

Trash Picker Robot Robby

Genetic Algorithm Problem in Melanie Mitchell's book Complexity



Ruohua Yin & Wanxin Chen

Group 513

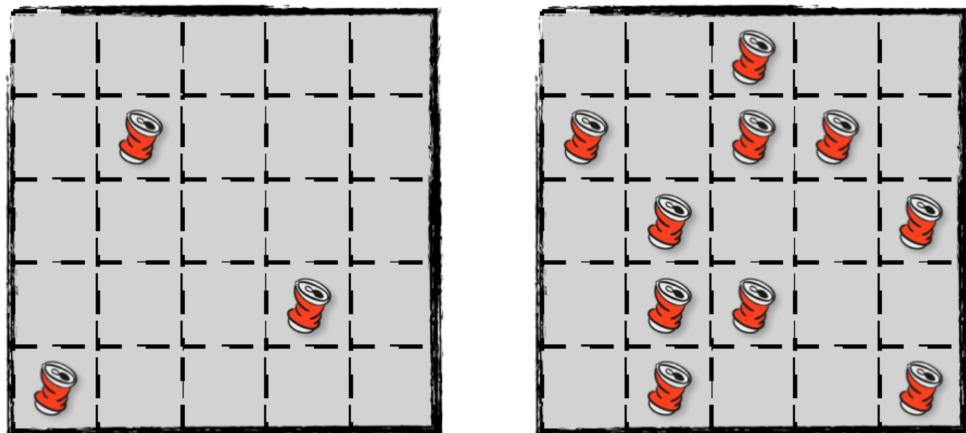
2018 Spring

INFO6205 Final Project Report

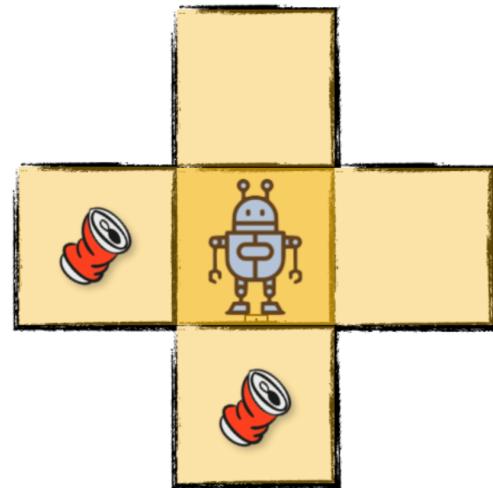
Problem Description

Robby is a self-directed robot tasked to pickup cans littered around a $M \times N$ board. Before the Robby can get into work, we definitely need to make rules for the work pattern he has, also the environment he is going to work on.

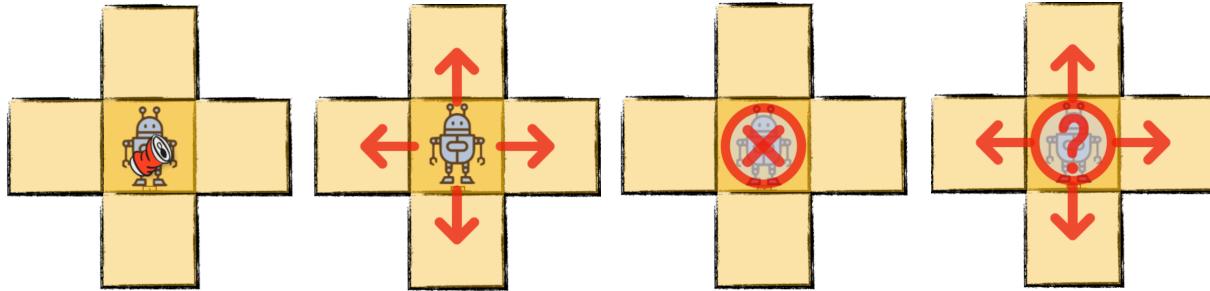
As mentioned before the environment we have here is a $M \times N$ board. The grids belong to it have a possibility having a trash awaited to be picked up.



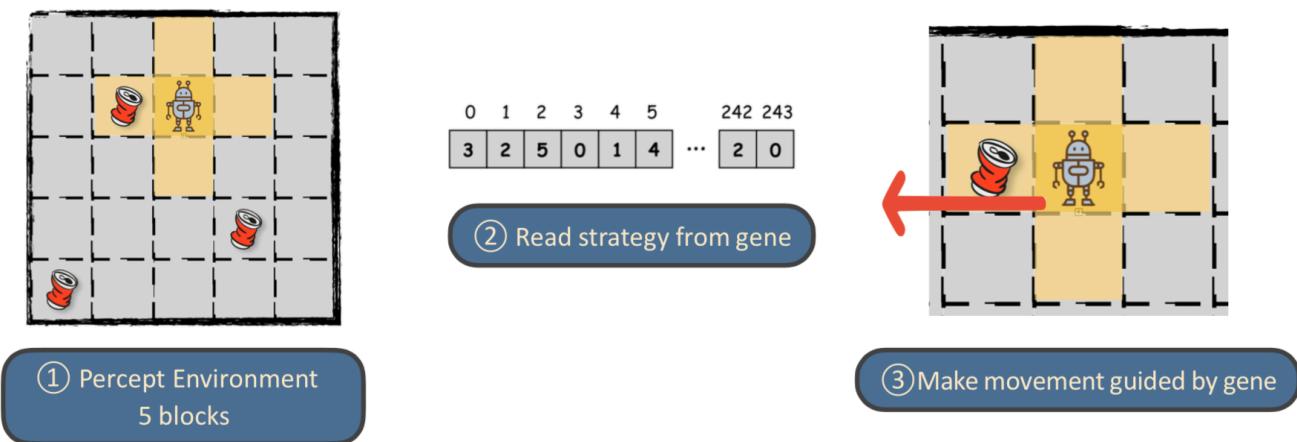
However, robot Robby is not clever and capable enough with his limited perception over 5 surrounding grid world - north, south, west, east and his original spot.



He has 7 different behaviors which are collect trash, move north, south, west, east, stay in the grid, and a random move respectively.



The working pattern is based on the different scenarios, which is written in each Robby's gene. Each time before he makes a single movement, he will percept the status of his surroundings, then read the gene inside him. Robby will follow the gene to make actions completely.



About how genotype (the order and content of genes) and phenotype (strategies which guide its movement) are designed will be introduced in details later.

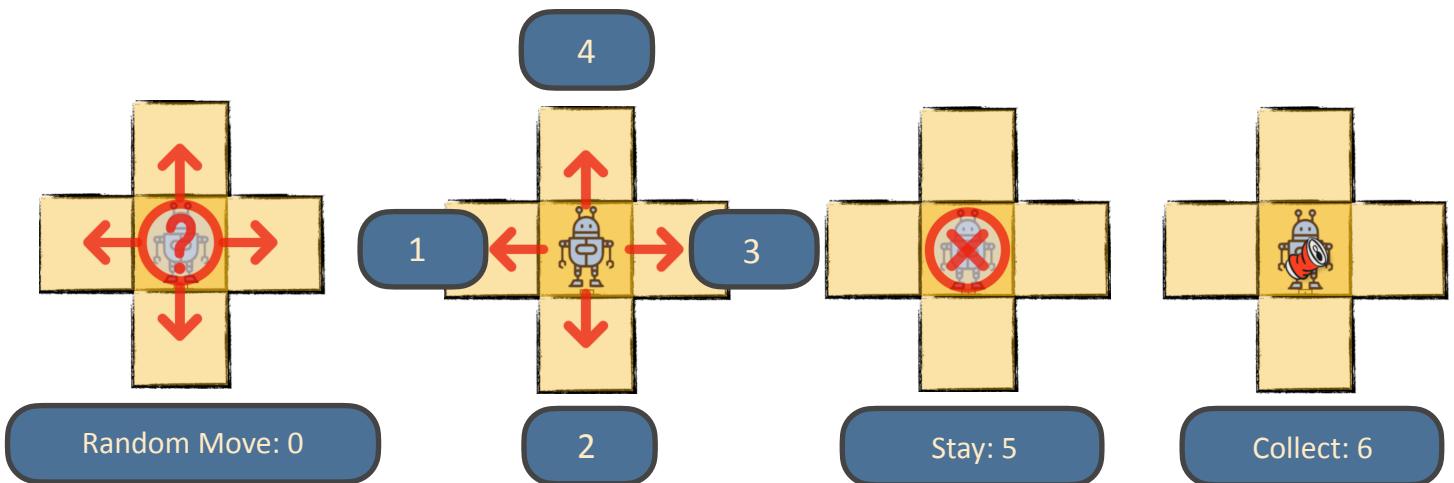
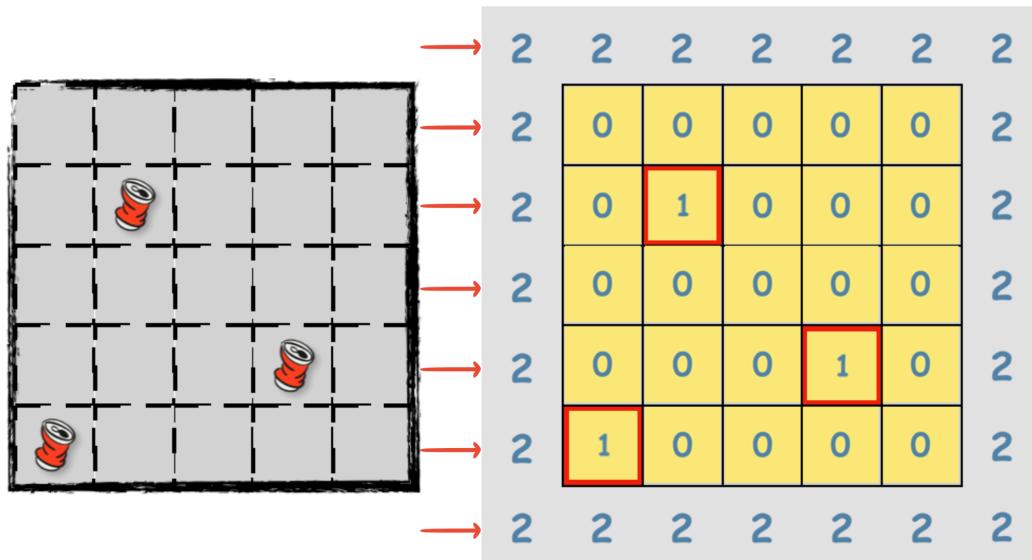
Model Detail Design

Environment Design:

There are 3 kinds of situations for a grid which are

- ① Grid with a trash ② Empty grid ③ Wall.

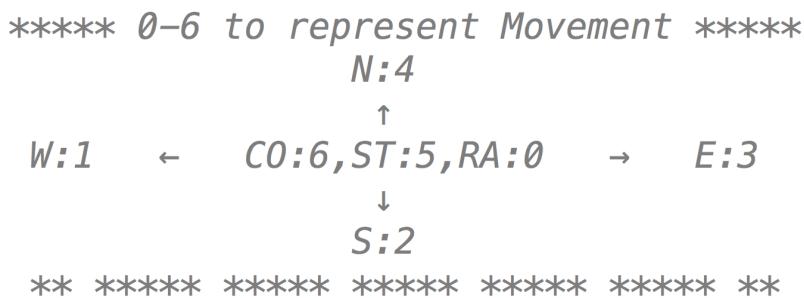
The third situation is going to be analyzed at first. The map is represented by a matrix with 3 different number inside(0 for empty, 1 for a trash, 2 for wall), but a map with $M \times N$ grids should be represented in a $(M+2) * (N+2)$ matrix. The extra 2 for M and N is to represent wall on the 4 edges.



Robot Design:

A Robot has 7 different kinds of movement:

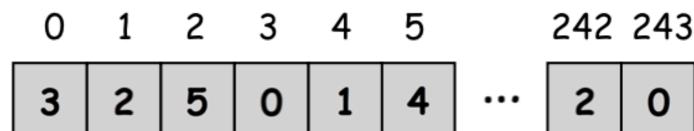
In the implement process in Java, we use the number from 0-6 to represent different movements.



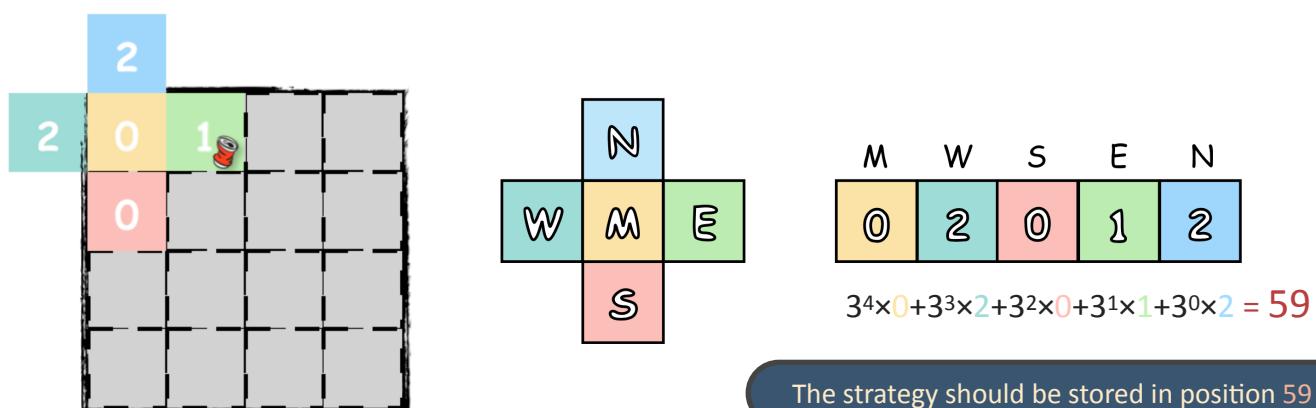
Gene Design:

1. Gene order:

As the gene inside a robot should be able to store the strategies for 243(= 3^5) situations (5 blocks view with 3 different stuff insides). So 243 gene positions should be included for each individual.

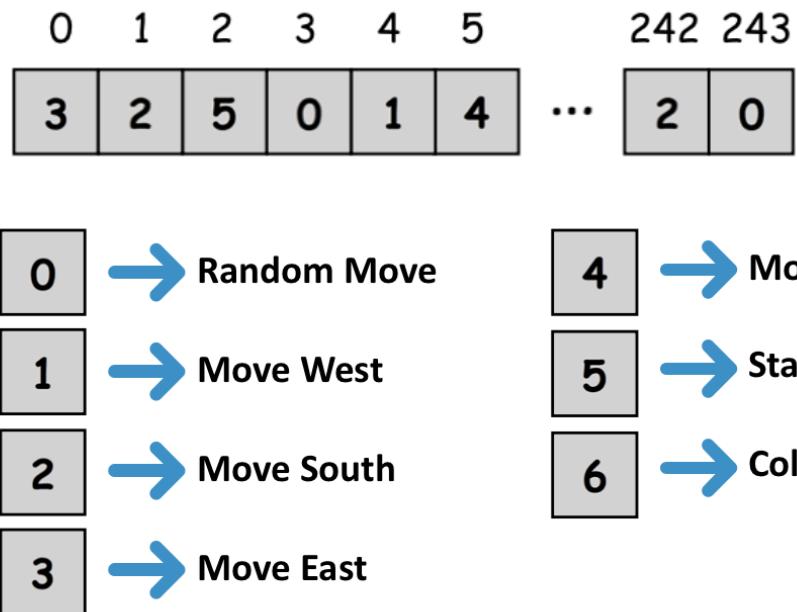


The way we decide the position of distinct strategies is a Ternary numeral system.



Note that there are some gene positions don't make sense. Like a roby cannot stay on the wall, hence the middle block represented by 2 won't come out for a robot, but it doesn't matter because even for human, there are some genes that have nothing to do with the environment. These kinds of genes will do mutation and crossover following the reproduction rules, while actually the environment won't select them at all.

2. Genotype and Phenotype Design:



Fitness Function Design:

- Pick up trash successfully → +10 scores
- Pick up trash in an empty grid → -2 scores
- Crash on the wall and bounced back → -5 scores
- Move or Stay → Nothing changes

Evolution Process

Initializations:

1. Initialization of individual:

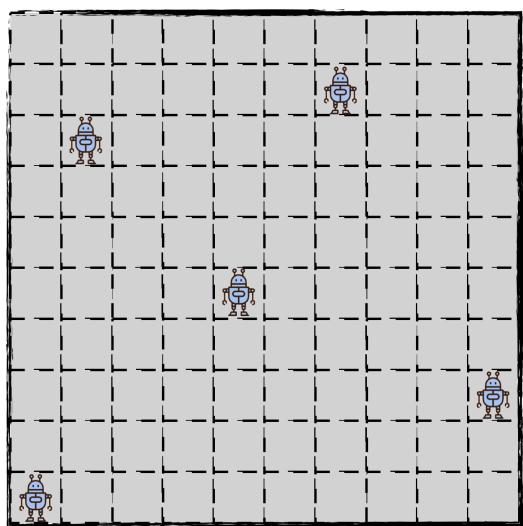
We set up the population of each generation, which is 200. The 243 genes for each individual are inputted randomly from 0 - 6.

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

2. Initialization of the map:

A map is the environment for a individual to gain scores. We initialize the environment to be a 10×10 board, each grid has the 50% possibility for having a trash. Before each individual gets into the game, the map will be initialized.

3. Initialization of the position:

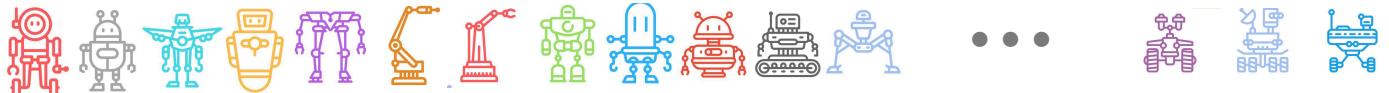


Robby will get a position in the board with random X and Y axes.

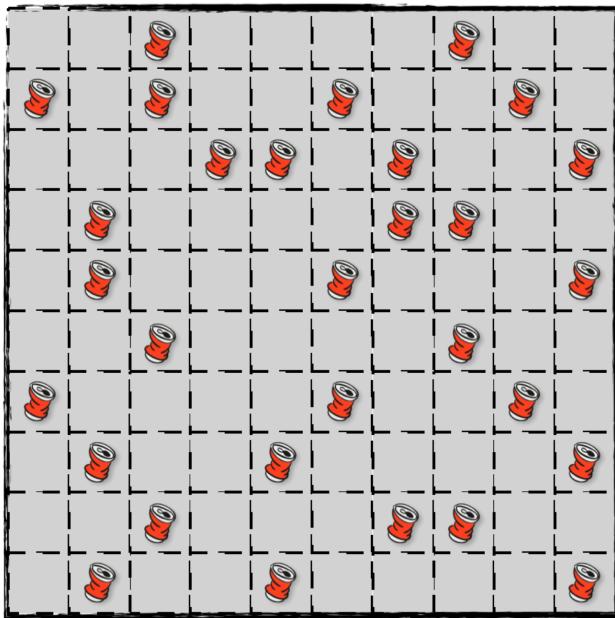
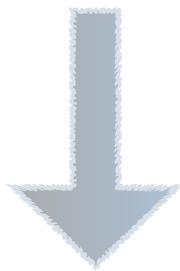
You may have questions here if we initialize the map and each individual's position, it may force the individuals to play different games.

But what we can tell it is that it doesn't matter for the individuals to have different starting point or play on different maps. For the reason that this is only an evolving process, not a real match. The grid in the board have the same possibility to have a trash. The

distinct starting point and map may let them face a different situation at the beginning of each game, which means as different select direction. If each Robby start at $(0, 0)$, the situation could be too similar. It may raise the selection speed on the gene on particular position, which is unwilling to be seen.



First Generation Initialization

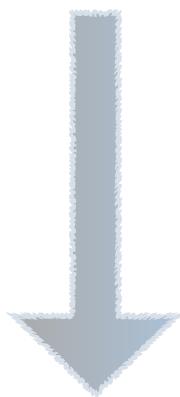


Play a Game of Trash Picking

Each individual in a generation is put into the map then play the trash picking following their strategy guided by their gene. The scores they gain are calculated by the fitness function we designed.

Fewer they crushed on the wall, made pick action in an empty grid and more correct pick on the trash, they will definitely perform better in the game.

Rank the Scores



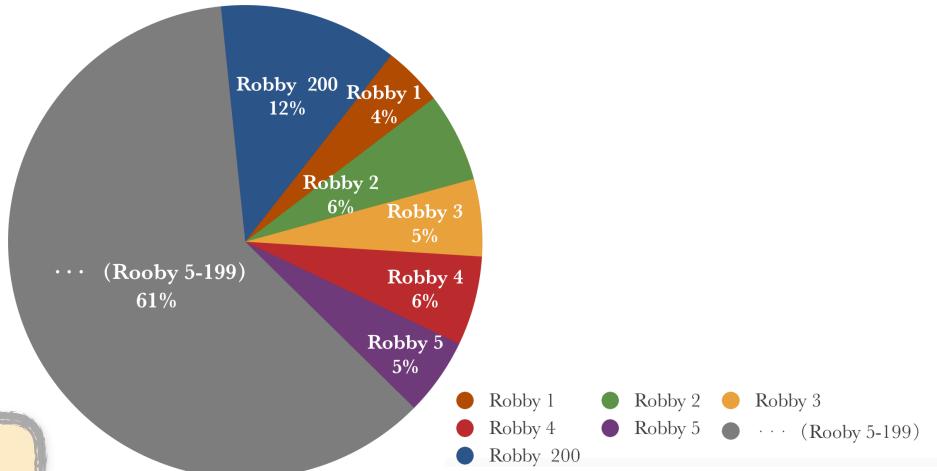
The individual with a higher score (which is calculated by the fitness function) has a larger possibility to be selected as a parent, which means a bigger chance to propagate its genes.

Selection → Roulette

Selection → Roulette

We choose the parents of the next generation by playing Roulette game. The ratio of area they have in the circle is decide by the score percentage among total points of the generation.

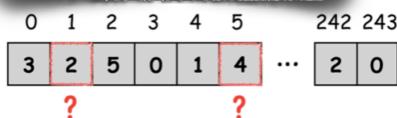
Roulette Game to pick a Parent



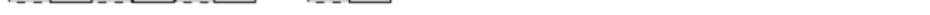
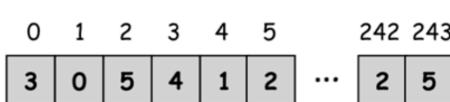
Reproduction

The reproduction pattern we designed for the robot is they have both asexual and sexual ability. Higher the sexual reproduction rate, faster the evolution process the generation has.

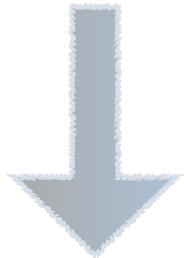
Asexual 10%



Sexual 90%

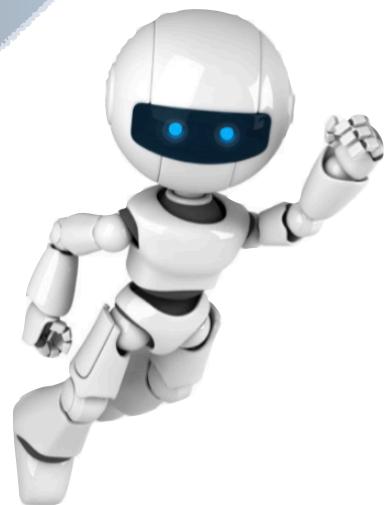
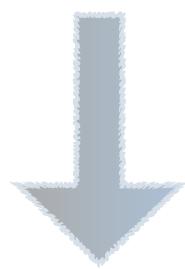
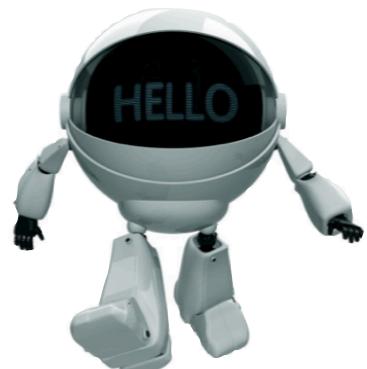
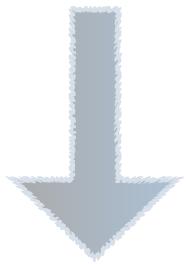


Mutation



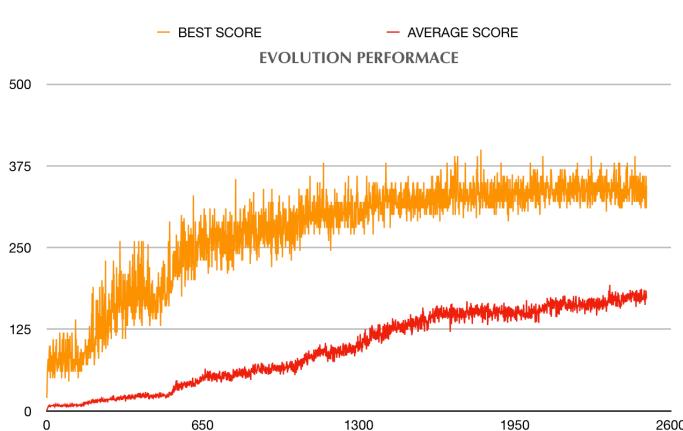
Before the new individual is being made, each gene out of 243 has a 0.05% possibility mutating to a new genetic code from 0-6.

The New Generation



Results & Conclusions

It can be predicted that the first generation could perform really bad because they have such a small amount of good genes generated by list of random number. Hence much of them couldn't make a right pick up action even if they have a trash in their current grid. And So much of them ended their game by make a crush action to the wall continuously because so long as the situation in the 5 block doesn't change, they will make the same action by reading the same gene. However, some of the individuals can make some scores although their genes are composed by random numbers. Higher the score they have, higher the chance they get to propagate their genes although their genes are still not good enough as initial generations. The trend is in the direction of a increasing performance on the game, their genes which hold a good strategy will transfer to their child increasingly.



The left diagram is the result came out from one of the evolving process. Both the best and average performance in every generations continuously made progress and converge to where theocratically the best point. But what is the reason why they didn't converge to the exact point which is the best performance.

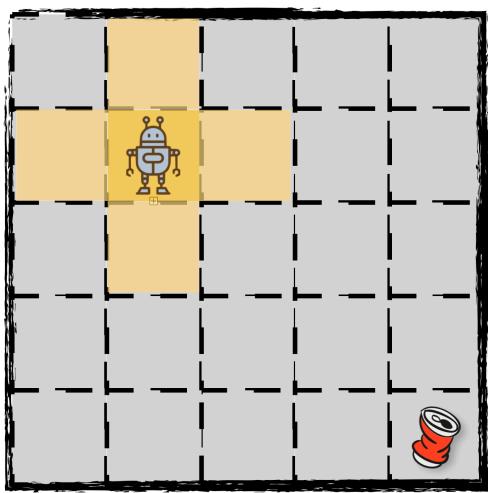
① Mutation Rate and Mortality rate

```
public static int EVOLUTIONGENERATION = 2500;
public static int ROW = 10;
public static int COL = 10;
public static double POSSIBILITY = 0.5;
public static int POPULATION = 200;
public static int MOVETIMES = 100;
public static double ASEXUALRATE = 0.1;
public static double MUTATIONRATE = 0.0005;
```

The former screenshot is the configuration of parameters for the evolution process. Although the mutation rate for each gene is only 0.05%, there are 243 genes for each of 200 individuals in 1 generation, so there will always have a group of genes change from the old to new one. The bad performance genes have chance to become a good one, in the mean time, a selected good gene have chance to become a bad one randomly.

And in the evolution pattern we designed, even the individual have a low score calculated from the fitness function, they still have the chance to be selected as a parent. So eliminate all individuals with a bad gene could be a much longer process than 2500.

② It's limited by the nature of the robot view



Another reason for the gap between is that Robby only has a view for 5 blocks. If you take a look at the situation which is shown left, you got to understand. In such situations, a robot with a good gene would move to a random direction. But a random move is obviously not a good news for him to pick up a trash out of his sight. If the map is large enough, no matter how good the gene inside him is, nothing could guarantee Robby to find a trash which is distant to him.

③ The Performance here is trash picking score, not gene itself.

What theoretically the best is the robot made movement like, he can pick up a trash every time he makes a move to a direction. Every 2 actions made 10 scores. So 100 action can make 500 grades. But it's not a guarantee that the grids with a trash is linked with each other.

During the evolving process, the strategy stored in their gene will become better and better, but the map is different and generated randomly.

So, we try to extract some of the gene on typical position of the individuals with the best performance in the final generation. As is printed below, no more than 20 individual's first 60-70 genes there are. We can conclude that different individuals have most of their gene similar after 2500 generation evolving. We can collect some critical gene and use statistic method to see how good they are being trained.

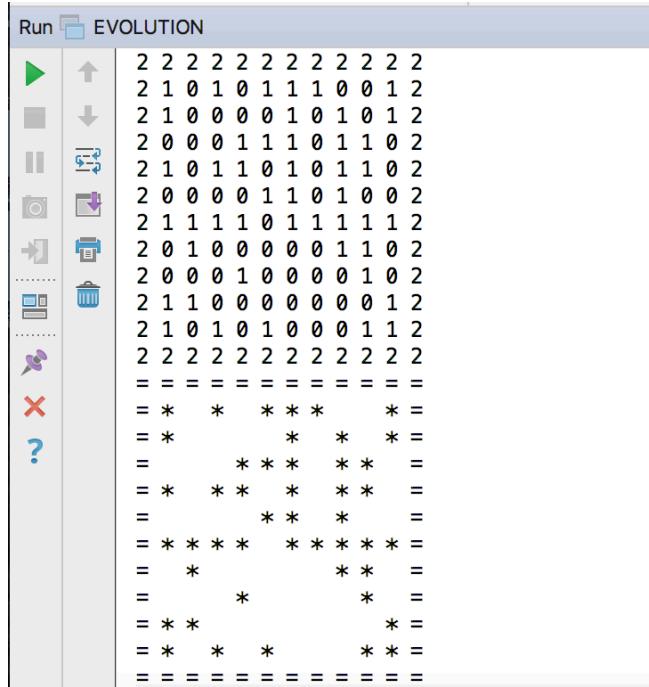
Best Gene:

Situation					
Gene Position	0	81	3	59	30
Description	5 blocks empty	A trash in the middle	A trash in the right grid	On the left and up most corner, a trash in the right grid	1 trash in the right grid, 1 trash in the left grid
0 Random Move	99.5%	0%	0%	99.5%	0%
1 ← Move Left	0.05%	0%	0%	0.5%	99.5%
2 ↓ Move Down	0%	0%	0%	0%	0%
3 → Move Right	0%	0%	100%	0%	0.5%
4 ↑ Move Up	0%	0%	0%	0%	0%
5 Stay	0%	0%	0%	0%	0%
6 Collect	0%	100%	0%	0%	0%

Evidence of Running

Program Outputs:

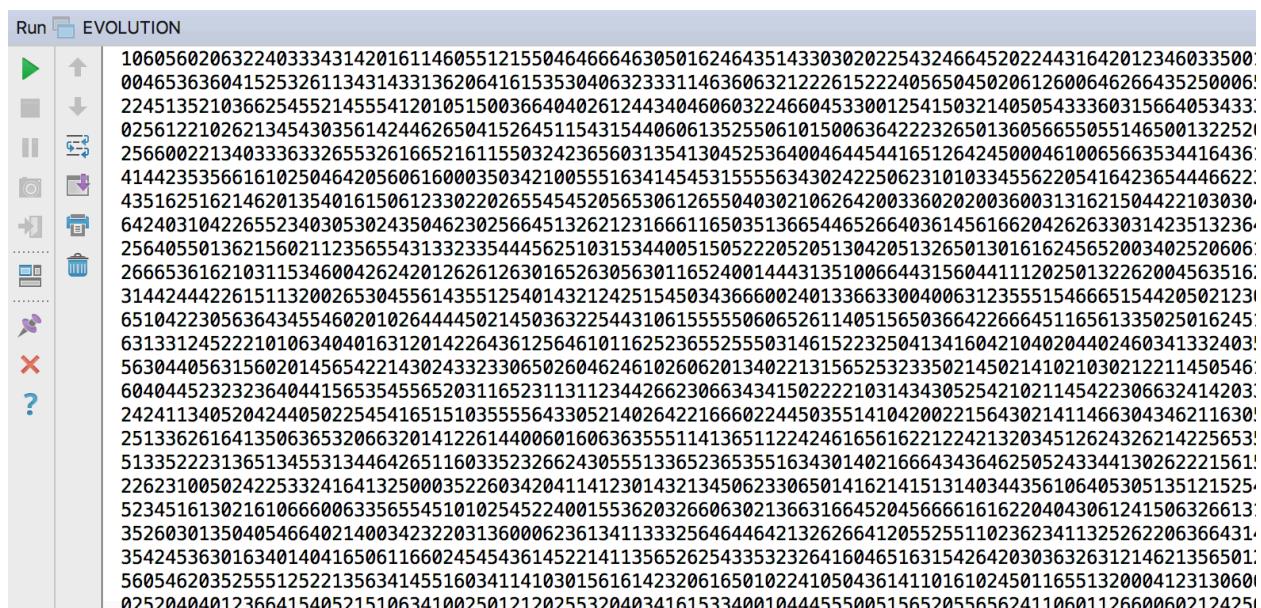
1. Map Initialization



The screenshot shows the Evolution software interface with the title bar "Run EVOLUTION". The left sidebar contains various icons for file operations like run, stop, save, and delete. The main area displays a grid of binary values (0s and 1s) representing the initial state of the map. The grid consists of 22 columns and 18 rows. Below the grid, there are several lines of asterisks (*) and equals signs (=), which likely represent boundary conditions or specific marker positions.

2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	1	0	1	0	1	1	1	1	0	0	1	2										
2	1	0	0	0	0	1	0	1	0	1	2											
2	0	0	0	1	1	1	0	1	1	0	2											
2	1	0	1	1	0	1	0	1	1	0	2											
2	0	0	0	0	1	1	0	1	0	0	2											
2	1	1	1	1	0	1	1	1	1	1	2											
2	0	1	0	0	0	0	0	1	1	0	2											
2	0	0	0	1	0	0	0	0	1	0	2											
2	1	1	0	0	0	0	0	0	0	1	2											
2	1	0	1	0	1	0	0	0	1	1	2											
2	2	2	2	2	2	2	2	2	2	2	2											
=	=	=	=	=	=	=	=	=	=	=	=											
=	*	*	*	*	*	*	*	*	*	*	=											
=	*				*		*		*		=											
=					*	*	*	*	*		=											
=	*	*	*	*	*	*	*	*	*		=											
=					*	*	*	*			=											
=	*	*	*	*	*	*	*	*	*		=											
=	*				*		*				=											
=	*	*	*	*	*	*	*	*			=											
=	*	*	*	*	*	*	*				=											
=	=	=	=	=	=	=	=	=			=											

2. Genes Initialization for the first generation



The screenshot shows the Evolution software interface with the title bar "Run EVOLUTION". The left sidebar contains various icons for file operations like run, stop, save, and delete. The main area displays a large number of randomly generated gene strings. Each string starts with a 10-digit ID followed by a sequence of 0s and 1s. There are approximately 20 such strings listed, each consisting of about 1000 digits.

106056020632240333431420161146055121550464666463050162464351433030202254324664520224431642012346033500
00465363604152532611343143313620641615353040632331146360632122261522240565045020612600646266435250006
22451352103662545521455541201051500366404026124434046060322466045330012541503214050543360315664053433
025612210262134543035614244626504152645115431544060163525506101500636422232650136056655055146500132252
256600221340333633265532616652161155032423656031354130452536400464454416512642450004610065663534416436
414423535661610250464205606160003503421005551634145453155556343024225062310103345562205416423654446622
435162516214620135401615061233022026554545205653061265504030210626420033602020036003131621504422103030
642403104226552340305302435046230256645132621231666116503513665446526640361456166204262633031423513236
2564055013621560211235655431332335445625103153440051505222052051304201326501301616245652003402520606
266653616210311534600426242012626126301652630630116524014443135100664431560441120250132262004563516
3144244226151132002653045561435512540143212425154503436660024913663300400631235551546665154420502123
65104223056364345546020102644445021450363225443106155550606526114051565036642266645116561335025016245
631331245222101063404016312014226436125646101162523655503146152232504134160421040204402460341332403
5630440563156020145654221430243323065026046246102606201340221315652532335021450214102103021221145046
604044523232364044156535455652031165231131123442662306634341502222103143430525421021145422306632414203
2424113405204244050225454165151035556433052140264221666022445035514042002215643021411466304346211630
25133626164135063653206632014122614400616063635551141365112242461656162212242132034512624326214225653
51335222313651345531344642651160335236624305551336523653551634301402166643436462505243344130262221561
226231005024225324164132500352260342041141230143213450623306501416214151314034435610640530135121525
523451613021610666006335655451010254522400155362032660630213663166452045666616162204043061241506326613
35260313504054664021400342322031360006236134113332564644642132626641205525511023623411325262206366431
35424536301634014041650611660245454361452214113565262543352326416046516315426420303632631214621356501
560546203525551252213563414551603411410301561614232061650102241050436141101610245011655132000412313060
0252040401236641540521510634100250121202553204034161533400104445550051565205565624110601126600602124251

3. Evolution Process

Run EVOLUTION

```

0 GENERATION BEST SCORE: 50 AVERAGE SCORE: 1.4384999999999997
1 GENERATION BEST SCORE: 20 AVERAGE SCORE: 2.0590000000000024
2 GENERATION BEST SCORE: 25 AVERAGE SCORE: 1.8585000000000047
3 GENERATION BEST SCORE: 55 AVERAGE SCORE: 1.7365000000000028
4 GENERATION BEST SCORE: 20 AVERAGE SCORE: 1.9105000000000047
5 GENERATION BEST SCORE: 40 AVERAGE SCORE: 2.4310000000000006
6 GENERATION BEST SCORE: 60 AVERAGE SCORE: 3.1300000000000012
7 GENERATION BEST SCORE: 30 AVERAGE SCORE: 3.2760000000000009
8 GENERATION BEST SCORE: 50 AVERAGE SCORE: 3.8245000000000124
9 GENERATION BEST SCORE: 40 AVERAGE SCORE: 3.2265000000000095
10 GENERATION BEST SCORE: 40 AVERAGE SCORE: 3.3005000000000093
11 GENERATION BEST SCORE: 40 AVERAGE SCORE: 3.1035000000000106
12 GENERATION BEST SCORE: 40 AVERAGE SCORE: 3.4235000000000086
13 GENERATION BEST SCORE: 40 AVERAGE SCORE: 4.5930000000000009
14 GENERATION BEST SCORE: 50 AVERAGE SCORE: 4.1010000000000001
15 GENERATION BEST SCORE: 30 AVERAGE SCORE: 3.7730000000000117
16 GENERATION BEST SCORE: 40 AVERAGE SCORE: 5.0675000000000106
17 GENERATION BEST SCORE: 40 AVERAGE SCORE: 3.2535000000000105
18 GENERATION BEST SCORE: 40 AVERAGE SCORE: 5.2455000000000006
19 GENERATION BEST SCORE: 70 AVERAGE SCORE: 3.5260000000000012
20 GENERATION BEST SCORE: 40 AVERAGE SCORE: 4.9925000000000011
21 GENERATION BEST SCORE: 50 AVERAGE SCORE: 4.3715000000000011
22 GENERATION BEST SCORE: 50 AVERAGE SCORE: 4.3975000000000012
23 GENERATION BEST SCORE: 50 AVERAGE SCORE: 4.6430000000000011

```

Navigate to the previous occurrence

Run EVOLUTION

```

2476 GENERATION BEST SCORE: 330 AVERAGE SCORE: 174.6769999999994
2477 GENERATION BEST SCORE: 315 AVERAGE SCORE: 170.5654999999993
2478 GENERATION BEST SCORE: 330 AVERAGE SCORE: 182.6379999999995
2479 GENERATION BEST SCORE: 325 AVERAGE SCORE: 178.4569999999994
2480 GENERATION BEST SCORE: 310 AVERAGE SCORE: 165.4149999999994
2481 GENERATION BEST SCORE: 350 AVERAGE SCORE: 177.4134999999989
2482 GENERATION BEST SCORE: 370 AVERAGE SCORE: 171.7669999999994
2483 GENERATION BEST SCORE: 380 AVERAGE SCORE: 177.9129999999999
2484 GENERATION BEST SCORE: 320 AVERAGE SCORE: 174.4354999999996
2485 GENERATION BEST SCORE: 350 AVERAGE SCORE: 188.0084999999999
2486 GENERATION BEST SCORE: 340 AVERAGE SCORE: 182.1369999999999
2487 GENERATION BEST SCORE: 330 AVERAGE SCORE: 178.4629999999999
2488 GENERATION BEST SCORE: 330 AVERAGE SCORE: 170.2649999999999
2489 GENERATION BEST SCORE: 310 AVERAGE SCORE: 178.28749999999994
2490 GENERATION BEST SCORE: 350 AVERAGE SCORE: 164.5485
2491 GENERATION BEST SCORE: 350 AVERAGE SCORE: 177.9619999999993
2492 GENERATION BEST SCORE: 350 AVERAGE SCORE: 168.4904999999999
2493 GENERATION BEST SCORE: 340 AVERAGE SCORE: 167.4949999999999
2494 GENERATION BEST SCORE: 330 AVERAGE SCORE: 165.99099999999987
2495 GENERATION BEST SCORE: 330 AVERAGE SCORE: 176.1629999999993
2496 GENERATION BEST SCORE: 320 AVERAGE SCORE: 172.5184999999996
2497 GENERATION BEST SCORE: 370 AVERAGE SCORE: 166.6924999999994
2498 GENERATION BEST SCORE: 330 AVERAGE SCORE: 175.7624999999999
2499 GENERATION BEST SCORE: 350 AVERAGE SCORE: 170.9699999999999

```

4. Statistic Results to Create Genes

Run EVOLUTION

```

===== gene Position: 0
Ratio of 0: 100.0% Ratio of 1: 0.0% Ratio of 2: 0.0% Ratio of 3: 0.0% Ratio of 4: 0.0% Ratio of 5: 0.0% Ratio of 6: 0.0%
gene Position: 81
Ratio of 0: 0.0% Ratio of 1: 0.0% Ratio of 2: 0.0% Ratio of 3: 0.0% Ratio of 4: 0.0% Ratio of 5: 0.0% Ratio of 6: 100.0%
gene Position: 3
Ratio of 0: 0.0% Ratio of 1: 0.5% Ratio of 2: 0.0% Ratio of 3: 99.5% Ratio of 4: 0.0% Ratio of 5: 0.0% Ratio of 6: 0.0%
gene Position: 30
Ratio of 0: 0.0% Ratio of 1: 100.0% Ratio of 2: 0.0% Ratio of 3: 0.0% Ratio of 4: 0.0% Ratio of 5: 0.0% Ratio of 6: 0.0%
gene Position: 59
Ratio of 0: 0.0% Ratio of 1: 0.0% Ratio of 2: 100.0% Ratio of 3: 0.0% Ratio of 4: 0.0% Ratio of 5: 0.0% Ratio of 6: 0.0%

```

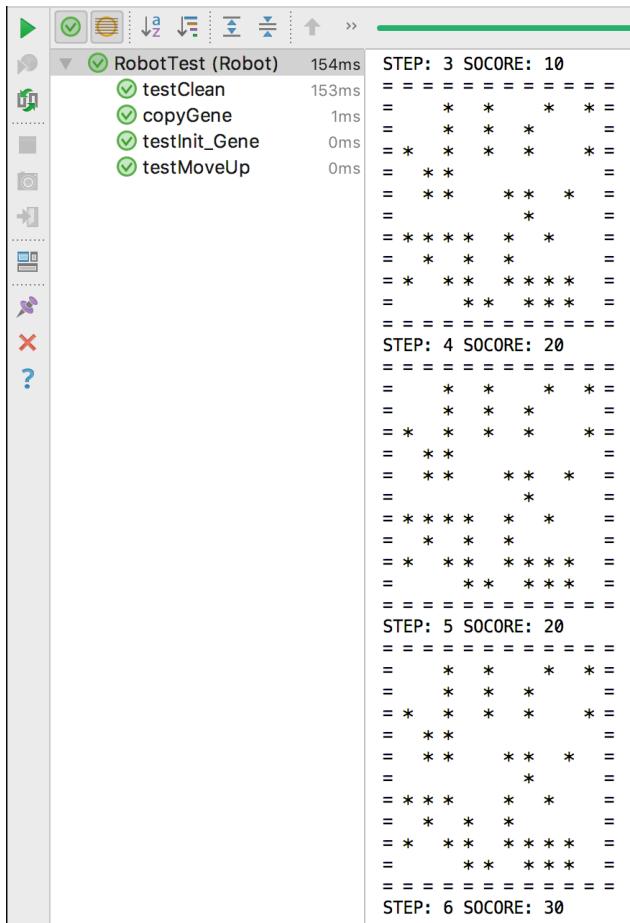
5. Best Gene

6. Route Records

STEP: 99 SOCORE: 250
===== * * * * * =====
===== ◆◆ * * * =====
===== ◆◆ * * ◆ * * * =====
===== * * ◆◆ ◆ * =====
===== ◆ * * * * * * * =====
===== * ◆◆◆◆◆◆◆◆◆ * =====
===== ◆◆◆◆◆◆◆◆◆ * * =====
===== ◆◆ * * ◆◆ ◆ * * =====
===== ◆ * * =====

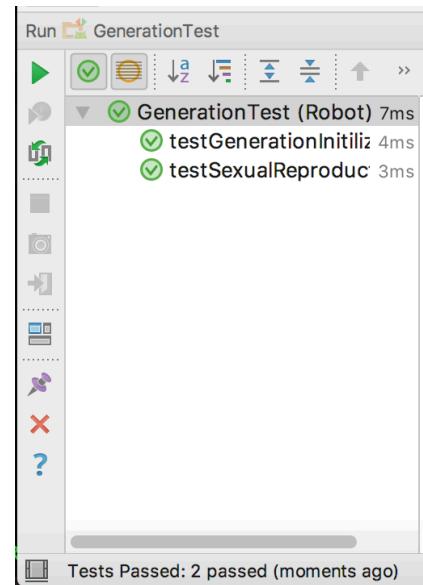
Unit Tests

Robot Tests:



The screenshot shows the Robot Test interface. On the left, there's a toolbar with icons for play, stop, run, and others. Below it is a tree view showing a test suite named "RobotTest (Robot)" which contains four tests: "testClean", "copyGene", "testInit_Gene", and "testMoveUp". The "testClean" test took 153ms and passed. The "copyGene" test took 1ms and passed. The "testInit_Gene" test took 0ms and passed. The "testMoveUp" test took 0ms and passed. To the right of the tree view is a large grid visualization consisting of a 10x10 grid of asterisks ('*'). The grid is divided into several sections by horizontal and vertical lines, representing different states or steps. Labels above the grid indicate "STEP: 3 SOCORE: 10", "STEP: 4 SOCORE: 20", "STEP: 5 SOCORE: 20", and "STEP: 6 SOCORE: 30".

Generation Tests:



The screenshot shows the Generation Test interface. On the left, there's a toolbar with icons for play, stop, run, and others. Below it is a tree view showing a test suite named "GenerationTest (Robot)" which contains two tests: "testGenerationInitializ" and "testSexualReproduc". Both tests passed, taking 4ms and 3ms respectively. At the bottom of the interface, a message box displays "Tests Passed: 2 passed (moments ago)".

Map Tests:

