Assignment 2: Python Code for BFGS and GA

AE413: Optimization techniques in engineering

Gaurav Gupta, SC21B026

1 Overview

This report discusses the implementation and testing of two optimization algorithms: **BFGS (Broyden–Fletcher–Goldfarb–Shanno)** and **Genetic Algorithm (GA)** in python. Both algorithms serve different optimization needs, with BFGS being suitable for smooth, differentiable functions and GA being more flexible for complex, non-linear, and non-differentiable problems.

BFGS Algorithm

BFGS is a quasi-Newton method used to find local minima of smooth functions. It approximates the Hessian matrix to iteratively update the search direction, making it efficient for problems where derivatives are available and relatively inexpensive to compute. The BFGS method is particularly well-suited for smooth, unimodal functions and is widely used in various scientific and engineering applications due to its convergence properties and computational efficiency.

Genetic Algorithm (GA)

GA is an evolutionary algorithm inspired by the principles of natural selection and genetics. This implementation includes:

- **Elitism-based selection**: Ensures the fittest individuals are carried over to the next generation.
- **Simulated Binary Crossover (SBX)**: Combines pairs of parents to produce offspring with a controlled level of diversity.
- **Normally Disturbed Mutation**: Introduces small variations in offspring to enhance exploration of the search space.

GA is particularly effective for global optimization, where the objective function may be non-linear, multi-modal, or non-differentiable.

2 Benchmark Functions

Two benchmark functions were used to evaluate the performance of BFGS and GA:

- **Bohachevsky Function**:

$$f(x, y) = x^2 + 2y^2 - 0.3\cos(3\pi x) - 0.4\cos(4\pi y) + 0.7$$

  This unimodal function tests the algorithms' ability to converge to a global minimum in a smooth landscape.

- **Ackley Function**:

$$f(x, y) = -20\exp\left(-0.2\sqrt{0.5(x^2 + y^2)}\right) - \exp\left(0.5(\cos(2\pi x) + \cos(2\pi y))\right) + 20 + \exp(1)$$

  Known for its numerous local minima, the Ackley function challenges the algorithms with a rugged, multimodal landscape.

3 Results

3.1 Comparison between BFGS and GA

Table 1 and **Table 2** present the results from the Python implementation of BFGS and GA alongside the in-built MATLAB function of *fminunc* and *ga*.

- **The results of BFGS is highly dependent on the initial point of search.**
  For example, in case of the Ackley Multimodal benchmark the algorithm converges to a local minima when started from $(-5, 5)$ whereas converges to a global minima when started from $(-0.1, 0.1)$.

- **GA is always converges in the neighbourhood of global minima.**
  In case of Ackley multimodal benchmark, the algorithm coverges to the same point which is very close to the global minima of $(0, 0)$.

- The computational time is comparable for both algorithms in case of MATLAB functions whereas in current Python implementation, GA is slower than BFGS.

Benchmark Case	Point of Minima		Computation Time (s)	
	BFGS	GA	BFGS	GA
Bohachevsky (Unimodal)	(0,0)	(0.133,-0.044)	0.236	134.72
Ackley (Local Minima)	(-4.986, 4.986)	(0.0484, -0.0418)	0.239	445.23
Ackley (Global Minima)	(0, 0)	(0.0484, -0.0418)	0.549	445.23

Tab. 1: Results from Python implementation of BFGS and GA

Benchmark Case	Point of Minima		Computation Time (s)	
	BFGS	GA	BFGS	GA
Bohachevsky (Unimodal)	(0.618, 0)	(-0.007, -0.020)	0.132	3.013
Ackley (Local Minima)	(-4.986, 4.986)	(-0.020,0.010)	0.0913	0.0583
Ackley (Global Minima)	(0, 0)	(-0.020,0.010)	0.0103	0.0583

Tab. 2: Results from MATLAB using *fminunc* (BFGS) and *ga* functions.

3.2 Comparison between Python implementation and MATLAB

- Python implementation of BFGS and *fminunc* perform almost equivalent. The solution is exact in all cases except for Unimodal case using *fminunc*. Although, the *fminunc* is slightly faster than Python implementation for all the cases.
- Python implementation of GA and *ga* yield similar results but the MATLAB function is exceptionally fast as compared to the Python implementation. The *ga* allows use of different kinds of crossover and mutation methods as compared to the fixed type of crossover and mutation in our code.

4 Conclusion

Based on the results from both the Python implementation and MATLAB functions, it is clear that BFGS works well for unimodal functions since it reliably converges to the global minimum. On the other hand, GA is better suited for complex multimodal functions due to its ability to explore and handle multiple minima effectively.

5 Code Availability

The code for the BFGS and Genetic Algorithm implementations, including tests on the Bohachevsky and Ackley functions, is available on GitHub at: `https://github.com/airwarriorg91/Optimization-Techniques/tree/main/Assignment-2`