Finding the minimum of optimization benchmarking functions using Genetic Algorithms and comparison with Hill Climbing and Simulated Annealing methods. University "Alexandru Ioan Cuza" of Iași

Ion Zmeu E4

November 29 2022

1 Abstract

This article describes the results of approximation the minimum of the Rastrigin,Michalewicz,De Jong 1,Schwefel functions using Genetic Algorithms , the Hill Climbing and Simulated Annealing methods by running the algorithms with 1000 iterations, 100 population size (for the Genetic Algorithms) and with dimensions 5,10 and 30.It also shows The definition of these functions and the graphics for them. Iterated Hill Climbing and Simulated Annealing are the 2 heuristic methods that were used.Both of these methods involve non-deterministic algorithms that give good results when trying to find, actually reach, or approach a global minimum.I optimized the Genetic Algorithm described in this paper using adaptive parameters. After the experiment I concluded that although the results are really close to to the best , different methods need different amount of time and resources to reach such a final solution.

2 Introduction

Genetic Algorithm (GA) is a metaheuristic inspired by the process of natural selection that belongs to the larger class of Evolutionary Algorithms (EA). Genetic Algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on biologically inspired operators such as mutation, crossover and selection[1].In mutation the bits of chromosomes will be flipped based on a probability in my case its 0.003.The crossover is responsible for creating new offspring by mixing the genes of two randomly choosen parent chromosomes.After these operators the new population will undergo the process of selection where they will be selected for the next generation based on their fitness value.In our case we randomly select the best chromosomes using a lucky wheel where the probability to be choosen is proportional to the fitness value.This process is repeated untill we have a full population (in our case its 100 chromosomes). The Genetic Algorithm was modified further using adaptive parameters in a way that if the last five best solutions were the same then the

mutation probability rises from 0.003 to 0.01 and the crossover probability rises from 0.4 to 0.6.Based on the results I concluded that Genetic Algorithms are faster and returns decent solutions close to the actual minima of the mentioned above functions compared to the Hill Climbing and Simulated Annealing.

3 Methods

3.1 Solution Representation

The solution is represented as a population which contains 100 vectors of bites of size B*D where B is the number of bites needed to represent our solution and D is the number of dimensions set.This vector is translated into floating point numbers with a precision of 5.

3.2 Neighbor

I am flipping one bit from current solution to get a neighbor at distance hamming of 1;

3.2.1 First Improvement

This improvement chooses a neighbor from the current solution and replaces the current solution if it is better than it .

3.2.2 Best Improvement

This improvement checks all the neighbors and replaces the current solution with the best neighbor which has the best improvement.

3.2.3 Worst Improvement

This improvement checks a number of random neighbors and replaces the current solution with the best neighbor (which may not be the best solution but it lets the algorithm to check more potentially better solutions).

3.3 Algorithms

3.3.1 Genetic Algorithm

This Algorithm starts with a random population and using a fitness function it creates a selection where it chooses the best chromosomes to be modified and operated on to obtain a solution to our function(which in our case is

$$\frac{1}{g + 0.00001 - minimum}$$

where a is the minimum of the specific function and g is the current candidate) using mutation and crossover operators.

3.3.2 Lucky wheel[2]

```
1.   Evaluate P                  for i:=0 to POP_SIZE
2.                                    eval[i] = f( P(i) )
3.   Total fitness                for i:=0 to POP_SIZE
4.                                    T += eval[i]
5.   Individual sel.prob.         for i:=0 to POP_SIZE
6.                                    p[i] = eval[i] / T
7.   Accumulated sel.prob.                    q[0] = 0
8.                                for i:=0 to POP_SIZE
9.                                    q[i+1] = q[i] + p[i]
10.   Selection                   for i:=0 to POP_SIZE
11.                                     uniform random r in (0,1]
12.                               select for the next generation the individual j
13.                                     for which q[j] ¡ r ¡= q[j+1]
```

3.3.3 Hill Climbing

This algorithm starts with an arbitrary solution to a problem, then attempts to find a better solution by making an incremental change to the solution.[3]

3.3.4 Simulated Annealing

This algorithm is a metaheuristic to approximate global optimization in a large search space for an optimization problem it uses temperature to make a change to the solution which can be decremental but this way it explores a larger space.[4]

3.4 Stop condition

3.4.1 Genetic Algroithm

The Genetic Algorithm stops after it is done with all 1000 iterations that were set for this experiment.

3.4.2 Hill Climbing

The Hill Climbing algorithm stops after going through all the iterations set.

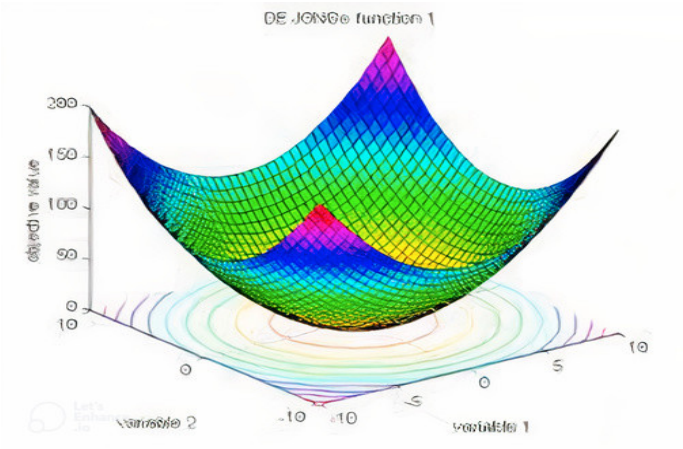
3.4.3 Simulated Annealing

The temperature decreases and The Simulated Annealing algorithm stops after its temperature reaches a low enough value.

4 Functions

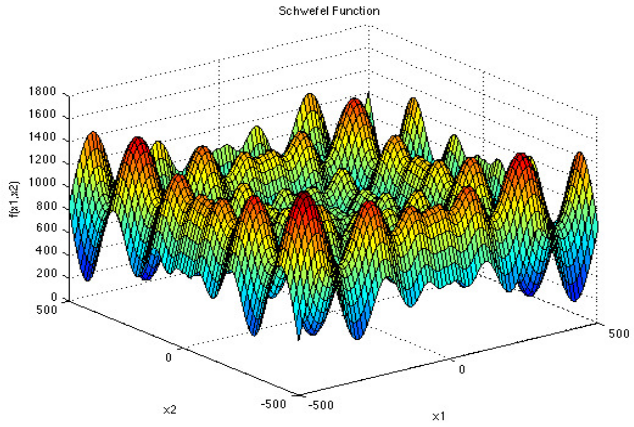
4.1 De Jong's function [5]

$$f_1(x) = \sum_{i=1}^{n} x_i^2, x_i \in [-5.12, 5.12]$$



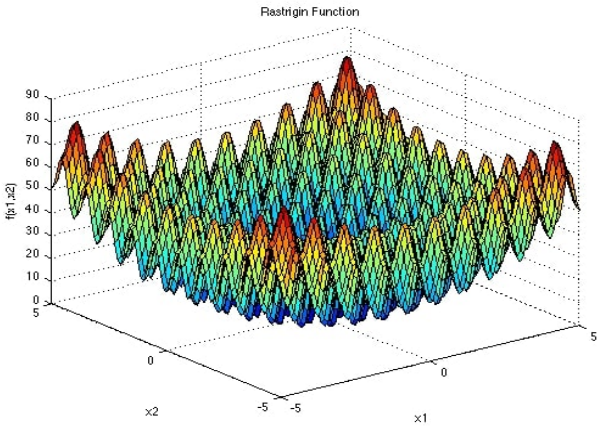
4.2 Schwefel's function [5]

$$f_7(x) = \sum_{i=1}^{n} -x_i \cdot sin(\sqrt[2]{|x_i|}), x_i \in [-500, 500]$$



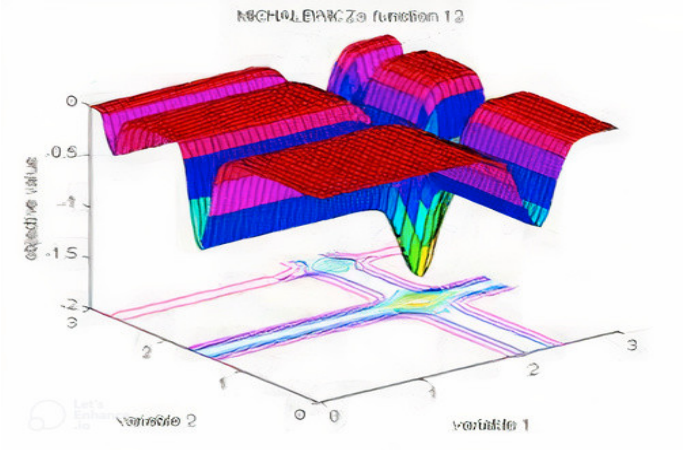
4.3 Rastrigin's function [5]

$$f_6(x) = 10 \cdot n + \sum_{i=1}^{n} x_i^2 - 10 \cdot cos(2 \cdot \pi \cdot x_i), x_i \in [-5.12, 5.12]$$



4.4 Michalewicz's function [5]

$$f_{12}(x) = -\sum_{i=1}^{n} sin(x_i) \cdot \left(sin\left(\frac{i \cdot x_i^2}{\pi} \right) \right)^{2 \cdot m}, x_i \in [0, \pi]$$



5 Results

5.1 De Jong's function 30 dimensions

	SA	First	Best	Worst	GA	GA+
dimension	30	30	30	30	30	30
average	0	0	0	0	0	0
best	0	0	0	0	0	0
avg. time	19.33	350.53	629.83	635.8	4.3	4.63
deviation	0	0	0	0	0	0

5.2 De Jong's function 10 dimensions

	SA	First	Best	Worst	GA	GA+
dimension	10	10	10	10	10	10
average	0	0	0	0	0	0
best	0	0	0	0	0	0
avg. time	9.8	55.9	34.46	56.86	3.2	3.63
deviation	0	0	0	0	0	0

5.3 De Jong's function 5 dimensions

	SA	First	Best	Worst	GA	GA+
dimension	5	5	5	5	5	5
average	0	0	0	0	0	0
best	0	0	0	0	0	0
avg. time	7.23	15.96	10.4	15.93	4.3	4.63
deviation	0	0	0	0	0	0

5.4 Schwefel's function 30 dimensions

	SA	First	Best	Worst	GA	GA+
dimension	30	30	30	30	30	30
average	-11588.03418	-10760.51667	-11312.33021	-11291.15928	-12567.6205	-12567.7136
best	-12097.9	-10937.8	-11873.2	-11604.5	-12569.4873	-12569.4873
avg. time	12.63	586.23	852.16	873.16	35.6	6.83
deviation	322.24088	109.32534	152.28449	151.73394	9.00268	6.74889

5.5 Schwefel's function 10 dimensions

	SA	First	Best	Worst	GA	GA+
dimension	10	10	10	10	10	10
average	-4018.37	-3910.2	-4082.39	-4058.55	-4189.59557	-4189.52908
best	-4189.61	-4128.29	-4189.21	-4155.28	-4189.8291	-4189.8291
avg. time	6.63	49.4	71.66	74.33	5.6	4.66
deviation	147.09506	80.13129	0.80615	0.84973	1.25650	0.93629

5.6 Schwefel's function 5 dimensions

	SA	First	Best	Worst	GA	GA+
dimension	5	5	5	5	5	5
average	-2017.06	-2071.02	-2093.9	-2094.78	-2094.87812	-2094.87812
best	-2094.91	-2094.91	-2094.91	-2094.91	-2094.91	-2094.91
avg. time	5.1	14.03	18.83	19.5	5.6	4.66
deviation	78.43734	21.84317	4.84271	0.09603	0.19564	0.04152

5.7 Rastrigin's function 30 dimensions

	SA	First	Best	Worst	GA	GA+
dimension	30	30	30	30	30	30
average	34.54481	36.60717	28.043	27.94346	19.92626	18.92965
best	23.30365	31.21634	18.1026	21.37	14.18555	12.44954
avg. time	13.1	489.23	710.83	717.36	35.8	42.5
deviation	6.73113	2.39853	2.63904	2.69739	3.26591	3.49055

5.8 Rastrigin's function 10 dimensions

	SA	First	Best	Worst	GA	GA+
dimension	10	10	10	10	10	10
average	10.22801	5.73265	3.90917	4.44261	5.36147	5.26257
best	4.22076	2.47168	1.99499	2.23587	14.18555	12.44954
avg. time	6.93	35.53	52.76	53.8	26.9	34.2
deviation	3.97924	1.22616	0.80615	0.84973	3.92153	2.56386

5.9 Rastrigin's function 5 dimensions

	SA	First	Best	Worst	GA	GA+
dimension	5	5	5	5	5	5
average	3.94587	0.89387	0.23232	0.36499	2.56803	2.45133
best	0.00006	0	0	0	0	0
avg. time	5.56	9.9	14.13	9.7	26.26	28.24
deviation	3.01231	0.41261	0.42112	0.47968	1.94183	2.84948

5.10 Michalewicz's function 30 dimensions

	SA	First	Best	Worst	GA	GA+
dimension	30	30	30	30	30	30
average	-27.20538	-26.20239	-26.96376	-26.88795	-25.91367	-26.184693
best	-28.26675	-26.75873	-27.59027	-27.83535	-27.15634	-27.56222
avg. time	14.6	420.16	697.13	703.33	17.8	19.6
deviation	0.58748	0.27563	0.00010	0.33875	0.89499	2.84948

5.11 Michalewicz's function 10 dimensions

	SA	First	Best	Worst	GA	GA+
dimension	10	10	10	10	10	10
average	-9.03964	-9.30231	-9.40138	-9.32930	-8.78709	-9.17501
best	-9.51256	-9.55196	-9.65427	-9.4484	-9.38638	-9.57849
avg. time	6.96	32.33	52.03	53.46	7.53	10.8
deviation	0.27974	0.11747	0.10284	0.07114	0.29004	0.24425

5.12 Michalewicz's function 5 dimensions

	SA	First	Best	Worst	GA	GA+
dimension	5	5	5	5	5	5
average	-4.53503	-4.68508	-4.68640	-4.68572	-4.59278	-4.51557
best	-4.68764	-4.68764	-4.68766	-4.68766	-4.68766	-4.68766
avg. time	4.46	7.73	12.33	12.83	3.56	7
deviation	0.12561	0.00489	0.00102	0.00418	0.16157	0.13005

6 Conclusion

From this experiment and all the data obtained I conclude that Genetic Algorithms as well as Hill climbing and Simulated Annealing algorithms are suitable for solving the problem of finding the minimum of optimization benchmarking functions all of them producing a result close enough to or even the best one.One big difference was that Genetic Algorithm has the best execution time while the results were more than decent enough.

7 References

[1]https://en.wikipedia.org/wiki/Genetic_algorithm
[2]https://profs.info.uaic.ro/~eugennc/teaching/ga/:Explanation for GA by Eugen Croitoru
[3]https://en.wikipedia.org/wiki/Hill_climbing
[4]https://en.wikipedia.org/wiki/Simulated_annealing
[5]http://www.geatbx.com/docu/fcnindex-01.html#P140_6155
[]https://www.calculator.net/standard-deviation-calculator.html
[]https://www.youtube.com/watch?v=_ThdIOA9Lbk:Hill Climbing Algorithm by edureka!
[]https://www.youtube.com/watch?v=XNMGq5Jjs5w:Simulated Annealing with Python by Johnnyboycurtis