# Finding the global minimum of standard benchmark functions using Genetic Algorithm

Dumitriu Oana-Florentina
Ionela Său

November 23, 2022

## 1 Abstract

This work aims to demonstrate the effectiveness of genetic algorithms in multimodal function solution. We compared our findings with those of the other heuristic search algorithms in order to undertake this research. In the "Methods" section we present the structure of the algorithm, describe all the functions we used, how we changed them to get better results and the values of all the parameters we used. In the "Experimental results" section are presented the comparisons between the Genetic Algorithm, Hill-Climbing Algorithm and Simulated Annealing. We can reach the conclusion that because genetic algorithms can compute favorable values in a decent amount of time, they can be used in the process of minimizing a function. Because it can search a larger space, the results are generally better to those of other heuristic algorithms used for functions with multiple minima.

## 2 Introduction

In computer science and operations research, a genetic algorithm is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the nest generations. By relying on biologically inspired operators like mutation, crossover, and selection, genetic algorithms are frequently employed to produce high-quality solutions to optimization and search problems.

The process of natural selection is performed by selecting the fittest individuals from a population, which can produce descendants that acquire the characteristics of their parents and replace them in the new generations. A genetic algorithm consists of five steps: initial population, fitness function, selection, crossover and mutation.

## 3 Methods

The whole population is represented with a vector of vectors of bits, that was generated at random. We applied Holland's roulette wheel selection, where each individual receives a number of descendants proportionate to their fitness divided by the population's fitness. For the improvement of the selection process we used elitism. In this way, we were able to choose the best individuals from the population and ensure their inclusion in the following generations. We used the fitness function to measure the chromosomal quality and make sure that the best candidates have a higher probability of being chosen. In addition, a selection pressure was utilized to determine the fitness of each individual.

Fitness function:
$f(x) = (\frac{maximum - g(x)}{maximum - minimum + 0.000001})^{selectionPreassure}$

For the mutation process, we randomly select a number for each bit of each individual from [0, 1], and if it's less than 0.01 we mutate it. In the crossover function, we gave each individual a random number between [0, 1], and we then ordered the individuals according to these numbers. Then, we couple the individuals two by two, but only if their given number is less than 0.6. We have a 50/50 probability of either forgetting it or picking the next individual if there are an odd number of individuals. We only use one randomly selected cutting point for the crossover process itself. Additionally, the descendants produced by this process are also replacing their parents in the current population.
Values used for this experiment are: sample size: 30, dimensions: 5, 10 and 30, precision: 5, population size: 150, number of generations: 1500, elitism percentage: 7%, selection pressure: 4, mutation probability 1%, crossover probability: 60%.

## 4 Functions

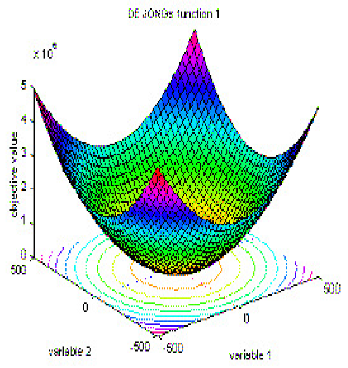

Figure 1: De Jong's function: $f_n(x) = \sum_{i=1}^{n} ix_i^2$
$-5.12 \leq x_i \leq -5.12$
global minimum: $f(x) = 0$

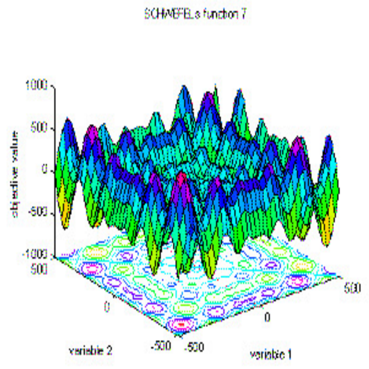

Figure 2: Schwefel's function: $f_n(x) = \sum_{i=1}^{n} -x_i sin(\sqrt{|x_i|})$
$-500 \leq x_i \leq 500$
global minimum: $f_n(x) = -n * 418.9829$

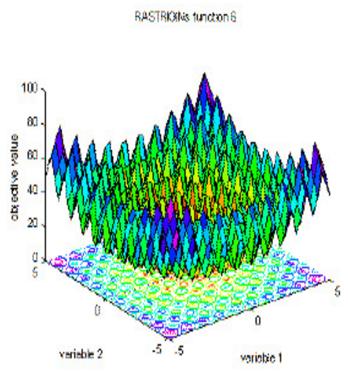

Figure 3: Rastrigin's function: $f_n(x) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$
$-5.12 \leq x_i \leq 5.12$
global minimum: $f(x) = 0$

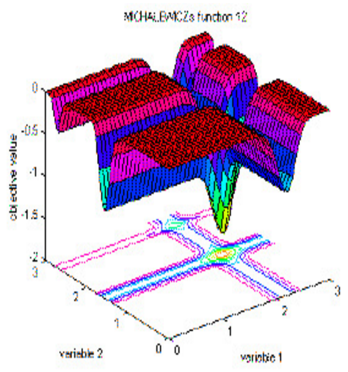

Figure 4: Michalewicz's function: $f_n(x) = -\sum_{i=1}^{n} sin(x_i)(sin\frac{ix_i^2}{\pi})^{20}$
$0 \leq x_i \leq \pi$
global minimum: $f_5(x) = -4.687, f_{10}(x) = -9.66$

## 5 Experimental results

| DeJong | | | | | |
|---|---|---|---|---|---|
| Dimension | Algorithm | Minim | Average | Deviation | Time |
| 30 | GA | 0.00024 | 0.00069 | 0.00020 | 331.691 |
| | HCB | 0 | 0 | 0 | 6839s |
| | HCF | 0 | 0 | 0 | 6543s |
| | SA | 0 | 0 | 0 | 28s |
| 10 | GA | 0 | 0 | 0 | 98.0363s |
| | HCB | 0 | 0 | 0 | 320s |
| | HCF | 0 | 0 | 0 | 216s |
| | SA | 0 | 0 | 0 | 15s |
| 5 | GA | 0 | 0 | 0 | 99.6859s |
| | HCB | 0 | 0 | 0 | 68s |
| | HCF | 0 | 0 | 0 | 46s |
| | SA | 0 | 0 | 0 | 12s |

| Rastrigin | | | | | |
|---|---|---|---|---|---|
| Dimension | Algorithm | Minim | Average | Deviation | Time |
| 30 | GA | 10.787 | 21.6776 | 6.699 | 338.115s |
| | HCB | 20.6256 | 28.53315 | 2.93313 | 5527s |
| | HCF | 30.8514 | 38.11441 | 3.4729 | 1110s |
| | SA | 17.8664 | 26.27384 | 6.00089 | 25s |
| 10 | GA | 0 | 1.18684 | 0.9753 | 128.242s |
| | HCB | 2.98993 | 4.72983 | 0.84127 | 254s |
| | HCF | 4.22575 | 6.02050 | 1.41440 | 202s |
| | SA | 2.04949 | 6.03797 | 2.02791 | 13 |
| 5 | GA | 0 | 0.04119 | 0.2218 | 100.497s |
| | HCB | 0 | 0.74636 | 0.45393 | 50s |
| | HCF | 0 | 1.28277 | 1.4144 | 41s |
| | SA | 0.00819 | 1.39702 | 0.7463 | 12s |

| Schwefel | | | | | |
|---|---|---|---|---|---|
| Dimension | Algorithm | Minim | Average | Deviation | Time |
| 30 | GA | -12533.2 | -12756.1 | 128.136 | 443.367s |
| | HCB | -11561 | -11245.6 | 149.6116 | 20113s |
| | HCF | -11027.7 | -10729.8 | 1941.254 | 18635s |
| | SA | -12533.7 | -12238.3 | 168.4887 | 31s |
| 10 | GA | -4189.72 | -4329.09 | 0.13486 | 165.453s |
| | HCB | -4155.59 | -4031.6 | 71.52654 | 1185s |
| | HCF | -4023.95 | -3861.95 | 68.35787 | 822s |
| | SA | -4189.52 | -4161.17 | 40.48079 | 15s |
| 5 | GA | -2094.91 | -2164.64 | 0.06402 | 95.675s |
| | HCB | -2094.91 | -2094.76 | 0.10903 | 184s |
| | HCF | -2094.81 | -2041.3 | 41.80897 | 117s |
| | SA | -2094.91 | -2093.54 | 2.25694 | 11s |

| Michalewicz | | | | | |
|---|---|---|---|---|---|
| Dimension | Algorithm | Minim | Average | Deviation | Time |
| 30 | GA | -28.5601 | -28.3189 | 0.31785 | 358.389s |
| | HCB | -27.4514 | -26.3447 | 0.28031 | 4708s |
| | HCF | -26.7597 | -25.6211 | 0.28769 | 8845s |
| | SA | -28.4653 | -27.9044 | 0.35712 | 25s |
| 10 | GA | -9.6535 | -9.70549 | 0.13880 | 167.981s |
| | HCB | -9.48767 | -9.29414 | 0.10129 | 159s |
| | HCF | -9.49299 | -9.17366 | 0.10630 | 106s |
| | SA | -9.61093 | -9.30006 | 0.16063 | 13s |
| 5 | GA | -4.68766 | -4.77097 | 0.09508 | 94.4736s |
| | HCB | -4.68766 | -4.68508 | 0.00347 | 25s |
| | HCF | -4.68707 | -4.67728 | 0.01099 | 18s |
| | SA | -4.68766 | -4.64236 | 0.04317 | 9s |

## 6 Comparison

Now we are going to compare the results displayed above.
We can observe that GA gives us better results almost every time, except for the 30-dimensional version of the first and third functions when it actually gives us worst results.
This method also has the smallest running time, compared to the Best Improvement Hill Climb and the First Improvement Hill Climb methods.
We can notice that compared to the Simulated Annealing method, it has a bigger running time but the results obtained are still better.

## 7 Conclusion

We described the evolutionary algorithm's structure, its functioning, the parameters we utilized, and the outcomes we obtained. From the experiments we executed, it was clear that the genetic algorithm is the best method to use out of those that were provided. It offers almost ideal solutions for every function, which are better in comparison to those of the Simulated Annealing and Hill Climbing algorithms.

## References

[1] https://en.wikipedia.org/wiki/Genetic_algorithm

[2] http://www.geatbx.com/docu/fcnindex-01.html

[3] https://profs.info.uaic.ro/~eugennc/teaching/ga/

[4] https://ro.wikipedia.org/wiki/Elitism

[5] https://en.wikipedia.org/wiki/Selection_)genetic_algorithm)

[6] https://www.geeksforgeeks.org/genetic-algorithms/

[7] https://annals-csis.org/Volume_1/pliks/167.pdf