Genetic Algorithms methods in global optimization

Bita Alexandru

December 3, 2019

1 Introduction

Genetic algorithms (GAs) are powerful heuristic random search techniques that mimic the theory of evolution and natural selection. The basic principles of the genetic algorithm were investigated by John Holland in the 1970's (Holland, 1975) in the University of Michigan. The genetic algorithm has proved its strength and durability in solving many problems, and thus it is considered as an optimization tool for many researchers.

The genetic algorithm was inspired by the Darwinian theory of "survival of the fittest" by producing new chromosomes (individuals) through recombination (crossover) and mutation operations, i.e. the fittest individual is more likely to remain and mate. Therefore, the inhabitants of the next generation will be stronger, because they are produced from strong individuals, i.e. the solution evolves from one generation to another.

The GA begins with a number of random solutions (initial population), these solutions (individuals) are then encoded according to the current problem, and the quality of each individual is evaluated through a fitness function. The GA depends mainly on three operators:

- Selection (competition): The process of choosing the "best" parents in the community for mating, "best" being defined based on the current problem.
- Crossover operator: Takes two parents (chromosomes), to create a new offspring by switching segments of the parent genes; it is more likely that the new offspring would contain good parts of their parents, and consequently perform better as compared to their ancestors.
- Mutation operator: Takes a single chromosome, and alters some of its genes to create a new chromosome, hopefully better than its parent.

The better choice of these processes and the better selection of their parameters (such as the crossover and mutation rates) enhance the performance of the GA.

1.1 Motivation

The goal of this project is to develop the Genetic Algorithms (GA) for solving the Sphere, Sum-Squares, Dixon-Price and Rastrigin functions on a total of 30 runs and compare the results to non-GA strategies, such as Hill-Climbing and Simmulated Annealing.

The paper will folow the Standard implementation of GA (with constant mutation and crossover rates) and two types of Hill-Climbing (next-ascent and steepest-ascent).

2 Method

The GA implementation starts with a population of 100 individuals(chromosomes) binary represented and runs for 1000 generations. It uses constant probabilities of mutation($p_m = 0.01$) and crossover($p_c = 0.3$).

Firstly, the chromosomes are being mutated.

Secondly, they go through a phase of a *uniform crossover*. This strategy allows the offspring to have any combination of the parent chromosomes. A random binary mask of the same length as a parent chromosome is generated with a probability usually set to 0.5. The first offspring will inherit the genes of the first parent in the positions for which the mask is 0 and the genes of the second parent for the positions where the mask is 1.

In the end, selection is made through a technique called the *Wheel of Fortune*. Consider a roulette wheel where all the chromosomes (xi) in a population of size N, are located in sectors proportional

to their fitness. Denote $F = \sum_{i=1}^{N} f(x_i)$ the total fitness of the population. If this is equal to the circumference of the roulette wheel, each chromosome in the population will have a corresponding sector bounded by an arc equal to its fitness. When spinning the wheel the probability that an individual will be selected is proportional to its fitness. In order to obtain N individuals in the mating pool, the wheel is spun N times. The same individual can be chosen several times, which is very likely to happen when it has a large fitness value.

The following pseudocode describes a standard GA:

```
Initial population: generate a population of N random individuals
while stop criterion not fulfilled do
    Reproduction: create new individuals by applying genetic operators
    (mutation, crossover)
    Selection: select the fittest individuals for reproduction
    Recombination: creation of the new population
end while
```

3 Experiment

A $\bar{Val}$ score is used to determine the fit of each implementation. The GA is configured as stated above while the Hill-Climbing and Simmulated Annealing as per my previous paper.

In statistics, 30 runs is typically considered acceptable for determining the average of algorithms. As a result, a $\bar{Time}$ value is used to determine the average execution time of each implementation.

4 Results

4.1 Sphere

$f(\mathbf{x}) = f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} x_i^2, x_i \in [-5.12, 5.12]$

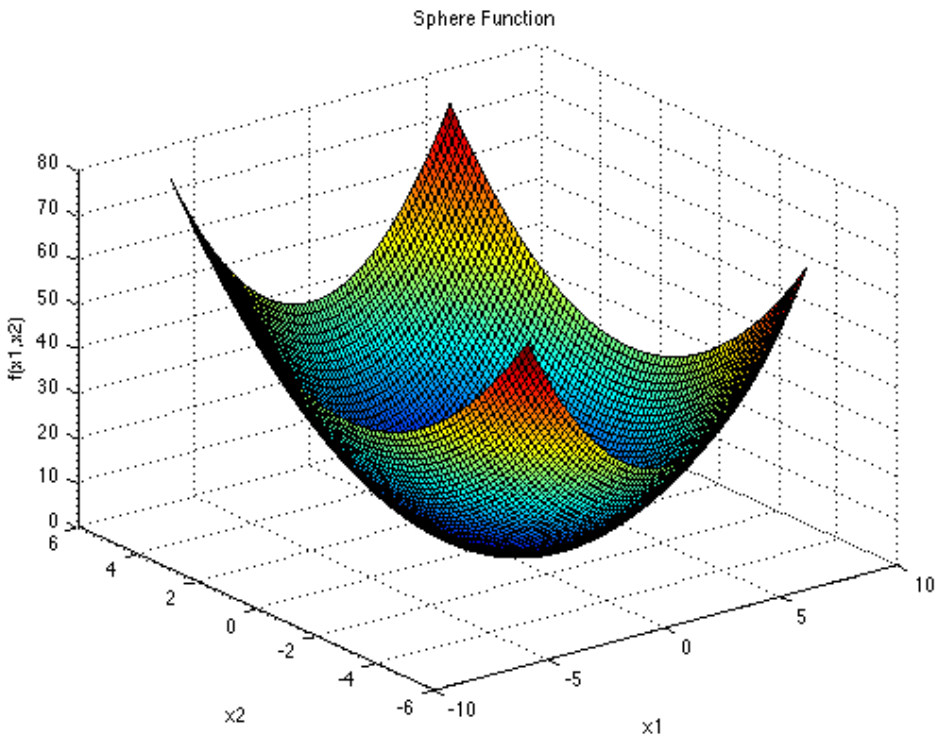

Figure 1: Sphere function [1] for 3 dimensions

	$\bar{Val}$	σ	$\bar{Time}$		$\bar{Val}$	σ	$\bar{Time}$
NAHC	0.0000	0.0000	1.0132	*GA*			
SAHC	0.0000	0.0000	1.9219		0.0000	0.0000	1.0527
SA	0.0001	0.0001	0.1386				

Table 1: Results on Sphere function for 2 dimensions

	$\bar{Val}$	σ	$\bar{Time}$		$\bar{Val}$	σ	$\bar{Time}$
NAHC	0.0000	0.0000	4.7341	*GA*			
SAHC	0.0000	0.0000	10.4517		0.3170	0.3971	4.6746
SA	0.0005	0.0002	1.8757				

Table 2: Results on Sphere function for 10 dimensions

	$\bar{Val}$	σ	$\bar{Time}$		$\bar{Val}$	σ	$\bar{Time}$
NAHC	0.0000	0.0000	20.1599	*GA*			
SAHC	0.0000	0.0000	33.9441		295.0917	24.3748	13.8355
SA	0.0017	0.0004	17.4792				

Table 3: Results on Sphere function for 30 dimensions

4.2 Sum-Squares

$f(\mathbf{x}) = f(x_1,...,x_n) = \sum_{i=1}^{n} ix_i^2, x_i \in [-5.12, 5.12]$

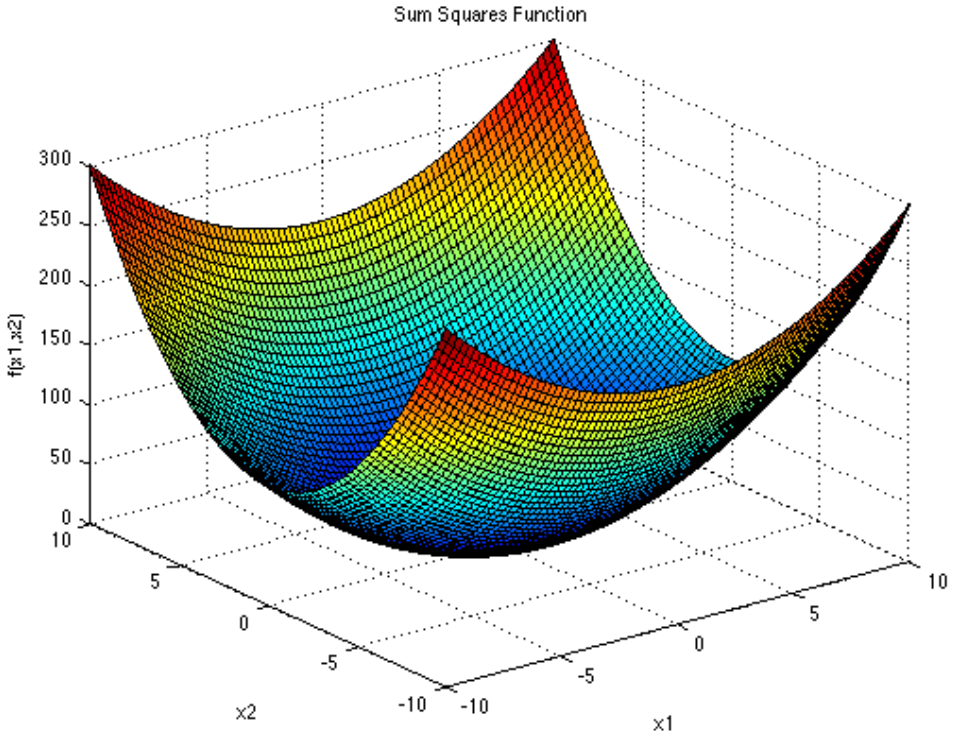

Figure 2: Sum-Squares function [2] for 3 dimensions

	$\bar{Val}$	σ	$\bar{Time}$		$\bar{Val}$	σ	$\bar{Time}$
NAHC	0.0000	0.0000	1.0142	*GA*			
SAHC	0.0000	0.0000	1.8856		0.0000	0.0000	1.0595
SA	0.0001	0.0001	0.1386				

Table 4: Results on Sum-Squares function for 2 dimensions

	$\bar{Val}$	σ	$\bar{Time}$		$\bar{Val}$	σ	$\bar{Time}$
NAHC	0.0000	0.0000	4.8128	*GA*			
SAHC	0.0000	0.0000	10.5111		3.1867	9.9838	4.7101
SA	0.0006	0.0003	1.9424				

Table 5: Results on Sum-Squares function for 10 dimensions

	$\bar{Val}$	σ	$\bar{Time}$		$\bar{Val}$	σ	$\bar{Time}$
NAHC	0.0000	0.0000	20.4952	*GA*			
SAHC	0.0000	0.0000	34.0636		4342.0479	653.5685	13.8417
SA	0.0017	0.0005	17.8632				

Table 6: Results on Sum-Squares function for 30 dimensions

4.3 Dixon-Price

$f(\mathbf{x}) = f(x_1,...,x_n) = (x_1-1)^2 + \sum_{i=2}^{n} 2(x_i^2 - x_{i-1})^2, x_i \in [-5.12, 5.12]$

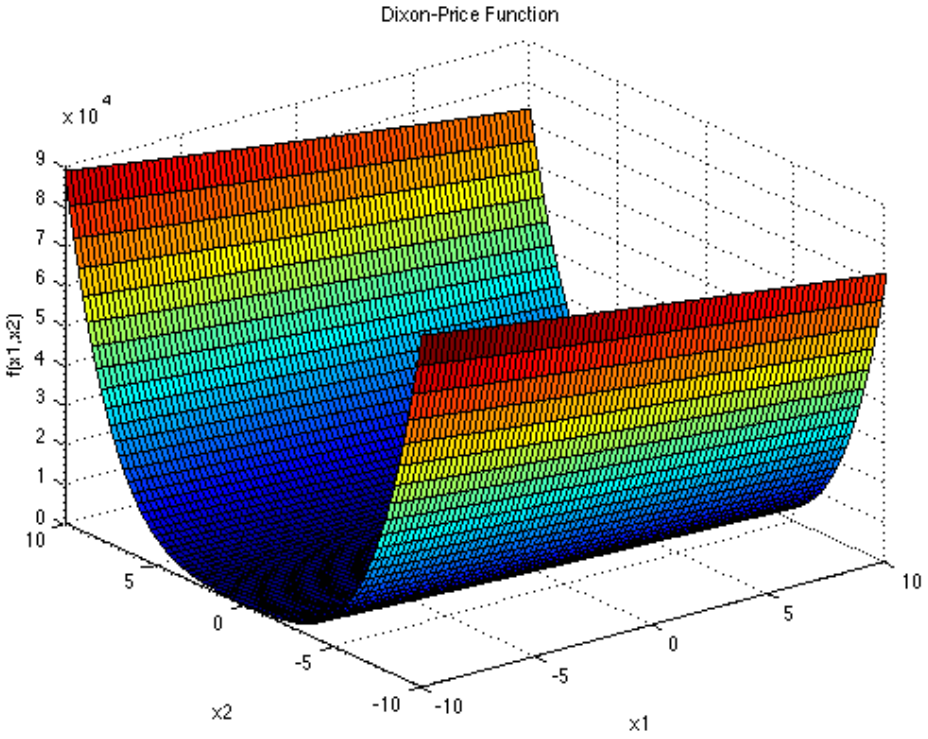

Figure 3: Dixon-Price function [3] for 3 dimensions

	$\bar{Val}$	σ	$\bar{Time}$		$\bar{Val}$	σ	$\bar{Time}$
NAHC	0.1881	0.0000	0.1660	*GA*			
SAHC	0.1881	0.0000	0.1262		0.0220	0.0307	1.1419
SA	0.1883	0.0005	0.2298				

Table 7: Results on Dixon-Price function for 2 dimensions

	$\bar{Val}$	σ	$\bar{Time}$		$\bar{Val}$	σ	$\bar{Time}$
NAHC	0.9811	0.1024	6.1580	*GA*			
SAHC	3.9705	15.9964	4.6558		6.4093	18.4152	5.5909
SA	0.6017	0.2556	3.8335				

Table 8: Results on Dixon-Price function for 10 dimensions

	$\bar{Val}$	σ	$\bar{Time}$		$\bar{Val}$	σ	$\bar{Time}$
NAHC	3588.8500	805.2574	42.9726	*GA*			
SAHC	3656.1083	841.3060	35.9190		246033.7402	66539.8795	13.2411
SA	49.2802	99.0769	23.0979				

Table 9: Results on Dixon-Price function for 30 dimensions

4.4 Rastrigin

$f(x,y) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i)), x_i \in [-5.12, 5.12]$

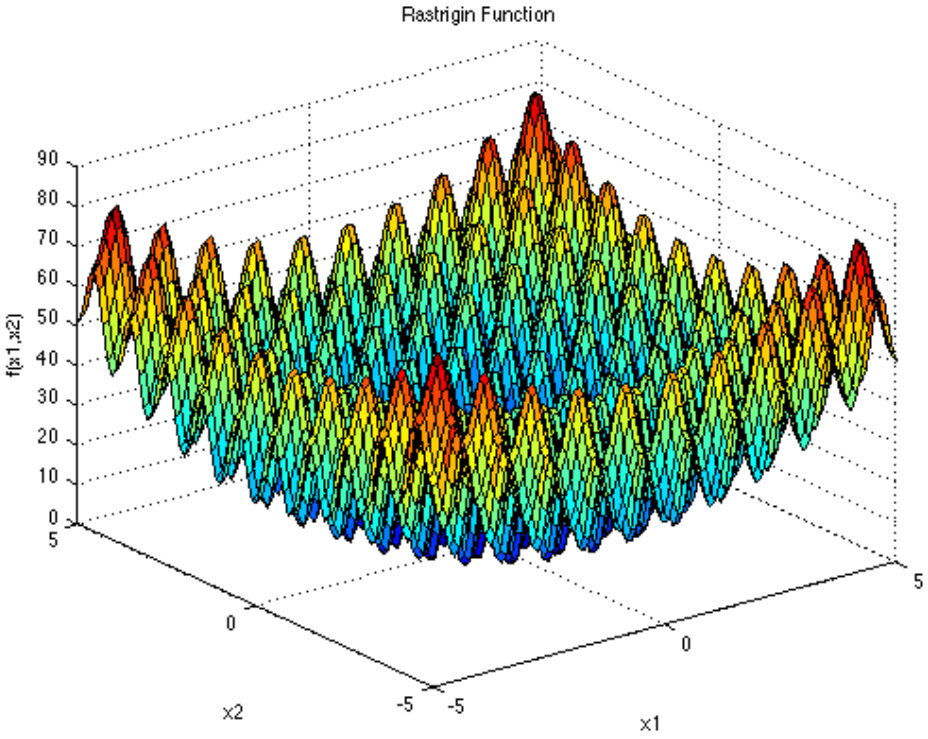

Figure 4: Rastrigin function [4] for 3 dimensions

	$\bar{Val}$	σ	$\bar{Time}$		$\bar{Val}$	σ	$\bar{Time}$
NAHC	0.0000	0.0000	1.1657	*GA*			
SAHC	0.0000	0.0000	1.6870		0.0000	0.0000	1.0844
SA	0.2241	0.2809	0.1663				

Table 10: Results on Rastrigin function for 2 dimensions

	$\bar{Val}$	σ	$\bar{Time}$		$\bar{Val}$	σ	$\bar{Time}$
NAHC	1.8793	0.5900	5.4819	*GA*			
SAHC	2.3282	0.6660	9.5314		108.5343	13.4287	4.8346
SA	9.0005	3.3089	2.1293				

Table 11: Results on Rastrigin function for 10 dimensions

	$\bar{Val}$	σ	$\bar{Time}$		$\bar{Val}$	σ	$\bar{Time}$
NAHC	22.5632	2.3905	22.9773	*GA*			
SAHC	23.4606	2.1677	30.2534		545.9192	37.0362	14.0657
SA	29.4942	5.6062	18.9126				

Table 12: Results on Rastrigin function for 30 dimensions

5 Interpretation

Several experiments have been carried out. Tables [1] to [12] show the comparison in performance of the four stated algorithms.

As shown, GA search achieves suboptimal results when running at higher dimensions. This can be ascribed to its inability to escape from local extremes. With no adaptive mechanism the search has not found the global optimum (0) in any of the 30 runs. The same can be said about both implementations of Hill-Climbing, while Simulated Annealing tends to have the best results on every function.

6 Conclusions

In conclusion, it is shown that GA has been proven to be a good method to reduce the cost of the initial feasible timetable. With a robust implementation, it managed to explore the search space efficiently and produce good results with a fast execution time.

However, the good cost obtained through the experiment with GA did not manage to outperform the results obtained by utilizing neither our proposed Hill-Climbing nor our Simulated Annealing, although the computational time taken by GA execution is much lower than both on higher dimensions.

Through the findings of this research, it makes it more understandable to us that sometimes GA is not a very good approach in solving problems.

References

[1] Sphere function
`https://www.sfu.ca/~ssurjano/spheref.html`

[2] Sum squares function
`https://www.sfu.ca/~ssurjano/sumsqu.html`

[3] Moved axis parallel hyper-ellipsoid function
`https://www.sfu.ca/~ssurjano/dixonpr.html`

[4] Rastrigin's function `https://www.sfu.ca/~ssurjano/rastr.html`