**LABORATORY 3**

**GROUP 5**

By Uhma Pawel and Maurizio Andrés Carrasquero

**INTRODUCTION**

In this laboratory, we implemented a **genetic algorithm** to optimize the **Rastrigin function**, a common test function in optimization that is known for its large search space and numerous local minima. The goal was to use **Rank Selection** for parent selection and apply the algorithm to find the global minimum of the Rastrigin function:



The Rastrigin function is typically used to evaluate the performance of optimization algorithms. It has a global minimum at (x,y)=(0,0)(x, y) = (0, 0)(x,y)=(0,0), where f(0,0)=0f(0,0) = 0f(0,0)=0. The complexity arises from its many local minima, making it a good benchmark for genetic algorithms.

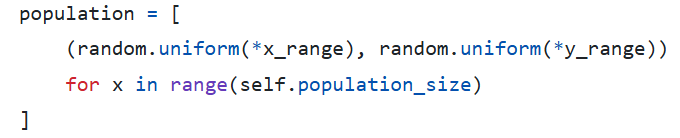
The algorithm used Gaussian mutation and random interpolation-based crossover, applying both operators to only a portion of the population while preserving the rest. The optimization was run on populations initialized with x,y∈[−5,5]x, y \in [-5, 5]x,y∈[−5,5].

**IMPLEMENTATION**

The genetic algorithm was implemented in Python, using a modular structure in genetic\_algorithm.py. The algorithm follows the classic structure: initialize → evaluate → select → crossover → mutate → replace.

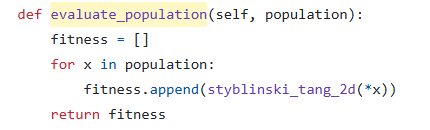
Below is a breakdown of the key components with relevant code snippets and explanations.

1. **Population Initialization:** The population is initialized randomly within the specified range (in this case, [−5,5][-5, 5][−5,5] for both x and y):

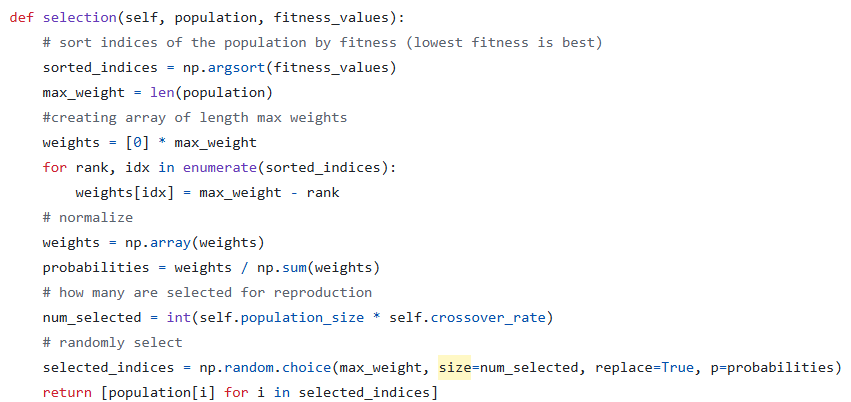


This generates a list of individuals (2D vectors) with uniformly distributed random values.

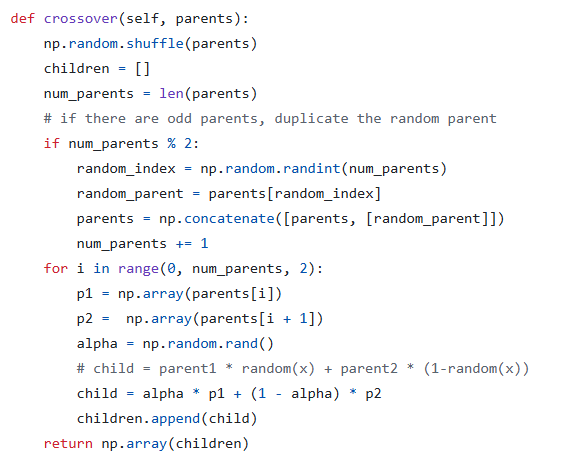
### **Evaluate the Population:** Each individual's fitness is determined by the function value (lower is better).



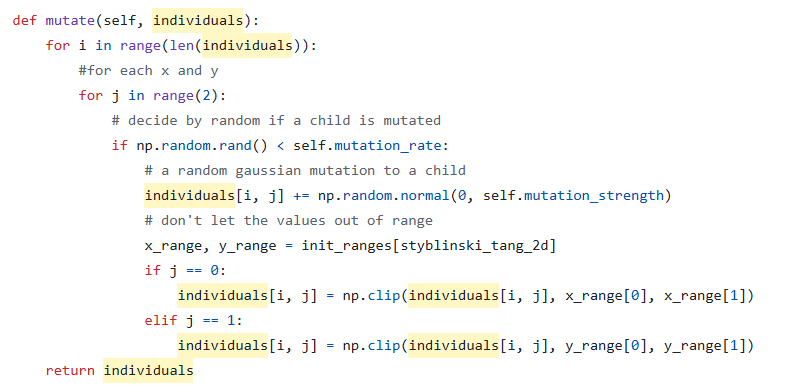
1. **Selection:** We use fitness-proportional selection, where better individuals (lower values) get higher selection weights.



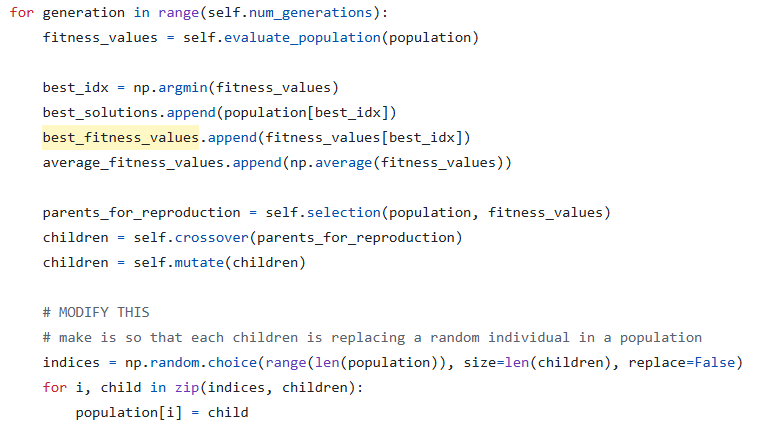
1. **Crossover:** Pair parents to create children by blending their coordinates using a random mixing factor α.



1. **Mutation:** With a certain chance, we randomly tweak each coordinate with Gaussian noise.



1. **Evolution loop:** Each generation, we: Evaluate current population, Select parents, Generate and mutate children, Replace part of the population with children



### **Conclusion**

In this lab, we successfully implemented a genetic algorithm to optimize the Rastrigin function using Rank Selection. We explored the effects of algorithm parameters such as mutation and crossover rates, showing how they influence convergence. The experiments demonstrated that genetic algorithms are effective for complex optimization problems, although sensitive to parameter tuning.

Challenges included balancing exploration and exploitation and handling the randomness of the process. Future work could explore hybrid methods or adaptive parameter adjustment for better stability and convergence.