Lab 6: Robby the Robot

In this lab you will experiment with using a genetic algorithm to evolve control strategies for Robby the Robot. Write down your answers to each of the starred (*) exercises as you go. You will submit these answers with your next homework assignment.

1. Start the Robby simulator. Two control panels will appear: *Robby Control Panel*, which can be used to directly control the robot, and *GA Control Panel*, which controls the GA. Manually move Robby around by clicking on the N/S/E/W buttons (you can also use the arrow keys). The *Grab* button will cause Robby to try to pick up a can. The *Random config* button and the *Can density* slider can be used to randomly distribute cans in the world.

2. Several strategies are available in the pull-down menu: Strategy M which was hand-designed by Melanie Mitchell; Strategy G which was found by her GA; the modified version of Strategy G described on page 140 of *Complexity*; a strategy that moves around at random; a strategy that always moves to the east; a randomly-generated strategy; and the best strategy in the current GA population. Try out these strategies using the *Step*, *Run*, *Stop*, and *Eval* buttons on different random soda can configurations. *Eval* reports the average reward earned by a strategy on a large number of cleaning sessions (simulated with the graphics turned off).

3. Now switch to the *GA Control Panel* window. The GA is set to a default population size of 200, with no elitism, a crossover rate of 1.0, and a mutation rate of 0.005. The *Show best* and *Show worst* buttons display the best and worst strategies in the current population, along with their fitness values (calculated as the average reward earned on 100 cleaning sessions, each consisting of 200 actions). The *Demo best* and *Demo worst* buttons simulate the respective strategies on one cleaning session with the graphics on, and report the total reward earned by the strategy. The *Show population* button shows all strategies in the current population. Try demoing the best and worst strategies a few times to get a sense of how they perform on different random starting configurations.

4. Start the GA by clicking *Evolve*, and let it run for 20 generations, then click *Stop*. Demo the best strategy again a few times. Do you notice any general improvement? Continue evolving for another hundred generations or so, stopping every 20 generations to demo the best strategy. After 200 generations, what is the fitness of the best strategy? What is the average fitness of the population?

5. We can speed things up by using a smaller population size, but this may make it harder for the GA to discover good strategies. Change the population size to 100, click *Initialize GA*, then run it for 200 generations. How do the results compare with your previous run using a larger population size? Try this experiment three times, each time recording the best fitness achieved in generation 200.

6. Now change the *Elitism* parameter to 0.10, meaning that on each generation, the best strategies in the top 10% of the population will get copied unchanged into the next generation. Then rerun your previous experiment. That is, with a population size of 100 and 10% elitism, what is the best fitness achieved by the GA in generation 200? Try this three times to test how reliable your results are.

7. (*) Reset *Elitism* to 0 for the next series of experiments, which will test the influence of the mutation rate on the GA. Which mutation rate will result in a higher final fitness on average: 0, 0.01, 0.05, 0.1, or 0.2? Make a hypothesis, briefly explain your reasoning, and then test it. For each mutation rate, run the GA five times with a population size of 50, and record the average value of the best fitness obtained in generation 50 in the table below:

Number of generations	Population size	Mutation rate	Best fitness in generation 50 (averaged over 5 runs)
50	50	0	
50	50	0.01	
50	50	0.05	
50	50	0.1	
50	50	0.2	

8. (*) Which crossover rate will result in a higher final fitness on average: 0, 0.5, or 1? Make a hypothesis, briefly explain your reasoning, and then test it. For each crossover rate, run the GA five times with a population size of 50, and record the average value of the best fitness obtained in generation 50 in the table below:

Number of generations	Population size	Crossover rate	Mutation rate	Best fitness in generation 50 (averaged over 5 runs)
50	50	0	0.05	
50	50	0.5	0.05	
50	50	1	0.05	

9. (*) Make a hypothesis about the effect of increasing elitism on the GA and then test it, by running a similar experiment using elitism values of 0, 0.05, 0.10, 0.20, 0.30, and 0.50. For this experiment, set the crossover and mutation rates to the default values of 1 and 0.005, respectively.

Number of generations	Population size	Elitism	Best fitness in generation 50 (averaged over 5 runs)
50	50	0	
50	50	0.05	
50	50	0.1	
50	50	0.2	
50	50	0.3	
50	50	0.5	

Was your hypothesis confirmed?

10. (*) Based on your previous experiments, set the GA parameters to a combination that you think will give optimum performance, and then run the GA for at least 300 generations. Can the GA discover a strategy with a fitness over 400? Over 500? If so, how many generations are required in each case?

11. (*) Once your GA has discovered a highly effective strategy, switch back to the *Robby Control Panel* window and choose *Current best GA strategy* from the pull-down strategy menu. Then run this strategy on some random can configurations with "nonstandard" densities such as 10%, 90%, or 100%. (To do this, set the *Can density* slider to the appropriate value and click the *Run* button. Don't click *Eval*, since that always evaluates a strategy using a density of 50%.) Given that the strategy was evolved with a fitness function that always used a 50% can density, does the strategy still appear to be as effective at cleaning up a more (or less) cluttered environment? Briefly describe the strategy's behavior on each of the three nonstandard can densities.