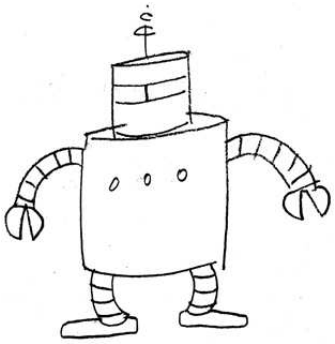
Project Assignment #1: Robby The Soda Can Collector

In this project your goal is to teach Robby a set of rules to collect cans. Robby is a baby-bot, and can not intelligently make any decisions. It also has a limited vision: it can only see five sites (Current, North, East, South, West). On top of this, Robby lives in a miserable small world, which consists of a 10×10 grid of cells. You will use genetic algorithm to improve a table of rules (DNA sequence).



1 Preparation To-Dos

1. Read the reading material posted on LMS. Chapter 9 from Complexity: A Guided Tour by Melanie Mitchell
2. Make sure you have Python 2.7 installed
3. Make sure you install Tkinter package
4. Download the source code (robby.tar.gz) from LMS, extract it to your local disk

2 Implementation

(60 points, 10 points each in the below items)

In the source code, you will observe that some functions and parts of functions are left out for your implementation (in six different places). So the code will not automatically run without your implementation. You need to fill out the necessary places with your code. Since you will have many bugs in your implementations, it is advisable that you

make the code run faster by decreasing its complexity. Find the file *Parameters.py* and decrease the parameter values such as NUM_GENERATIONS, etc.

Here are the six places that require your code.

1. `Robby.getRandomRobby()`: Creates a random robot by creating a random DNA sequence: 1D array of 3^5 actions.

2. `Robby.getNextAction()`: Some part of this function is already implemented, you will implement the rest. This function moves the robot by checking its five sites and retrieving the appropriate action from the DNA sequence.

3. `Robby.giveBirth()`: This functions takes another robot instance as an input (its mate) and produces two children from them. You need to implement the cross-over logic and call the mutation function.

4. `Robby.mutate()`: Mutates a DNA sequence with a given mutation probability (MUTATION_PROBABILITY).

5. `Generation.getRouletteWheelSelection()`: Uses roulette wheel selection logic. Picks a number depending on the rankings probabilistically.

6. `Generation.applyEvolution()`: Apply evolution logic on a list of sorted population. Use `getRouletteWheelSelection()` to pick a mate for a robot, and let them have two children using the `giveBirth()` function

3 Your tasks (40 points, 10 points each)

After finishing the implementation you will asses the performance of the genetic algorithm and analyze how well Robby will perform in the world.

1. Create a plot of fitness scores of a generations best performing robot versus the generation ids.

2. Set MUTATION_PROBABILITY to 0.0 and 0.5. And discuss the results in terms of the best robot at the end of the generations (perfect robot).

3. Check if the perfect robot can collect all three vertically or horizontally located cans starting from the middle. You need to find out which actions does this job correspond to in the DNA sequence (indexing). You may need to play with the configuration parameters so that Robby can learn this trick in time. See pages 138 and 139 from the reading material.

4. Count the number of average STAY_PUT actions in all generations. And plot this number versus the generation ids. The number should go down as the generations progress.

4 Bonus Questions

1. Find a bug in the code (20 points)
2. Implement a webpage which takes a DNA sequence (comma separated 243 action values) and visualizes the robot on a randomly created grid. (30 points).

Good luck!