Report

Farid Gahramanov

Farid Gahramanov

Introduction

This project uses a genetic algorithm to solve a problem: optimizing a linear function using given datasets. The goal is to demonstrate how the genetic operators like crossover, mutation and selection can be used together with a right strategy to find the best solutions.

In this report I will examine the structure and operation of the genetic algorithm, describing the reasons for the selection of specific genetic operators and their impact on algorithms performance. I did two case scenarios, one with different structure of a genetic algorithm and one with other. To demonstrate the algorithm's performance in handling optimization issues, I also evaluated its performance on several kinds of datasets that was given already.

Methodology (Main Case)

Case#1

A genetic algorithm's primary function is to generate a population of solutions to a given problem that change over a period of generations. The individuals (chromosomes) in this population are evaluated, selected for reproduce, and crossed over to create new offspring’s and then altered through mutation to add diversity. The effectiveness of a genetic algorithm mainly depends on its genetic operators: elitism, selection, crossover, and mutation. For case #1 I choose this structure, here is how each operator works in my algorithm:

- Selection: Tournament selection strategy ○ Tournament Selection is a Selection Strategy used for selecting the fittest candidates from the current generation in a Genetic Algorithm. These selected individuals are then

passed on to the next generation. In this way many such tournaments take place, and we have our final selection of candidates who move on to the next generation. Tournament selection is one of the most used selection operators in GAs due to its simplicity and effective at maintaining genetic diversity within the population.

- Crossover: Single-point crossover strategy ○ Crossover is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. Two strings are picked from the mating pool at random to crossover in order to produce superior offspring which then are passed on to the next generation. Single-point crossover is straightforward and achieves a nice mix between exploring new solutions and taking use of the most effective ones that have already been discovered.

- Mutation: Random mutation strategy ○ Mutation Operator is a unary operator, and it needs only one parent to work on. It does so by selecting a few genes from our selected chromosome and then by applying the desired mutation operator on them.

The strategies we use for selection, crossover, and mutation allowed me to find a good mix of trying out new genetic options and using the best solutions I have already found. This balance is important for the success of a genetic algorithm.

Methodology

Case#2

Farid Gahramanov

I created the “Case#2” in order to use alternative strategies of genetic algorithms in order to compare and contrast their effectiveness with the standard approaches used in Case #1. Determining if different strategies of crossover, mutation, and selection procedures may improve the algorithm's capacity to handle optimization problems more effectively and efficiently. The main difference in Case#2 is the genetic operations, more detailed, the strategies that was used for these operations.

- Selection: Roulette wheel selection
  - Instead of tournament selection, Case #2 uses the roulette wheel selection method. Roulette selection is a stochastic selection method, where the probability for selection of an individual is proportional to its fitness. The choice of using roulette wheel selection was made because of its probabilistic features, which may offer a smoother selection process.

- Crossover: Uniform crossover
  - Unlike the single-point crossover used in Case #1, Case #2 implements uniform crossover. In this method, each gene from the parent chromosomes has an equal probability of being chosen for the offspring, resulting in a more mixed genetic combinatios. This method is different from “single-point” mainly because it has an ability to combine genetic material more diversely, which can explore new areas in the solution space that single-point crossover might miss.

- Mutation: Increased Mutation Rate with Gaussian Noise
  - In Case #2, the mutation rate is raised, and mutations are applied using Gaussian noise instead of basic random mutations.

Farid Gahramanov

The plan for Case #2 was put together to see if modifying genetic operators may improve the algorithm's performance in complex situations by finding a better balance between exploring new possibilities and exploiting known good solutions. The experiment was made for to see whether modifying each genetic operator may lead to a more efficient search strategy that could adapt and find the best or almost best answers more quickly or was it better at optimization.

Implementation Details

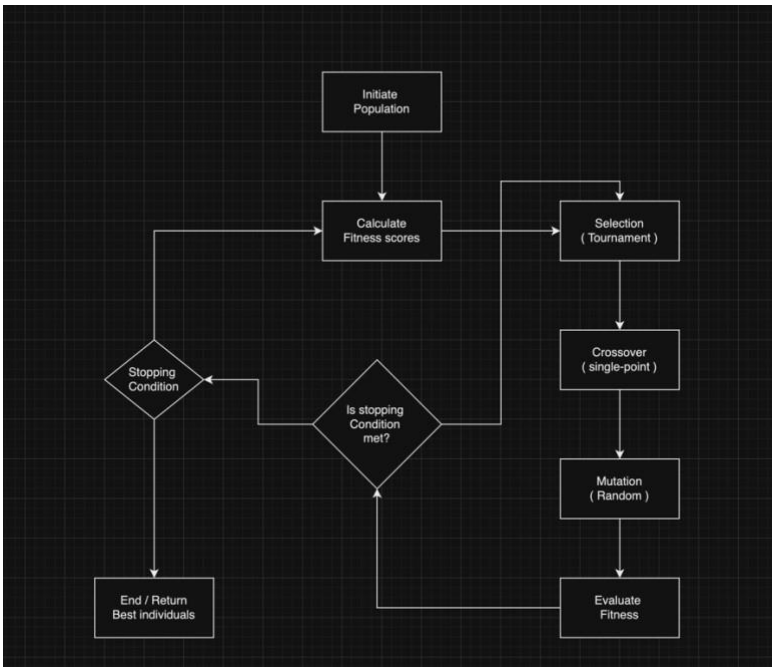
- Population Initialization:
  - The first step of the genetic algorithm is to initialize a population of feasible solutions, where each one is represented by an array of parameters. The uniform function in NumPy is used in our approach to produce the population by randomly creating starting parameter values within a given range (-10 to 10 for each parameter).
- Fitness Calculation:
  - Finding the mean squared error (MSE) between the real target values and the expected results allows us to assess the fitness of every member of the population. The process of doing this calculation is using the parameters of the individual in a prediction model and calculating how much these predictions deviate from the true values.
- Operator Implementation:
  - Selection (Tournament strategy):
    - Individuals from the population are selected at random to engage in small tournaments using this approach. The individual with the best performance in each group is chosen for reproduction, ensuring that those with higher fitness have a better chance of contributing to the next generation.

- Crossover (Single-point strategy):
  - During the crossover step a single crossover point is chosen randomly, and parts of two individuals' genetic codes are swapped to produce new offspring.
- Mutation (Random mutation strategy):
  - After crossover, there's a possibility that the new offspring might mutate. This step randomly changes some of the offspring's genetic traits, adding variety to the population and helping the algorithm search new areas of the solution space that haven't been explored yet.

✞ Stopping Condition:

- The algorithm stops when a stopping condition is met, which is in my code is set to by either reaching a maximum number of generations (100 generations) or achieving a fitness level below a predefined threshold (fitness < 0.01), indicating a satisfactory solution.

Flowchart



Performance Evaluation

Farid Gahramanov

For Case #1:

	Objective Value	**Elapsed Time (ms)**	**A**	**B**	**C**	**Seed**
Dataset1	**2.051**	**0.18**	**0.17**	**-1.54**	**-4.31**	**123**
Dataset2	**0.720**	**0.18**	**1.34**	**1.00**	**-2.12**	**123**
Dataset3	**4.551**	**0.21**	**-0.49**	**-1.54**	**5.64**	**123**
Dataset4	**0.480**	**0.19**	**1.99**	**0.33**	**-0.53**	**123**
Dataset5	**13.057**	**0.18**	**-2.25**	**-2.70**	**2.71**	**123**
Dataset1	**5.472**	**0.17**	**-1.06**	**0.47**	**5.22**	**457**
Dataset2	**13.028**	**0.17**	**-1.06**	**0.47**	**5.22**	**457**
Dataset3	**7.839**	**0.19**	**1.91**	**0.47**	**5.22**	**457**
Dataset4	**6.768**	**0.19**	**-0.24**	**0.47**	**4.92**	**457**
Dataset5	**1.248**	**0.18**	**-1.06**	**0.47**	**5.22**	**457**

In Case #1, the genetic algorithm was evaluated across multiple datasets that was given in the project with varying degrees of complexity and size. Here's the output for each dataset of the performance:

- Dataset 1 and Dataset 2 demonstrated good objective values, demonstrating an acceptable fit to the data with Dataset 1 obtaining an MSE of 2.051 and Dataset 2 achieving 0.720.
- Dataset 3 had a higher MSE at 4.551, suggesting some challenges in fitting the model to this particular dataset.
- Dataset 4 performed exceptionally well with the lowest MSE of 0.480, indicating an excellent fit.
- Dataset 5 was the most challenging, with an MSE of 13.057 indicating that the genetic parameters may have been unsuitable or overfit for this dataset.

Farid Gahramanov

For Case #2:

	Objective Value	**Elapsed Time (ms)**	**A**	**B**	**C**	**Seed**
Dataset1	**286.002**	**0.18**	**0.17**	**-1.54**	**-4.31**	**123**
Dataset2	**104.041**	**0.18**	**1.34**	**1.00**	**-2.12**	**123**
Dataset3	**252.132**	**0.21**	**-0.49**	**-1.54**	**5.64**	**123**
Dataset4	**190.997**	**0.19**	**1.99**	**0.33**	**-0.53**	**123**
Dataset5	**771.262**	**0.18**	**-2.25**	**-2.70**	**2.71**	**123**
Dataset1	**87.826**	**0.17**	**-1.06**	**0.47**	**5.22**	**457**
Dataset2	**540.253**	**0.17**	**-1.06**	**0.47**	**5.22**	**457**
Dataset3	**444.326**	**0.19**	**1.91**	**0.47**	**5.22**	**457**
Dataset4	**185.303**	**0.19**	**-0.24**	**0.47**	**4.92**	**457**
Dataset5	**293.505**	**0.18**	**-1.06**	**0.47**	**5.22**	**457**

Case #2 was implemented to see the and observe their impact on performance:

- Dataset 1 and Dataset 2 showed a significant increase in MSE, with values of 286.002 and 104.041, this tells us that compared to Case #1 it has a poor performance.
- Dataset 3 also showed an increase in MSE to 252.132.
- Dataset 4 had an MSE of 190.997, which is lower than some other datasets in Case #2, still represents a significant degradation from Case #1.
- Dataset 5 had an MSE of 771.262, the highest across all datasets and cases, showing us that there are significant challenges with the dataset under the modified genetic operations.

Farid Gahramanov

Comparison of Case #1 and Case #2

Looking at our outputs from Case#1 and Case#2 we can defiantly say that Case #1 consistently outperformed Case #2 across all datasets. This outputs suggests that the original genetic operators (tournament selection, single-point crossover, and random mutation) were more effective and better suited to these particular datasets. Performance did not increase as a result of the changes made in Case #2, which included adjustments to the mutation rate and strategies. As a matter of fact, they caused performance to drop.