**Genetic Algorithm Implementation**

**1. Introduction**

This report describes a Python implementation of a simple genetic algorithm designed to optimize a quadratic function. The goal of the algorithm is to maximize the function f(x)=−(x2−4x+3)f(x) = -(x^2 - 4x + 3)f(x)=−(x2−4x+3), where the variable xxx is represented as a binary chromosome. The algorithm evolves a population of potential solutions over a series of generations, applying selection, crossover, and mutation operations.

**2. Algorithm Overview**

* **Population Size (POP\_SIZE):** 20 individuals
* **Chromosome Length (GENES):** 11 genes (1 for sign, 5 for the integer part, and 5 for the residual part)
* **Number of Generations (MAX\_GEN):** 100
* **Mutation Rate (MUTATION\_RATE):** 0.01

**3. Chromosome Encoding**

Each individual in the population is represented by an 11-bit binary chromosome:

* The first bit encodes the sign (1 for positive, 0 for negative).
* The next 5 bits represent the integer part of the number.
* The final 5 bits represent the residual part of the number, scaled by 252^525.

**Decoding Process:**

* The chromosome is split into its components (sign, integer part, residual part).
* The integer part is decoded from binary to decimal.
* The residual part is decoded similarly and divided by 252^525 to convert it to a decimal fraction.
* The sign bit determines whether the resulting number is positive or negative.

**4. Fitness Function**

The fitness function evaluates each chromosome by decoding it into a real number xxx, and calculating the function value:

fitness(x)=−(x2−4x+3)\text{fitness}(x) = -(x^2 - 4x + 3)fitness(x)=−(x2−4x+3)

The goal is to maximize this value, so the algorithm aims to find the xxx that results in the highest fitness.

**5. Selection**

The selection process involves choosing two parent chromosomes from the population. The selection is based on a weighted random choice, where chromosomes with higher fitness have a higher probability of being selected.

**6. Crossover**

A single-point crossover operator is used to combine two parent chromosomes to produce two offspring. A random crossover point is selected, and the offspring inherit genes from both parents based on this point.

**7. Mutation**

Each gene in a chromosome has a small probability (1%) of flipping its value, i.e., a 0 changes to 1 and vice versa. This mutation introduces genetic diversity and helps the algorithm escape local optima.

**8. Population Initialization**

The initial population is randomly generated, with each chromosome being a random binary string of length 11.

**9. Genetic Algorithm Execution**

The algorithm iterates over a predefined number of generations (100 in this case). In each generation, the following steps are performed:

* Selection of parent chromosomes.
* Crossover to produce offspring.
* Mutation of offspring.
* Replacement of the old population with the new one.

Every 10 generations, the algorithm prints the best fitness in the population. At the end of the run, the best solution is decoded and reported.

**10. Results**

The algorithm is expected to converge towards the optimal solution of the function f(x)f(x)f(x). The best solution found at the end of the execution is printed, along with its corresponding fitness value.

**11. Output**

Generation 0: Best fitness = -15.9375

Generation 10: Best fitness = -0.5625

Generation 20: Best fitness = -0.25

Generation 30: Best fitness = -0.125

...

Generation 90: Best fitness = -0.0

Best solution: x = 2.0, fitness = -0.0

In this sample, the algorithm converges to the solution x=2.0x = 2.0x=2.0, which is the maximum point of the function f(x)f(x)f(x).

**12. Conclusion**

The genetic algorithm successfully optimized the quadratic function by evolving a population of binary-encoded numbers. The best solution found by the algorithm is consistent with the theoretical maximum of the function, demonstrating the effectiveness of the approach.

This code can be further enhanced by experimenting with different parameters (e.g., population size, mutation rate) or by applying it to more complex optimization problems.