Course: CS30A1570 Complex Systems

Assignment 5: Genetic Algorithms

Completion Date: 27 April 2024

Submitted by: Turzo Ahsan Sami

Title: Optimization of Genetic Algorithms: An Exploration of Parameter Effects

Introduction:

Genetic algorithm (GA) is an evolutionary algorithm that mimics the process of natural selection such as selection, crossover, mutation, and inheritance to solve complex optimization problems. GA has a set of properties that guide their behavior, these properties include mutation rate, population size, and crossover probability. In this study, GA is being used to solve the optimization problem using strategies within a controlled environment (population size, mutation rate, and the number of generations) to calculate fitness. By experimenting with these parameters, it can be observed how these parameters impact the model and the result. It can also be observed how GA is used to find the optimum solutions.

Research Question

The research question related to this investigation are as follows:

- Task 1.1: How does the mutation rate affect the optimization process of the genetic algorithm?
- Task 1.2: How does the population size influence the optimization process of the genetic algorithm?
- Task 2.1: How do the "hidden" parameters (rewards and penalties for various actions, the number of environments each strategy is tested on to compute its fitness, and the number of actions each strategy takes in each environment) affect the optimization process of the genetic algorithm?
- Task 2.3: How does the crossover probability influence the optimization process of the genetic algorithm?

Results:

The result obtained from the thorough investigation of this research are presented below:

Task 1.1: Experiment with Different Mutation Rate

Table-1 displays the result of simulation run with different mutation rates (0.05, 0.1, and 0.01) while keeping the number of generations (200) and population size (100) constant. The average best fitness in the final generation seems to be higher with a mutation rate of 0.05 compared to the other two values. The result reflects that the evolutionary process appeared to be disturbed by a greater mutation rate (0.1) and lower mutation rate (0.01). A higher mutation rate may introduce more features which are not advantageous upon the model, and this results in a substantially poorer fitness. On the other hand, lower fitness may not have enough diversity to develop more effective solutions.

Table 1: Fitness Dependence on Mutation Rate

Simulation Parameters	Mutation rate	Average Best Fitness
Number of Generations: 200	0.05	161.3
Population Size: 100	0.1	59.7
	0.01	62.4

Task 1.2: Experiment with Different Population Sizes:

Table-2 displays the result of simulation run with different mutation rates (0.05, 0.1, and 0.01) while keeping the number of generations (200) constant. However, this time the population size was varied slightly (100 and 300).

Table 2: Fitness Dependence on Population

Number of Generations	Population Size	Mutation Rate	Average Best Fitness
200	100	0.05	153.6
200	100	0.1	55.6
200	100	0.01	69.1
200	300	0.05	186.8
200	300	0.1	52.9
200	300	0.01	202.9

The population size in a genetic algorithm (GA) significantly affects the optimization process in two main ways: exploration vs. exploitation and convergence speed.

Exploration vs. Exploitation: Larger population sizes allow for more diverse individuals in the population. This increases the chance of exploring a wider range of potential solutions in the search space, which can be helpful for finding the global optimum (the absolute best solution). However, a larger population can lead to slower convergence (finding a good solution) because there's a higher probability of selecting fewer fit individuals during selection. Smaller population sizes on the other hand leads to faster convergence as selection focuses on fitter individuals within a limited pool. This can be good for quickly finding a locally optimal solution (good but not necessarily the absolute best). However, they reduce exploration of the search space, potentially missing the global optimum if it's located in a less explored area.

Convergence Speed: A larger population size generally leads to slower convergence because there are more individuals to evaluate for fitness. With more evaluations needed per generation, it takes more iterations to reach a good solution.

There's a trade-off between exploration and exploitation when choosing a population size. If there are multiple good solutions or the search space is complex, a larger population size might be better to explore more possibilities. If finding a good solution quickly is a priority, a smaller population size might be sufficient, especially if the solution quality is acceptable.

From the data obtained from running multiple simulation runs, it seems like a population size of 300 (compared to 100) with a mutation rate of 0.05 achieves a higher average best fitness (186.8 vs 153.6). This suggests that the larger population allows for better exploration and potentially leads to finding better solutions.

Task 2.1: Experimenting with "hidden" parameters

The hidden parameters of the netlogo model namely Wall-Penalty and Can-Reward were modified to observe their impact upon the model (Appendix: Attachment 1).

Varying the parameter value of wall penalty resulted in the average best fitness to increase as the wall penalty gets higher. This is because evolutionary pressure favors strategies that avoid hitting walls. Strategies that take more risks and bump into walls more often will be penalized heavily, leading to their selection against during evolution. Table 3 displays the result of this simulation. The results show that as the wall penalty increases (from 2 to 8), the average best fitness also increases (from 55.7 to 188.5). This suggests that the genetic algorithm is successfully evolving strategies that prioritize avoiding walls in environments with a higher penalty for collisions.

Table 3: Fitness Dependence on Wall Penalty

Simulation Parameters	Wall Penalty	Average Best Fitness
Number of Generations: 200	5	161.3
Population Size: 100	2	55.7
Mutation Rate: 0.05	8	188.5

Varying Can Reward parameter values results in the average best fitness to increase as the can reward gets higher. This is because evolutionary pressure will favor strategies that collect more cans. With a higher reward for collecting cans, strategies that prioritize finding and picking up cans will be more likely to be selected and reproduced in future generations. The results obtained (Table 4) shows that there is a increase in average best fitness from 22.6 to 161.3 when the can reward increases from 2 to 10.

Table 4: Fitness Dependence on Wall Penalty

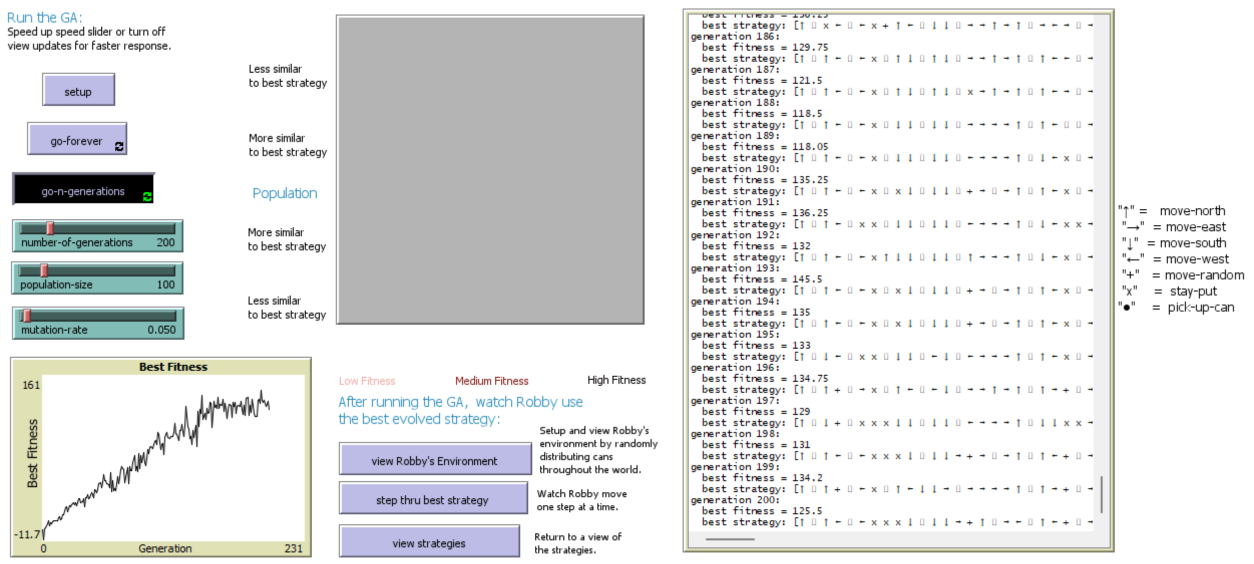
Simulation Parameters	Can Reward	Average Best Fitness
Number of Generations: 200	10	161.3
Population Size: 100	2	22.6
Mutation Rate: 0.05	7	119.4

Varying number of actions result taken in each environment also impacts the best average fitness. For calculating fitness from 10 to 250 it was noticed that the average fitness increased from 10 to 134.8. However, it is maximum for the number of actions 100. This indicates that as the number of actions increases, the population becomes more capable of finding an optimum solution.

Table 5: Fitness Dependence on Number of actions

Simulation Parameters	Number of Actions	Average Best Fitness
Number of Generations: 200	100	161.3
Population Size: 100	10	40
Mutation Rate: 0.05	200	134.8

Image 1: RobbyGA.nlogo model simulation



Speed up speed slider or turn off view updates for faster response.

number-of-generations 200
population-size 100
mutation-rate 0.050

Setup and view Robby's environment by randomly distributing cans throughout the world.

Watch Robby move one step at a time.

Return to a view of the strategies.

```
  best fitness = 130.25
  best strategy: [↑ ▯ x ← ▯ ← x + ↑ ← ▯ ↓ ↓ ▯ → → ↑ → ↑ ▯ → ← → ▯ -
generation 186:
  best fitness = 129.75
  best strategy: [↑ ▯ ↑ ← ▯ ← x ▯ ↑ ↓ ▯ ↑ ↓ ▯ → → ↑ → ↑ ▯ ↑ ← ← ▯ -
generation 187:
  best fitness = 121.5
  best strategy: [↑ ▯ ↑ ← ▯ ← x ▯ ↑ ↓ ▯ ↑ ↓ ▯ x → ↑ → ↑ ▯ ↑ ← → ▯ -
generation 188:
  best fitness = 118.5
  best strategy: [↑ ▯ ↑ ← ▯ ← x ▯ ↓ ↓ ▯ ↓ ↓ ▯ → → → → ↑ ▯ ↑ ← ▯ ▯ -
generation 189:
  best fitness = 118.05
  best strategy: [↑ ▯ ↑ ← ▯ ← x ▯ ↓ ↓ ▯ ↓ ↓ ▯ ← → → → ↑ ▯ ↓ ← x ▯ -
generation 190:
  best fitness = 135.25
  best strategy: [↑ ▯ ↑ ← ▯ ← x ▯ x ↓ ▯ ↓ ↓ ▯ + → ▯ → ↑ ▯ ↑ ← x ▯ -
generation 191:
  best fitness = 136.25
  best strategy: [↑ ▯ ↑ ← ▯ x x ▯ ↓ ↓ ▯ ↓ ↓ ▯ ← → → → ↑ ▯ ↓ ← x x -
generation 192:
  best fitness = 132
  best strategy: [↑ ▯ ↑ ← ▯ ← x ↑ ↓ ↓ ▯ ↓ ↓ ▯ ↑ → → → ↑ ▯ ↓ ← x ▯ -
generation 193:
  best fitness = 145.5
  best strategy: [↑ ▯ ↑ ← ▯ ← x ▯ x ↓ ▯ ↓ ↓ ▯ + → ▯ → ↑ ▯ ↑ ← x ▯ -
generation 194:
  best fitness = 135
  best strategy: [↑ ▯ ↑ ← ▯ ← x ▯ x ↓ ▯ ↓ ↓ ▯ + → ▯ → ↑ ▯ ↑ ← x ▯ -
generation 195:
  best fitness = 133
  best strategy: [↑ ▯ ↓ ← ▯ x x ▯ ↓ ↓ ▯ ← ↓ ▯ ← → → → ↑ ▯ ↑ ← x ▯ -
generation 196:
  best fitness = 134.75
  best strategy: [↑ ▯ ↑ + ▯ → x ▯ ↑ ← ▯ ← ↓ ▯ → → ↑ → ↑ ▯ ↑ → + ▯ -
generation 197:
  best fitness = 129
  best strategy: [↑ ▯ ↓ + ▯ x x x ↓ ↓ ▯ ↓ ↓ ▯ ← → → → ↑ ▯ ↓ ↓ x x -
generation 198:
  best fitness = 131
  best strategy: [↑ ▯ ↑ ← ▯ ← x x x ↓ ▯ ↓ ↓ → + → ▯ → ↑ ▯ ↑ ← + ▯ -
generation 199:
  best fitness = 134.2
  best strategy: [↑ ▯ ↑ + ▯ ← x ▯ ↑ ← ↓ ↓ → ▯ → → → → ↑ ▯ ↑ → + ▯ -
generation 200:
  best fitness = 125.5
  best strategy: [↑ ▯ ↑ ← ▯ ← x x x ↓ ▯ ↓ ↓ → + ↑ ▯ → ← ▯ ↑ ← + ▯ -
```

"↑" =	move-north
"→" =	move-east
"↓" =	move-south
"←" =	move-west
"+" =	move-random
"x" =	stay-put
"●" =	pick-up-can

Task 2.2: Allowing user to set crossover probability

There are two opposing forces at play when varying the crossover probability. High Crossover Probability (close to 1) promotes exploration of the search space by creating many new combinations of genes from the parents. This can be beneficial for finding new and potentially better solutions, especially early in the evolution process. Low Crossover Probability (close to 0) promotes exploitation of good existing solutions by creating offspring that are similar to the parents. This can help refine good solutions that have already been discovered. Therefore, a non-monotonic relationship exists between crossover probability and average best fitness. Table 6 shows the data obtained from netlogo model run with varying crossover.

High Crossover Probability (0.7 & 0.9): Both these scenarios show a significant decrease in average best fitness compared to the 0.3 and 0.5 probabilities. This suggests that excessive crossover disrupts good gene combinations, hindering the evolution of high-performing strategies.

Low Crossover Probability (0.1): The performance with a 0.1 crossover probability is lower than the 0.3 and 0.5 probabilities, but not as dramatically low as the high crossover settings. This suggests that even with a low probability, some beneficial crossover might still be happening, but the exploration of new possibilities might be limited.

Intermediate Crossover Probability (0.3 & 0.5): These settings show the highest average best fitness (59.5 and 104.5 respectively). This suggests that these crossover probabilities strike a good balance between exploration and exploitation, allowing the genetic algorithm to explore new possibilities while also preserving beneficial traits from parent strategies.

Table 6: Fitness Dependence on Crossover probability

Number of Generations	Population Size	Mutation Rate	Crossover Probability	Average Best Fitness
200	100	0.05	0.1	240.7
200	100	0.05	0.3	59.5
200	100	0.05	0.5	104.5
200	100	0.05	0.7	62.5
200	100	0.05	0.9	54.8

Discussion: The results of this study highlight the balance amongst the parameters of a genetic algorithm. The mutation rate, population size, and crossover probability all play crucial roles in the algorithm's ability to effectively explore for an optimal solution. Furthermore, the "hidden" parameters, such as rewards and penalties for various actions, can significantly impact the behavior of the evolved strategies. For instance, a higher mutation rate can introduce more diversity into the population, potentially leading to the discovery of novel and superior solutions. However, if the mutation rate is too high, it can disrupt beneficial traits in the population and hinder the algorithm's performance. Similarly, a larger population size can enhance the algorithm's exploratory capabilities but may slow down the convergence speed. The crossover probability also presents a trade-off between exploration and exploitation. A high crossover probability promotes the generation of diverse offspring, potentially leading to the discovery of better solutions. In contrast, a low crossover probability favors the preservation of good existing solutions, which can help refine these solutions but may limit the exploration of new possibilities. The "hidden" parameters, such as the rewards and penalties for various actions, can shape the behavior of the evolved strategies. For example, increasing the penalty for hitting walls or the reward for collecting cans can drive the evolution of strategies that prioritize avoiding walls or collecting cans, respectively**.** In conclusion, this study highlights the importance of carefully setting the parameters of a genetic algorithm to achieve effective optimization. It also highlights the potential of the RobbyGA.nlogo program as a versatile tool for studying genetic algorithms and their applications.

Appendices:

1. Attachment 1: https://github.com/ahsan-sami-turzo/complex-system-code/blob/main/CS_Assignment_5/RobbyGA_v_6.1.2.nlogo