CL603

Course Project

Team

Optimus

Optimization using Real Valued Genetic Algorithm

Atharva Kulkarni 210070047
Ayush Raj 210100037
Faizan Ansari 210100058
Rishit Shrivastava 210050136

Table of Contents

I Introduction

While gradient based algorithms utilize the nature of the objective function, they have the following limitations:

- Computationally demanding.
- Not applicable to discontinuous, non-differentiable and discrete valued functions.
- Solution may get stuck at local minima, or become slow to obtain due to saddle points.
- Slow convergence.

- To overcome these issues, John Holland and his collaborators in the 1960s and 1970s devised Genetic Algorithm, by taking inspiration from the famous biological evolution theory of Charles Darwin.
- This algorithm imitates biology and finds the optimum solution using the concepts of selection of fittest individuals, their reproduction, and mutation in the process to generate a better new generation.
- This can also be used for constrained optimization by simply replacing the infeasible points in the search space by feasible ones.
- On an average, this method provides faster convergence than derivative based algorithms while providing nearly equal accuracy.

II Details of the algorithm

Brief explanation of how Real valued genetic algorithm works:

1. **Devise a representation:** Design a representation for the candidate solutions in the form of chromosomes that can be used to encode the values of the problem's parameters.
2. **Define fitness function:** Define a fitness function that evaluates the quality of a given chromosome. This function is problem-specific and should reflect the objective of the optimization problem. For our need we defined it to be reciprocal of the function value to minimise the objective function.
3. **Initialization:** Create an initial population of candidate solutions by generating a set of chromosomes randomly.

4. **Selection**: Select the top n chromosomes based on their fitness values, where n is a predetermined fraction of the population.

5.**Mating**: Create offspring by combining the genes of two parent chromosomes. The mating process is circular, so that the last chromosome is mated with the first. This ensures that there are an equal number of offspring as parents. The mating is done such that the some fraction of the genes are copied from the father and other fraction are given to the offspring from mother, the parameter for this fraction is decided in the beginning.

6. **Mutation**: Introduce random changes to the offspring chromosomes by flipping the value of a gene with a fixed probability.

7.**Replacement**: Append the offspring to the parent population and create a new generation of chromosomes.

8.Termination: Repeat the selection, mating, mutation, and replacement steps until a satisfactory fitness level is achieved or the number of iterations has reached a predetermined value.

9 Output: The algorithm outputs the best chromosome found in the final generation, which represents the optimal solution to the optimization problem.

Overall, the genetic algorithm works by iteratively creating new populations of candidate solutions and selecting the fittest individuals to mate and produce offspring that are then mutated and inserted into the next generation. Over time, the population evolves to contain fitter and fitter individuals until the optimal solution is found.

III Literature Survey

- **Singiresu S. Rao; Engineering Optimization: Theory and Practice, John Wiley & Sons, Inc. 2009, 4th edition**
- **Belegundu A. and T. Chandrupatla Optimization Concepts and Applications in Engineering, Prentice Hall, 1999.**
  The books above explain the need of genetic algorithm, intuition behind it and the algorithm in detail.

- **Genetic Algorithms for Use in Financial Problems**
  This paper explores the application of real valued genetic algorithm in the domain of financial optimization problems of asset allocation using the traditional mean variance approach and secondly using a direct utility maximization method for a step utility function. It also compares performance of genetic algorithm with Newton's method.

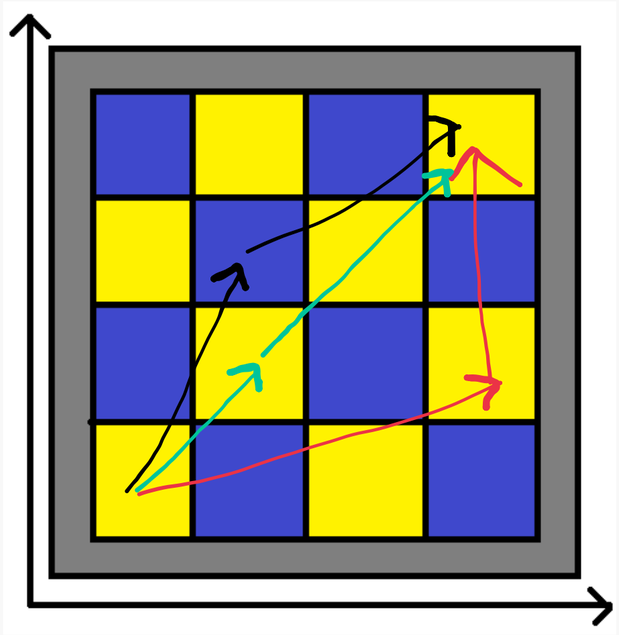
III Literature Survey

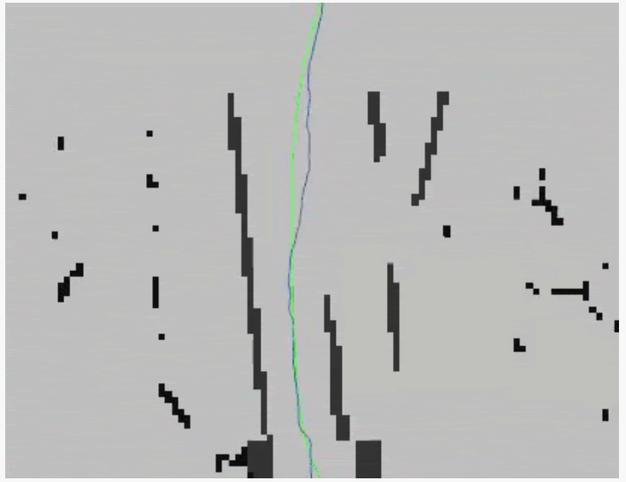
- **John D. Hedengren; ME575/CE575: Optimization Techniques in Engineering, BYU Prism Group, Chapter 6, 2018, Edition 2**
  This book explains genetic algorithm on discrete as well as real design variables, and provides examples for the same. Various modifications in genetic algorithm such as different ways of crossover (single point, uniform, blend crossover, etc.), mutation(uniform, dynamic) are given. This algorithm can be run using parallel computing to get higher speeds, and thus can be deployed in real time optimization. Real valued genetic algorithm can be extended to multi-objective optimization. Genetic algorithm can generate good designs called Pareto designs (nondominated designs from a given set of designs) which can be said to be an optimum tradeoff between multiple objective functions.

III Literature Survey

- **Manikas, Theodore & Ashenayi, K. & Wainwright, R.L.. (2008). Genetic algorithms for autonomous robot navigation. Instrumentation & Measurement Magazine, IEEE. 10. 26 - 31. 10.1109/MIM.2007.4428579.**

  To obtain information about specific environments, measurements of various parameters are needed. This task cannot be done by humans in hostile environments such as nuclear sites, disaster struck regions, etc. Autonomous robots can be used to survey these environments. This paper talks about implementing motion planning algorithms on such robots using genetic algorithm for optimal local path planning.





IV Explanation of code

```python
from numpy import random
import numpy as np
import matplotlib.pyplot as plt
import math
```

Importing the required libraries

```python
#test function
def func(x,y):
    return 32*x**2 + 21*y**4 + 12
```

Defining the objective function

```python
#some parameters which will be used later
best_gene=[]
best_gene_fitness=[]
max_gen=10000
fitness_threshold=600000
mutation_prob=0.4

spread=10 #will be used for generating 1st generation and further for generating random number for mutation
```

Initializing the required parameters

```python
def fitness(x,y):
    if func(x,y)==0:
        result = 9999999
    else:
        result = abs(1/func(x,y))
    return result
```

We define a function so that a smaller value of the function yields a higher fitness value, and to counter the function becoming zero, we arbitrarily assign a very large fitness value (assuming that the function we use minimizes to zero)

```python
n_chromosomes = 16
n_genes = 2 #basically the number of variables
size = (n_chromosomes,n_genes)
gen = 1
generation=random.uniform(-spread,spread,size)
print(f"The 1st generation is as follows:")

for i in range(n_chromosomes):
    print(generation[i])
```

As our function is defined in two variables we are randomly generating the initiall chromosomes
with 2 genes. Here we have restricted the value of genes between -10 and +10

```
ranked_generation = []
for i in generation:
    ranked_generation.append( (fitness(i[0],i[1]), i) )
ranked_generation.sort(reverse=True)

best_gene.append(ranked_generation[0][1])
best_gene_fitness.append(ranked_generation[0][0])

print(f"The 1st generation after sorting by their fitness is as follows:")
for i in range(n_chromosomes):
    print(f"Rank {i+1}: {ranked_generation[i][1]} , fitness = {ranked_generation[i][0]}")
```

Here we are sorting the current generation with respect to the fitness value, while keeping the chromosome with highest fitness at the top.

```
parents = ranked_generation[:int(n_chromosomes/2)]
parent_chromosomes=[]

print(f"The selected parent chromosomes are as follows:")

k=0
for i in parents:
    parent_chromosomes.append(list(i[1]))
    print(parent_chromosomes[k])
    k+=1
```

Here we are selecting the top n/2 fitter generation and are discarding the bottom half of the chromosomes.

```python
def mating(parents,crossover_point):
    offsprings=[]

    #we will be mating parent i with i+1 and we will go through this in a circular manner, so that in the end,
    #last index parent is mated with 0th
    for i in range(len(parents)-1):
        child_chromosome=list(parents[i][:crossover_point])+list(parents[i+1][crossover_point:])
        offsprings.append(child_chromosome)

    child_chromosome=list(parents[len(parents)-1][:crossover_point])+list(parents[0][crossover_point:])
    offsprings.append(child_chromosome)

    return offsprings
```

We are using the crossover method for mating in which the genes before the cross over point are taken from first parent and the remaining are taken from the second parent . Here mating is done in an circular manner where 1st chromosome is mated with the 2nd, 2nd with the third and so on till the last chromosome is again mated with the first. This algorithm ensures that the number of parents are equal to offsprings.

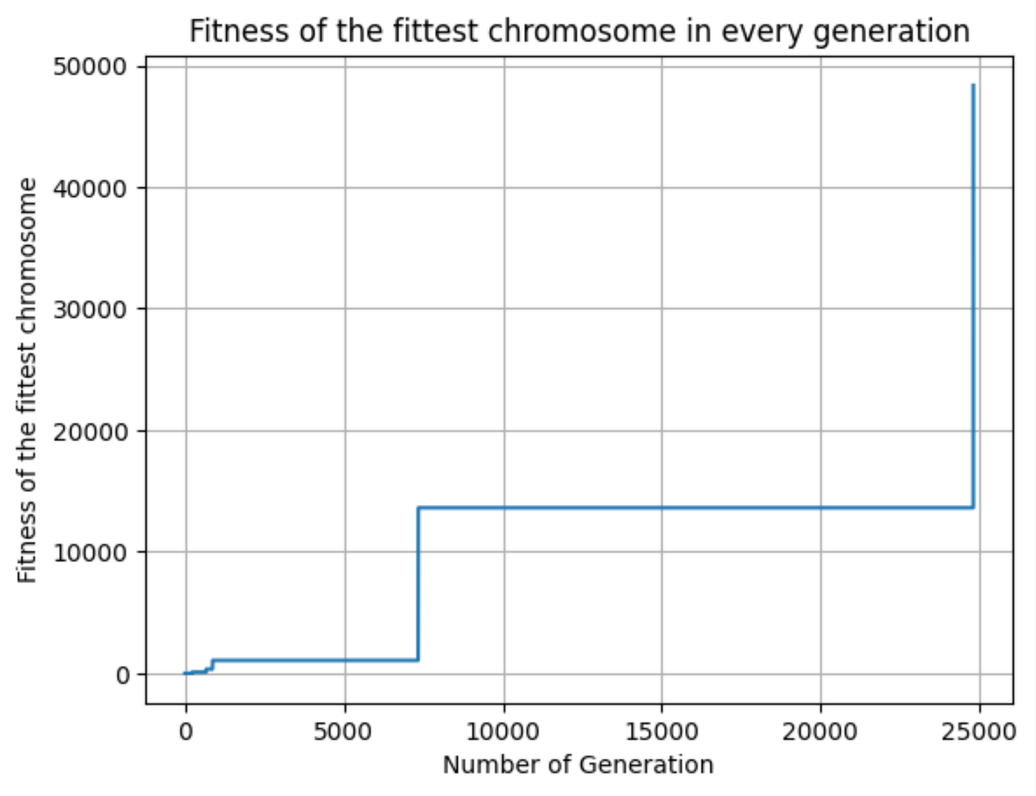
```python
print(f"Mutated offsprings are as follows:")
for i in offsprings:
    chance = random.rand()
    if chance<mutation_prob:
        index=random.randint(0,n_genes)
        i[index]=random.uniform(-spread,spread)
    print(i)

generation=parent_chromosomes+offsprings
gen+=1
```

Mutation is done only when chance is less than the predefined mutation probabililty and is done by changing the value of a random gene to a random number between -10 and +10.

This process is repeated on and on untill convergence is achieved or the maximum number of iterations are done.

Fitness of the fittest chromosome in every generation



V Conclusion & Remarks

- Since the real valued genetic algorithm mimics natural selection, it is an interesting and intuitive way of optimization.
- Genetic algorithm is a heuristic, which is cheap and efficient to compute, and guarantees optimality in most cases.
- This algorithm is robust to discontinuities and non-differentiabilities and does not need much preconditioning compared to other problems.
- The tunable parameters are maximum generations (limited by computation time), fitness threshold (decided by desired accuracy) and mutation probability (higher probability enables more exploration).

Thank you for listening!

Team Optimus