Benchmark Optimization with Genetic Algorithms

Mihai-Cristian Farcaș

May 2025

1 Introduction

This project explores the application of Genetic Algorithms (GAs) for finding the minima of benchmark optimization functions. It implements two multimodal benchmark functions in Python, optimizes them using different GA configurations, and conducts a statistical analysis of the performance results.

2 Function Selection

Two multimodal benchmark functions were selected for this study:

2.1 Rastrigin Function

Domain: $[-5.12, 5.12] \times [-5.12, 5.12]$

Formula:

$$f(x, y) = 20 + x^2 + y^2 - 10(\cos(2\pi x) + \cos(2\pi y))$$

The Rastrigin function is highly multimodal, with many local minima arranged in a regular grid-like manner. It has a global minimum of 0 at $(0, 0)$.

2.2 Ackley Function

Domain: $[-5, 5] \times [-5, 5]$

Formula:

$$f(x, y) = -20 \cdot \exp\left(-0.2 \cdot \sqrt{0.5(x^2 + y^2)}\right) - \exp\left(0.5(\cos(2\pi x) + \cos(2\pi y))\right) + e + 20$$

The Ackley function combines exponential terms with cosine modulation, creating a surface with many local minima and a narrow global minimum at $(0, 0)$.

3 Genetic Algorithm Implementation

3.1 Overview

The GA implementation is modular and configurable:

- **Encoding**: Binary and real-valued
- **Selection**: Tournament
- **Crossover**: Several types depending on encoding
- **Mutation**: Bit-flipping or Gaussian perturbation
- **Elitism**: Preserving top individuals

3.2 Encoding Options

3.2.1 Binary Encoding

Each variable is encoded with 16 bits and decoded to a real value in the function domain.

3.2.2 Real-Valued Encoding

Individuals are represented directly by $[x, y]$ vectors, enabling more precise solutions.

3.3 Crossover Types

3.3.1 Binary Encoding Crossovers

- **One-Point Crossover**
- **Two-Point Crossover**

3.3.2 Real-Valued Encoding Crossovers

- **Arithmetic Crossover**
- **BLX-α Crossover**

3.4 Mutation Operators

- **Binary**: Bit flip with a given probability
- **Real-valued**: Add Gaussian noise scaled to domain

3.5 Selection and Elitism

Tournament selection is used, and the best individuals are preserved using elitism.

4 Experimental Setup

4.1 Parameters

- Population size: 100
- Crossover rate: 0.8
- Mutation rate: 0.1

- Elite size: 2
- Runs: 30 per configuration
- Evaluations: 10,000 per run

4.2 Experimental Configurations

Combinations tested:

- Functions: Rastrigin, Ackley
- Encodings: Binary, Real-valued
- Crossovers: One-point, Two-point (Binary); Arithmetic, BLX-α (Real)

5 Results and Analysis

5.1 Performance Metrics

- Best Fitness
- Convergence Generation
- Best Solution Coordinates

5.2 Statistical Comparison

T-tests were conducted to compare:

- Binary vs. Real-valued encoding
- Crossover types within each encoding

5.3 Expected Results

1. Real-valued encoding is likely better for continuous functions.
2. Two-point crossover may outperform one-point for Binary.
3. BLX-α may surpass arithmetic crossover.
4. Rastrigin is expected to be harder than Ackley.

6 Code Structure

6.1 `functions.py`

Benchmark and visualization functions.

6.2 `genetic_algorithm.py`

Core GA logic and utilities.

6.3 `analysis.py`

Experimental runners, statistical tests, and plots.

6.4 `main.py`

Main script to define and run all components.

7 How to Run the Code

1. Install dependencies:

```
pip install numpy matplotlib scipy pandas seaborn
```

2. Run the script:

```
python main.py
```

3. Outputs: visualizations, statistical analysis, and results.

8 Conclusion

This study illustrates the effectiveness of GAs on multimodal benchmark functions. Through comparative analysis, we assess how encoding and crossover strategies affect performance.

9 References

1. Handbook of Test Problems in Local and Global Optimization
2. Holland, J. H. (1975). *Adaptation in natural and artificial systems*
3. Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*
4. Mitchell, M. (1998). *An introduction to genetic algorithms*