Genetic Algorithms.
Finding global minima

Mitrofan Alexandru
3B5

December 2022

Abstract

In this paper we will analyze the efficiency of genetic algorithms in finding the global minima of four mathematical functions. Through several experiments I found the parameters of the genetic algorithm that produce the best result. Finally, by comparing the genetic algorithm with other adaptive algorithms, we find that it is the most accurate.

1 Introduction

In this paper I analyzed a genetic algorithm with the aim of finding the global minima of 4 mathematical functions. To obtain the best results I experimented with different parameters. To analyze efficiency and accuracy I compared the Genetic Algorithm with 3 heuristic methods for finding the global minima (Hill Climbing First Improvement, Hill Climbing Best Improvement and Simulated Annealing).

The first section of this work is a short introduction. The second part shows how the genetic algorithm works and the methods used.

The third part describes the mathematical functions on which the tests were done, presents the parameters used and how they were chosen. The choice is supported by conducting experiments presented in the form of tables.

The fourth part shows us the results of the genetic algorithm with the best parameters in finding the global minima on the 4 functions using 5, 10 and 30 dimensions. The results are followed by their interpretation.

In the fifth part, I compared the genetic algorithm with the 3 heuristic methods (HC first, HC best, SA). The seventh part represents the conclusion of the experiments and the comparison, and the last part represents the bibliography.

2 Methods

The Genetic Algorithm works with bitstrings that represent binary numbers. It is specified in the problem data that the precision of representing numbers is 5 decimals and we also know the domain field of every function.

A bitstring within the genetic algorithm is called a chromosome. A group of such bitstrings forms a population on which various operations are performed (selection, crossover, mutation). Each iteration of the algorithm represents a generation. The purpose of the genetic algorithm is to improve the population at each generation. The improvement is based on the fitness of the chromosomes.

- The fitness function is used to measure the quality of the chromosomes. It is formulated starting from the numerical function to be optimized. It must be positive and is built for maximization (adapted individuals obtain high values of the fitness function).
- The purpose of selection is to choose for survival the most adapted individuals from the population with the hope that their descendants will had a higher fitness. In combination with variations of the crossover and mutation operators, the selection must preserve a balance between exploration and exploitation. A high selection pressure leads to the creation of a low diversity in the population created from well-adapted but suboptimal individuals, which leads to a limitation of changes and implicitly progress. On the other hand on the other hand, a low selection pressure slows down evolution.
- The crossover operation is performed between 2 chromosomes and involves their crossing. It can be done in one or more cutting points many. In the case of the genetic algorithm built by me, the crossover is done in 2 points, and the choice of chromosomes is random.
- The mutation operation involves changing one or more bits of a chromosome with a certain probability.
- Elitism represents the group of the best individuals in a population who are guaranteed to be chosen for the new generation and on whom the cross and mutate operations are not applied.

3 Experiement Description

3.1 Functions

De Jong

$$f_1(x) = \sum_{i=1}^{n} x_i^2$$

$-5.12 \leq x_i \leq 5.12$
global minima: $f(x) = 0$, $i = 1 : n$.

Schwefel

$$f_7(x) = \sum_{i=1}^{n} -x_i \cdot \sin\left(\sqrt{|x_i|}\right)$$

$-500 \leq x_i \leq 500$
global minima: $f(x) = -n \cdot 418.9829$; $i = 1 : n$.

Rastrigin

$$f_6(x) = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2 \cdot \pi \cdot x_i))$$

$-5.12 \leq x_i \leq 5.12$
global minima: $f(x) = 0$, $i = 1 : n$.

Michalewicz

$$f_{12}(x) = -\sum_{i=1}^{n} \sin(x_i) \cdot \left(\sin \frac{i \cdot x_i^2}{\pi}\right)^{2 \cdot m}$$

$i = 1 : n.$, $m = 10$, $0 \leq x_i \leq \pi$
global minima: f(x)=-4.687 (n=5); f(x)=-9.66 (n=10); f(x)=-29.63088 (n=30)

3.2 Parameters

For the experiments of finding the best parameters, I studied only the functions with 30 dimensions. For the other experiments I studied the functions with 5, 10 and 30 dimensions. The results have a precision of 5 decimal places and I performed for each result 30 runs of the algorithm.

- Population size = 200
- Number of generations = 2000
- Crossover rate: 90% (Rastring, Michalewicz), 50% (De Jong), 80% (Schwefel)
- Mutation rate: 0.001 (all functions)
- Elitism: 15 (De Jong), 35 (Rastring, Michalewicz, Schwefel)
- Termination Condition: 2000 iterations (generations)
- Fitness Function: $\frac{1}{999999999 + value}$

I chose the population of 200 because I had poor results with a smaller population, and with a larger population the execution time increased.

The crossover rate must be between 50% and 90%. I performed the parameter selection experiments on crossovers of 50%, 70%, 80% and 90%. I also mention that the crossover is done in 2 cutting points, and the chromosomes that are crossed are chosen randomly. I chose crossover with 2 cut points because the results with 1 or more than 2 cut points were very poor.

The mutation rate must be lower than 0.01. For higher mutation rates, the results are no longer accurate. The experiments were done performed on mutation rates of 0.001, 0.002, 0.003, 0.005.

I experienced elitism values of 15, 25 and 35. For values higher than 35 I did not notice improvements.

3.3 Parameters Experiment

For all the functions below I studied the version with 30 dimenisions. The population size is 200, number of generations is 2000 and the number of runs for each line is 30.
In the first four tables, I studied all combinations of cross rate and mutation rate, 16 combinations for each table. The elitism for them is 15.
De Jong

p_cross	p_mut	min	max	mean	stdev
0.5	0.001	0	0	0	0
	0.002	0	0	0	0
	0.003	0	0	0	0
	0.005	0.00001	0.00004	0.00002	0
0.7	0.001	0	0	0	0
	0.002	0	0.00003	0	0.00001
	0.003	0.00001	0.00004	0.0001	0
	0.005	0.00005	0.00031	0.00015	0.00005
0.8	0.001	0	0.00002	0	0
	0.002	0	0.00002	0	0
	0.003	0.00001	0.00009	0.00001	0
	0.005	0.00020	0.00063	0.00015	0.00005
0.9	0.001	0.00002	0.00045	0.00006	0.00010
	0.002	0.00002	0.00028	0.00006	0.00007
	0.003	0.00006	0.00048	0.00020	0.00012
	0.005	0.00067	0.00162	0.00094	0.00030

Rastring

p_cross	p_mut	min	max	mean	stdev
0.5	0.001	7.06105	18.33252	10.58040	3.30463
	0.002	9.62419	21.33405	14.81499	3.64109
	0.003	8.72923	25.54334	17.90798	5.77033
	0.005	14.05964	32.43602	22.65999	5.15734
0.7	0.001	6.38990	11.89329	9.25164	1.49487
	0.002	9.25497	17.49201	13.55996	2.99670
	0.003	10.60295	18.83493	14.02858	2.79083
	0.005	11.40258	26.39581	17.08557	4.31607
0.8	0.001	6.69774	12.18403	9.88822	1.80146
	0.002	9.30452	18.66720	11.96575	2.76871
	0.003	9.82395	21.43221	14.56172	3.65719
	0.005	13.20436	24.38427	19.29304	4.40785
0.9	0.001	3.00898	11.21676	8.29169	2.47425
	0.002	4.85756	14.61014	10.8233	3.45828
	0.003	8.29634	17.87596	14.16339	2.74692
	0.005	16.13060	36.28601	21.77508	4.85248

Michalewicz

p_cross	p_mut	min	max	mean	stdev
0.5	0.001	-28.51741	-27.67856	-28.16633	0.24266
	0.002	-28.24609	-26.50931	-27.33366	0.59404
	0.003	-28.63305	-25.85651	-27.05028	0.91291
	0.005	-27.82495	-24.72613	-26.57369	1.10404
0.7	0.001	-28.67695	-27.65289	-28.05210	0.38380
	0.002	-28.66527	-26.48004	-27.58958	0.68985
	0.003	-28.16663	-26.43321	-27.28057	0.62431
	0.005	-27.61165	-26.01993	-26.69883	0.57895
0.8	0.001	-28.92175	-27.29012	-28.07445	0.51591
	0.002	-29.08220	-27.20266	-28.07544	0.57282
	0.003	-27.82113	-26.64950	-27.39124	0.45629
	0.005	-28.46118	-26.41734	-27.23915	0.55677
0.9	0.001	-28.55194	-27.41842	-28.17247	0.33523
	0.002	-28.43797	-27.11749	-27.94565	0.46958
	0.003	-28.31174	-26.36259	-27.27627	0.57170
	0.005	-28.06250	-26.20576	-27.19605	0.51756

Schwefel

p_cross	p_mut	min	max	mean	stdev
0.5	0.001	-12528.91016	-12376.23242	-12462.3593	61.12763
	0.002	-12565.17285	-12338.03027	-12452.6105	77.02305
	0.003	-12464.04102	-12147.69922	-12331.4609	117.48717
	0.005	-12380.26172	-11801.66309	-12155.1382	205.14087
0.7	0.001	-12541.58008	-12489.72363	-12520.2755	22.89826
	0.002	-12567.57031	-12411.33789	-12502.6324	59.89977
	0.003	-12567.86523	-12346.62012	-12461.2418	75.87923
	0.005	-12494.92871	-11929.89648	-12333.8451	182.76500
0.8	0.001	-12568.13574	-12351.52246	-12531.4799	74.32565
	0.002	-12567.57617	-12346.18555	-12482.4304	88.05929
	0.003	-12568.12500	-12161.40137	-12427.1399	135.79045
	0.005	-12508.23535	12027.27930	-12270.652	179.42105
0.9	0.001	-12568.61426	-12414.48633	-12522.0009	51.44852
	0.002	-12566.59863	-12311.84180	-12476.6144	91.06019
	0.003	-12533.02637	-12291.54590	-12436.3176	72.71800
	0.005	-12469.23047	-12258.35156	-12376.0767	89.68403

In the following 3 tables, I studied the combinations between elitism and cross rate. The mutation rate remained 0.001 because I noticed that a higher mutation rate produces weaker results.

Rastring

elitism	p_cross	p_mut	min	max	mean	stdev
15	0.9	0.001	3.00898	11.21676	8.29169	2.47425
25			2.00009	7.99527	4.38706	1.93200
35			0.99497	5.22236	2.60623	1.23969

Michalewicz

elitism	p_cross	p_mut	min	max	mean	stdev
15	0.9	0.001	-28.55194	-27.41842	-28.17247	0.33523
25			-29.32378	-28.15184	-28.59430	0.30580
35			-29.27469	-28.32583	-28.82810	0.27272

Schwefel

elitism	p_cross	p_mut	min	max	mean	stdev
15	0.8	0.001	-12568.13574	-12351.52246	-12531.4799	74.32565
25			-12568.53711	-12533.99316	-12562.3949	11.70283
35			-12568.64941	-12567.82324	-12568.2442	0.27206

Therefore, I found the best parameters for each function:
- De Jong: elitism=15, cross-rate=0.50, mutation rate=0.001
- Rastring: elitism=35, cross-rate=0.90, mutation rate=0.001
- Michalewicz: elitism=35, cross-rate=0.90, mutation rate=0.001
- Schwefel: elitism=35, cross-rate=0.80, mutation rate=0.001

4 Results

De Jong
$elitism = 15$, $p_{cross} = 0.5$, $p_{mut} = 0.0001$

D	global_min	$f(x)_{min}$	$f(x)_{max}$	$f(x)_{mean}$	St_{dev}	t_{min}	t_{max}
5		0	0	0	0	14.78247	15.71676
10	0	0	0	0	0	20.12442	21.52442
30		0	0	0	0	43.31158	44.96529

Rastring
$elitism = 35$, $p_{cross} = 0.9$, $p_{mut} = 0.0001$

D	global_min	min	max	mean	stdev	tmin	tmax
5		0	0	0	0	13.65004	14.09618
10	0	0	0	0	0	19.14063	19.89344
30		0.99497	5.22236	2.60623	1.23969	42.00233	42.79230

Michalewicz
$elitism = 35$, $p_{cross} = 0.9$, $p_{mut} = 0.0001$

D	global_min	min	max	mean	stdev	tmin	tmax
5	-4.68765	-4.68766	-4.68766	-4.68766	0	13.58961	14.43540
10	-9.66015	-9.66015	-9.66015	-9.66015	0	18.62850	19.41970
30	-29.63088	29.27469	28.32583	28.82810	0.27272	40.08584	40.75270

Schwefel
$elitism = 35$, $p_{cross} = 0.8$, $p_{mut} = 0.0001$

D	global_min	min	max	mean	stdev	tmin	tmax
5	-2094.91455	-2094.91455	-2094.91455	-2094.91455	0	14.74586	15.21702
10	-4189.82910	-4189.82910	-4189.82910	-4189.82910	0	22.23386	22.87598
30	12569.487	-12568.64941	-12567.82324	-12568.2442	0.27206	50.65664	59.67078

In the case of each function for 5 and 10 dimensions, the genetic algorithm finds the global minimum with a precision of 5 decimal places, and for 30 dimensions the results are quite close to the global minimum.

5 Comparison with other adaptive algorithms

Function	global_min	Algorithm	min	max	mean	stdev	tmin	tmax
De Jong	0	HC first	0	0	0	0	44.627	45.441
		HC best	0	0	0	0	80.543	87.471
		SA	0.00020	0.00063	0.00036	0.0001	24.809	26.997
		GA	0	0	0	0	43.311	44.965
Rastring	0	HC first	37.4072	49.8073	43.70897	3.75739	39.044	42.061
		HC best	28.9757	39.2327	35.47829	3.10688	70.141	74.585
		SA	5.21600	28.44792	17.72855	7.31128	26.309	28.368
		GA	0.99497	5.22236	2.60623	1.23969	42.002	42.792
Michalewicz	-29.63088	HC first	-27.1615	-24.6143	-25.43349	0.56259	43.061	52.250
		HC best	-26.8392	-25.9418	-26.33364	0.22927	80.082	90.701
		SA	-28.76496	-27.12881	-27.78552	0.47401	39.871	41.155
		GA	-29.27469	-28.32583	-28.82810	0.27272	40.085	40.752
Schwefel	-12569.487	HC first	-10697	-10254.8	-10439.7714	143.575	106.851	112.33
		HC best	-11315.5	-10892.3	-11046.442	156.935	196.591	201.89
		SA	-12465.94	-11567.92	-11948.159	258.051	49.577	53.102
		GA	-12568.64	-12567.82	-12568.244	0.2720	50.656	59.670

In the case of the De Jong function, the Genetic Algorithm, HC first and HC best have maximum precision, but the Genetic Algorithm has the best execution time. Simulated Annealing is twice faster, but does not get the result as precisely. Therefore, for De Jong the genetic algorithm is the best.

In the case of the Rastring function, the Genetic Algorithm produces by far the best result. The time efficiency is about the same as HC first, twice as fast as HC best, but twice as slow as Simulated Annealing. Overall the genetic algorithm is better.

In the case of the Michalewicz function, the genetic algorithm produces the best result and is the fastest, with the execution time being equal to Simulated Annealing.

For Schwefel, the genetic algorithm produces the best result, but is slightly slower than Simulated Annealing.

6 Conclusions

In conclusion, the Genetic Algorithm produces very good results as long as the best parameters are found. If we were to make a hierarchy of the 4 algorithms, it would look like this: $GA > SA > HC_{best} > HC_{first}$.

For the Schwefel function, I noticed that the Genetic Algorithm produces the best results among all algorithms for any configuration of parameters. For Rastring and Michalewicz functions it produces the best result for most parameter configurations, but there are parameter configurations where the Genetic Algorithm is weaker than Simulated Annealing.

The process of finding the optimal parameters is a long one, so in case of lack of time, Simulated Annealing can also be used to find the global minimum for Rastring and Schwefel, but it is not very precise.

The time efficiency of the Genetic Algorithm for 5 and 10 dimensions is poor because the stopping condition is to reach 2000 generations. A better stopping condition would be to get a certain number of generations that do not produce a noticeable improvement. Thus, the genetic algorithm would no longer iterate 2000 generations.

7 Bibliography

https://profs.info.uaic.ro/ eugennc/teaching/ga/
https://www.overleaf.com/
https://www.tablesgenerator.com/
https://www.calculatorsoup.com/
https://en.wikipedia.org/wiki/Genetic$_algorithm$

"Choosing Mutation and Crossover Ratios for Genetic Algorithms.A Review with a New Dynamic Approach": https://www.mdpi.com/2078-2489/10/12/390

https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

https://www.mathworks.com/help/gads/how-the-genetic-algorithm-works.html
http://www.geatbx.com/docu/fcnindex-01.html