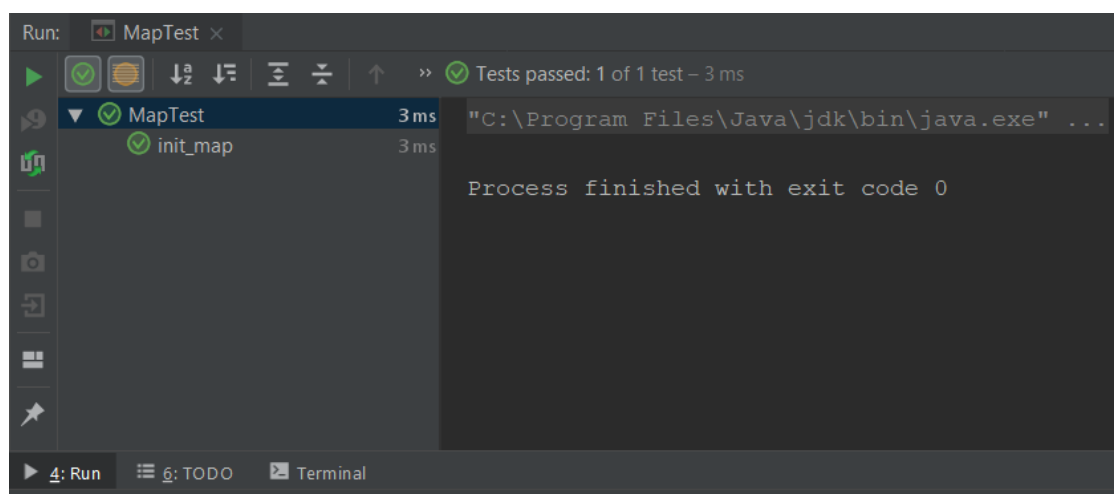


Fig.5. population10000

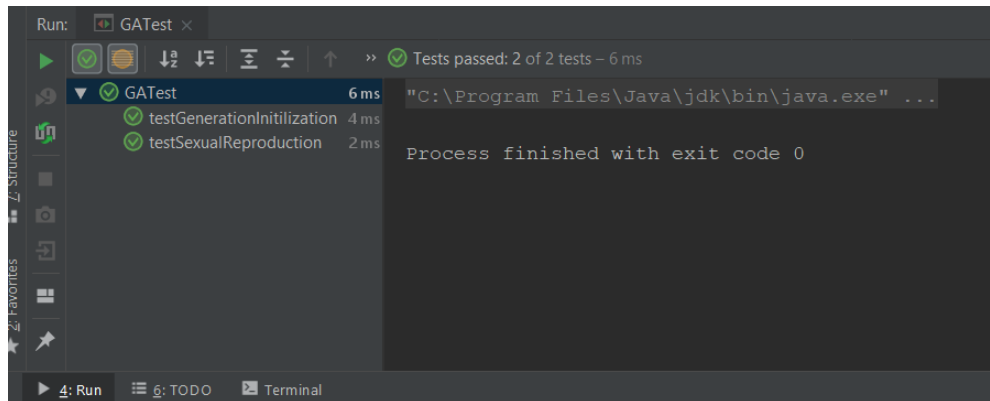
10times population, the value of low swing point at 425. Only increasing 3.6%.

### 3.Test result

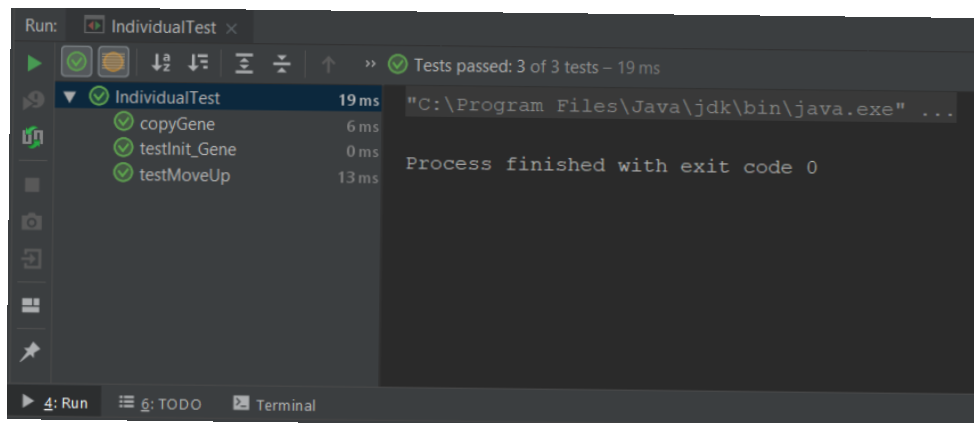
#### 1.MapTest



## 2.GATest



## 3.IndividualTest



## Conclusions

Actually , we also calculate the average score. It has not obversely different Value between each different population situation at the final generation. As you see ,when I fixed population, only changing generations will not change any thing. The population control the final result.But the generation