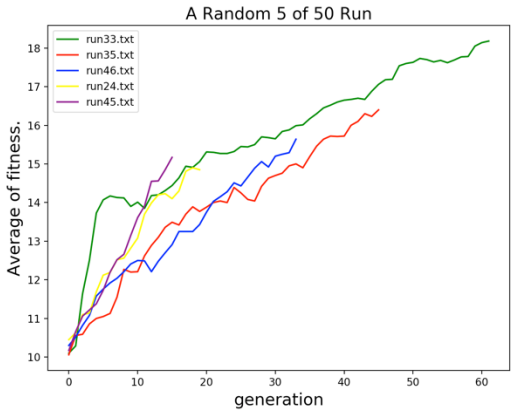
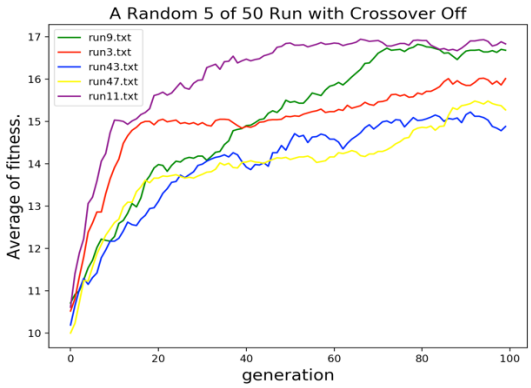
Name: Chakrya Ros

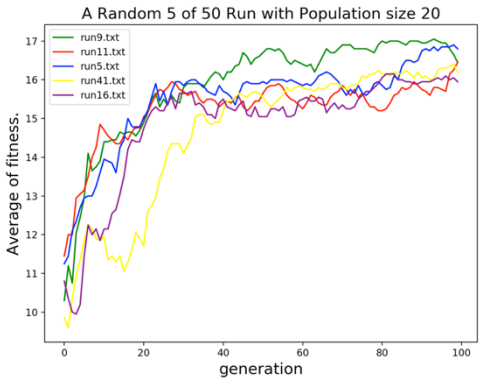
Part 1

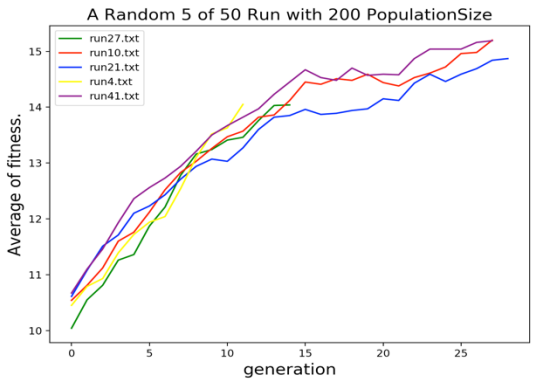
1. Perform 50 runs, maximum are 20 for every run.
2. The similarity is increasing of generation also makes the average of fitness increase. With the high probability of crossover rate i.e. 0.7, we can find the optimal solution that all string is one between generation 16 to 25. The differences are that these five runs of runGA find different optimal string. Like the purple color, find the optimal string just in 15 generations, and the green color, find the optimal string in next 60 generations.





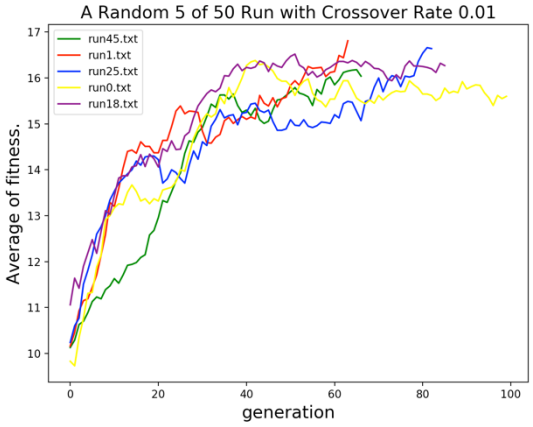
3. When I performed the same experiment with crossover turned off (p = 0), I cannot find the optimal string. In GA, the crossover combined two parents to form children for the next generation. Crossover rate is the ratio of next generation population born by crossover operation. Without doing crossover rate and just small mutation rate, we just ran the population with randomly gens that passed to next generation. The mutation is randomly changed individual parents to form the children. Mutation let GA to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, so it slows or stops evolution. That's why when crossover rate is off, just randomly changed individual parent to produce offspring for next generation. Therefore, crossover is very effect on GA.
4. Population size:
   - when I decreased population size to 20 with the same crossoverRate = 0.7 and mutateRate = 0.001, it didn't find the optimal strings because it does not much pair of parents to produce better offspring.
   - when I increased population size to 200 with the same crossoverRate = 0.7 and mutateRate = 0.001, we found the optimal strings in between 13 to 26 generations because it has more populations to produce better offspring. Thus, the population size is effect GA.

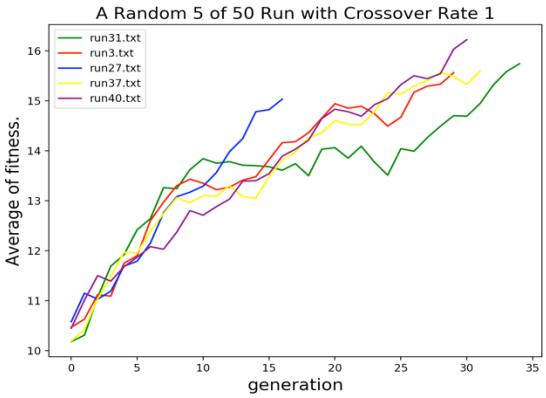




CrossoverRate:

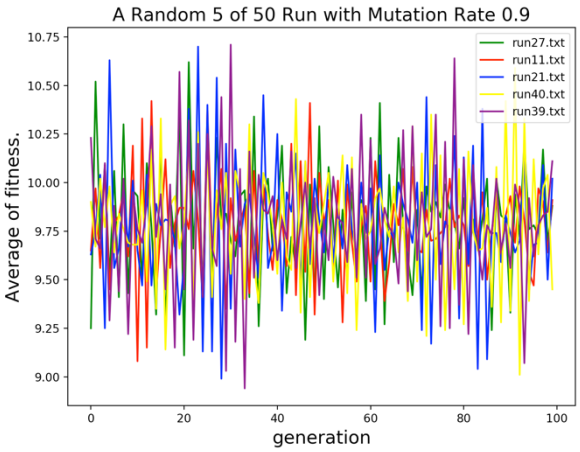
- When I decreased to 0.01 with the same mutateRate = 0.001 and population size, it's take more generation to find the optimal string in between 70 to 90 generations, but not all five runs get optimal string.
- When crossoverRate = 1 with the same mutateRate = 0.001 and population size, we find the optimal string in all five runs in between 15 to 35 generation. Thus, crossover rate is effect GA. The high crossover rate makes it get fast to find optimal string in next generation.

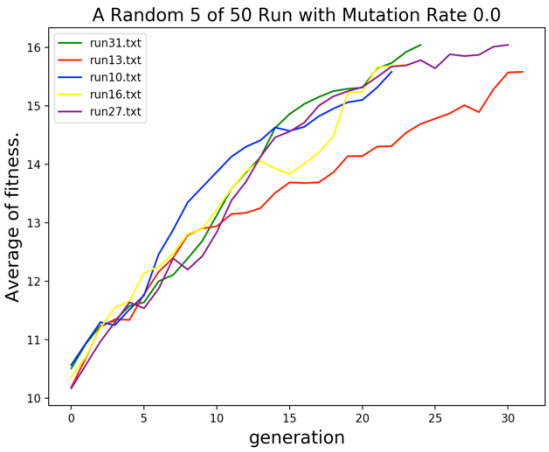




MutateRate:

- When I increased muateRate to 0.9 with the same crossoverRate = 0.7 and population size, we can't find the optimal spring. It's increasing and decreasing of the average of fitness. Looking at the graph, it looks like the wave that go up and down because we randomly changed individual parents to form the children is very high. Too high mutation rate prevents population to converge to any optimal string.
- When I decreased muateRate to 0.0 with the same crossoverRate = 0.7 and population size, just between 23 to 30 generation, we could find the optimal string. So It means that without mutateRate, offspring that passed to next generation get better with high crossover rate.

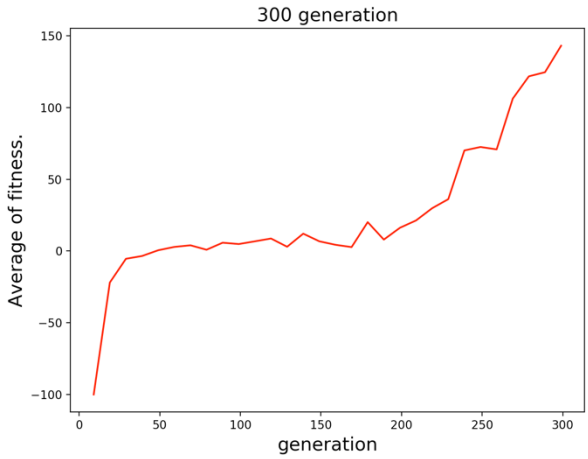




Part 2

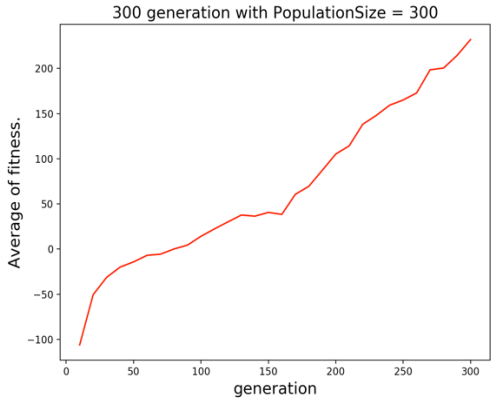
From my experiment with 100 populations, crossover rate 1 (100%) , mutate 0.005 and 300 generation runs, I can see it's improved the average fitness from -100.50 in the first generation to 143.18 in 300th generation . It means that the first generation didn't have a better offspring, it's hit a wall a lot that's why It got negative average fitness. Another factor that it got a lot of negative average is agent start a state at (0,0) that is on the top left corner in the grid. Then it started improving from one generation to another generation because of crossover and mutate, Robby the Robot started to pick up more cans than hit the wall.

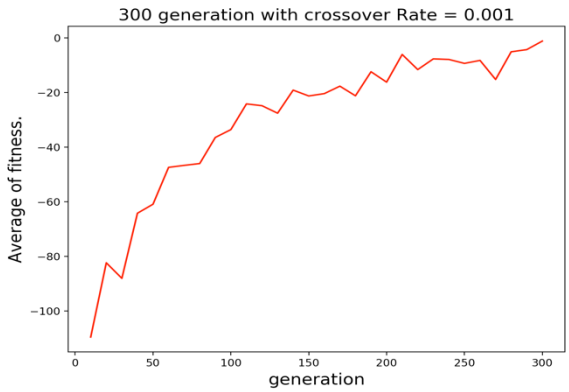
In genetic algorithm, selection, crossover and mutate are variation to make Robby a Robot perform better. Firstly, Selection is the phase that select the fitness individuals and pass their gens to next generation. We need to pick a pair of parents based on their fitness scores, the higher fitness score, the better chance to select for reproduction. Secondly, crossover is the most important phase. Crossover take place by picking a random point in the genome and exchanging genes before and after such points from its parents for every pair of parents to form children in next generation. Because in these 300 generations, we set crossover rate to 1, it did crossover operation every generation in 100 population. This help to improve performance of agent. Another variation to improve this performance is mutation. The mutation is randomly changed individual parents to form the children. We need to set mutate rate as low as possible because Mutation helps to maintain diversity within the population and prevent the premature convergence. That's why's improve. Look at the graph below.



In second experiment, I set population size to 300 with the same crossover rate, mutate rate. This is experiment, I also initial start state at random positions. This experiment is improved much performance more than the first experiment from -113.1 to 231.88. That's good. The reason that it's improved a lot, I think because the starting state is not always in the top left corner. It could be in the middle that have more cans

and not close to wall. Another reason is that more population also produce the better offspring for next generation. Looking at graph on the left, you can see it's increasing.





When I tried another experiment with the same population, generation and mutate rate, but I set crossover rate to 0.001 and random starting state. The output did not improve much as crossover rate at 1 and larger population size. The reasons that it didn't improve because we created 300 generations, program mostly selected the high fitness score of each pair of parents pass to next generation. Lower crossover rate really affects this run because it had only 0.1 percent to do crossover. With lower crossover, it didn't produce the better offspring for next generation. For example, if the Robot got stuck in the corner at top left, the genome contains strings of "02020202020202024420244", it would hit the wall and stay put there for the whole string. It would repeat that in next generation. With mutate rate alone, it didn't help to improve the performance because it just randomly replaces gene of individual parent form the next children. If the mutate mostly random pick a gene of "4", it would not help this program, the Robot would keep hit the wall and did not move. Thus, mutate rate and crossover rate and population size play very important role in genetic algorithm to improve the performance to reach the global optimal.