Cmsc 191 Exercise 6: Looking for the Global Minimum of 10 Standard Test Functions Using the Genetic Algorithm and the Simulated Annealing Algorithm

Gimel David F. Velasco

September 29, 2016

Abstract

This paper demonstrates a simple implementation of the Genetic Algorithm and the Simulated Annealing in solving the global minimum for 10 continuous n-variabled standard test functions.

1 Introduction

The Genetic Algorithm is an algorithm made by Holland on 1975. The Algorithm is inspired by the concept of Genetics and Natural Selection. It approaches a particular problem by looking for the fittest individual and considers it as the solution to the problem. It approaches a particular problem by looking for the fittest individual and considers it as the solution to the problem. The Simulated Annealing is another type of Heuristics which is used for solving complex problems. This algorithm is of the analogy that if a liquid material cools slowly, the system will then get into a state of minimum energy optimally. On the onther hand, if the liquid material cools too quickly, it will have a sub-optimal configuration. Also, For these standard test functions, the Genetic Algorithm and Simulated Annealing Algorithm will solve or get close to their respective global minimum.

2 10 Test Functions

The algorithm will seek the global minimum of the following standard test functions [1]:
Problem 1: De Jong's Function ($|x| \leq 5.12; n = 5, 10$)

$$f(x) = \sum_{i=1}^{n} x_i^2 \tag{1}$$

Global Minimum: $f(x) = 0$
Problem 2: Axis Parallel Hyper-Ellipsoid Function ($|x| \leq 5.12; n = 5, 10$)

$$f(x) = \sum_{i=1}^{n} (i * x_i^2) \tag{2}$$

Global Minimum: $f(x) = 0$
Problem 3: Rotated Hyper-Ellipsoid Function ($|x| \leq 65.536; n = 5, 10$)

$$f(x) = \sum_{i=1}^{n} \sum_{j=1}^{i} x_j^2 \tag{3}$$

Global Minimum: $f(x) = 0$
Problem 4: Rastrigin's Function ($|x| \leq 5.12; n = 5, 10$)

$$f(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i)] \tag{4}$$

Global Minimum: $f(x) = 0$
Problem 5: Griewangk's Function ($|x| \leq 600; n = 5, 10$)

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{5}$$

Global Minimum: $f(x) = 0$
Problem 6: Ackley's Function ($|x| \leq 32.768; n = 5, 10$)

$$f(x) = -a * exp(-b * \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(cx_i)) + a + exp(1) \tag{6}$$

Global Minimum: $f(x) = 0$
where $a = 20, b = 0.2, c = 2\pi$

Problem 7: Branin's Function ($|x| \leq 12.5; n = 2$)

$$f(x_1, x_2) = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1 - f)cos(x_1) + e \tag{7}$$

Global Minimum: $f(x_1, x_2) = 0.397887$
where $a = 1, b = \frac{5.1}{4\pi^2}, c = \frac{5}{\pi}, d = 6, e = 10, f = \frac{1}{8\pi}$

Problem 8: Easom's Function ($|x| \leq 100; n = 2$)

$$f(x_1, x_2) = -cos(x_1)cos(x_2)exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2) \tag{8}$$

Global Minimum: $f(x_1, x_2) = -1$
Problem 9: Six-Hump Camel Back Function ($|x| \leq 3; n = 2$)

$$f(x_1, x_2) = (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2 \tag{9}$$

Global Minimum: $f(x_1, x_2) = -1.0316$
Problem 10: Shubert's Function ($|x| \leq 5.12; n = 2$)

$$f(x_1, x_2) = -\sum_{i=1}^{5} icos((i + 1)x_1 + 1)\sum_{i=1}^{5} icos((i + 1)x_2 + 1) \tag{10}$$

Global Minimum: $f(x) = -186.7309$

3 Methodology

The fitness function or objective function of which is defined as:

$$OBJ(x) = |f(x) - globalminimum| \tag{11}$$

A Minimization Problem is implemented in the algorithm wherein the objective of the algorithm is to get very close if not equal to zero.

3.1 The Genetic Algorithm

The Genetic Algorithm implemented in this program is inspired by how the Genetic Algorithm is implemented in Hermawanto's article on solving a 4-tuple function [2]. The pseudocode of the Genetic Algorithm implemented in the program is shown below:

```
1. Determine the number of chromosomes, population, generation, mutation rate and crossover rate value
2. Generate first generation of chromosomes
3. Tournament Selection of the quarter of the population
3. Crossover of the selected parents based on the crossover rate
4. Mutation of part of the generation according to the mutation rate
```

6. Evaluation of fitness value of chromosomes by using the fitness function
7. Repeat steps 2-6 until the maximum number of generations is met or until desired fitness is reached
8. Solution (Best Chromosome)

A modification in the code was made. The algorithm tends stick to the fittest chromosome and mutates less randomly when the chromosomes get fitter and fitter. This is done so that the algorithm would be able to be much more exact and precise in looking for the global minimum.

3.2 The Simulated Annealing Algorithm

The Simulated Annealing Algorithm implemented in this program is based on how the algorithm is explained in the presentation made by Oliver de Weck, Ph.D. [3]. The pseudocode of the Simulated Annealing Algorithm is shown below:
1. Choose a random Xi, select the initial system temperature, and specify the cooling (i.e. annealing) schedule
2. Evaluate E(Xi) using a simulation model
3. Perturb Xi to obtain a neighboring Design Vector (Xn)
4. Evaluate E(Xn) using a simulation model
5. If $E(Xn) < E(Xi)$, Xn is the new current solution
6. If $E(Xn) > E(Xi)$, then accept Xn as the new current solution with a probability $e^{-D/T}$ where $D = E(Xn) - E(Xi)$
7. Reduce the system temperature according to the cooling schedule $T_{k+1} = ratio^k * T_k$
8. Terminate the algorithm
where X is the Design Vector, E is the System Energy (i.e. Objective Function), T is the System Temperature and D is the Difference in System Energy Between Two Design Vectors. Also, a modification in the code was made. The algorithm tends stick to the temporary solution Xi and mutates less randomly as the system cools. This is done so that the algorithm would be able to be much more exact and precise in looking for the global minimum.

4 Results and Discussion

After 10 runs of each test function, the number of successes are then counted below:

Problem 1: De Jong's Function

Function 1	GA Score	aveg.runtime	SA Score	aveg.runtime
n=5	10	6.8s	10	4.4s
n=10	10	23s	10	7s

Problem 2: Axis Parallel Hyper-Ellipsoid Function

Function 2	GA Score	aveg.runtime	SA Score	aveg.runtime
n=5	10	5.5s	10	6.9s
n=10	10	44s	10	7.5s

Problem 3: Rotated Hyper-Ellipsoid Function

Function 3	GA Score	aveg.runtime	SA Score	aveg.runtime
n=5	10	21s	10	8.5s
n=10	10	250s	10	10s

Problem 4: Rastrigin's Function

Function 4	GA Score	aveg.runtime	SA Score	aveg.runtime
n=5	10	18s	10	8.5s
n=10	10	120s	10	9s

Problem 5: Griewangk's Function

Function 5	GA Score	aveg.runtime	SA Score	aveg.runtime
n=5	10	250s	10	8s
n=10	10	200s	10	9.5s

Problem 6: Ackley's Function

Function 6	GA Score	aveg.runtime	SA Score	aveg.runtime
n=5	10	111s	10	8.6s
n=10	10	590s	10	8.5s

Problem 7: Branin's Function

Function 7	GA Score	aveg.runtime	SA Score	aveg.runtime
n=2	10	60s	0	6.8s

Problem 8: Easom's Function

Function 8	GA Score	aveg.runtime	SA Score	aveg.runtime
n=2	10	200s	10	6.5s

Problem 9: Six-Hump Camel Back Function

Function 9	GA Score	aveg.runtime	SA Score	aveg.runtime
n=2	10	6s	0	6.6s

Problem 10: Shubert's Function

Function 10	GA Score	aveg.runtime	SA Score	aveg.runtime
n=2	0	n/a	10	7.7s

A run of the genetic algorithm is considered a failure when it reached its maximum number of generations. On the other hand, the simulated annealing is considered a failure when its objective function value is greater than 1. Considering the modification made to the algorithm, since as the population or system gets fitter and fitter, the chromosomes or solution tends to move a lot less. This can explain why there are some functions that the algorithm can no longer solve for the global minimum.

5 Conclusion

The Genetic Algorithm and the Simulated Annealing Algorithm implemented in the program is capable of getting close to the global minimum of the functions presented. Though some functions' global minimum are hard to get close to. The results have yielded a score of either perfect 10 or a straight 0. Though there were no instances that both of the algorithms have failed. If one algorithm is incapable, one on the other hand is capable of solving or getting close to the global minimum.

6 References

[1] Molga, M. and Smutnicki, C., "Test Functions for Optimization Needs", 2005

[2] Hermawanto, D., "Genetic Algorithm for Solving Simple Mathematical Equality Problem", Indonesian Institute of Sciences, n.d.

[3] de Weck O., "Simulated Annealing: A Basic Introduction", Massachusetts Institute of Technology,2010