Finding the global minimum of optimization functions via variations of Hill Climbing and Simulated Annealing algorithms

University "Alexandru Ioan Cuza of Iasi"

Habasescu Andrei 2E2

8^{th} of November 2022

Abstract

In this paper we will be going over the properties of the Hill Cliimbing and Simulated Annealing algorithms on the particular case of being tasked with finding the global minimum of optimization functions, namely **DeJong's, Michalewicz's, Schwefel's** and **Rastrigin's**.
We will be applying the 3 variations of the Iterative Hill Climbing algorithm (First Improvement, Best Improvement, Worst Improvement) as well as the meta-heuristic Simulated Annealing algorithm in order to probe the efficiency of the said algorithms on various sizes of data, serving as gauging criteria will be : *Execution Time*, *Best, Worst and Average Results*, *Deviation*.
As previously mentioned the algorithms will run on different sizes of data to be computed *(5, 10 and 30 dimensions)* in order to see the differences in algorithm's behaviour on the Optimization Functions. By inserting the data obtained through testing in tables we are to see which algorithms are best suited for the different sizes and types of functions. All the algorithms will be running with 1000 iterations as minimum, the precision is set to 10^{-5}.
By the end of this paper we will have explained why Simulated Annealing is best fitted to large sizes of data while Best Improvement variation of the Hill Climbing algorithm is best suited for smaller dimensions of the data.

1 Introduction

In the following report we will see the results of running Optimization Functions *(Schweifel's, DeJong's, Rastrigin's, Michalewicz's)* through the Hill Climbing Variations *(First, Worst and Best Improvements)* and the Simulated Annealing algorithms on 5, 10 and 30 dimensions each, with 1000 iterations. We will break down the obtained data so as to determine which methods are best for computing the Global Minimum of the aforementioned functions, main upsides and downsides being the Execution Time and the Accuracy of computation.

2 Method

In order to make the algorithms run the Optimization Functions we will generate strings of bits which will represent the possible solutions to the said functions. To do so we require the domains of each function to determine the range, the precision (in this case 10^5) to compute the necessary number of bits for the representation of a single candidate.[**uaic-ag**]

$$number\ of\ bits = log_2(10^5 \cdot\ (upper\_limit\ -\ lower\_limit))$$

As we are operating on several dimensions, we multiply the bit count with the number of dimensions, computing the total length of the bit string. Now using a pseudo-random number generator, in this particular case the *Mersenne Twister 19937 Generator*, we create each bit of the candidates' bit string, which is saved in a boolean type vector, and whose decimal value we compute for each

dimension and store in a separate int vector. At this point we got our candidates ready to be tested against the Optimization Functions via the Hill Climbing and Simulated Annealing algorithms.

2.1 Hill Climbing

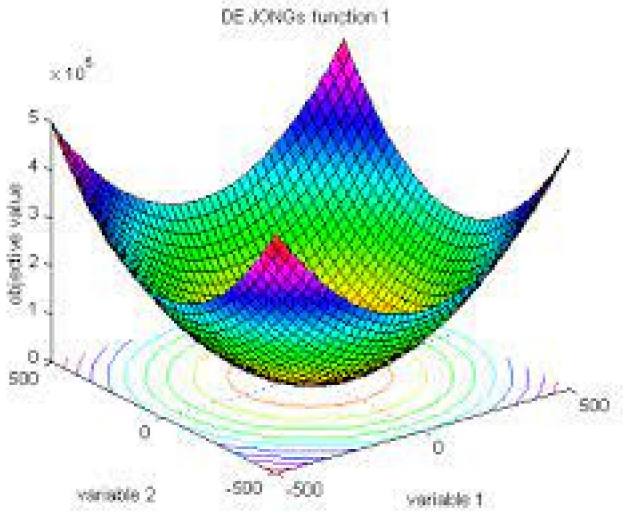
Hill Climbing[**wikipedia-HC**] is a mathematical optimization technique which operates via local search. It is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by making an incremental change to the solution. The said changes are the **Hamming Neighbours** of the candidates' bit strings, which we compute by negating bits of the potential solutions, later tested against the original value in the analysed Optimisation Function in order to see the improvement of the solution. The algorithm ends once completes a set number of iterations. *(1000 in our case)*

2.2 Simulated Annealing

Simulated Annealing[**wikipedia-SA**] is a meta-heuristic optimization technique for approximating the global optimum of a given function in a large search space for an optimization problem. The specific of this algorithm is that it uses a variable *Temperature* which determines the shift between iterations, which is later tested for the analysed function and we select the solutions which are closer to the desired result (in this case the Global Minimum). As the Temperature decreases over time, so does the scope of improvement, eventually zeroing on the Global Minimum or a Local Minimum value. Once the Temperature hits the value of 0, there is no change in the generated results, consequently marking the end of the algorithm's runtime.

3 Optimization Functions

3.1 DeJong's Function

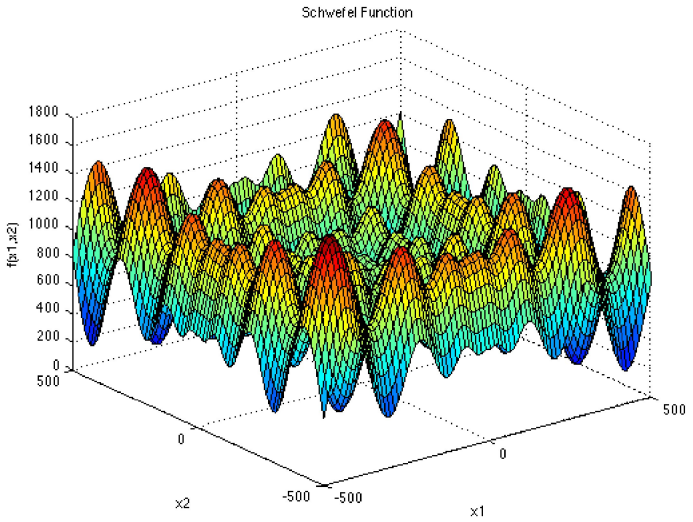


[**geatbx**]

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

Global Minimum : $f(x) = 0,\ x_i = 0, \forall i \in [1; n]$

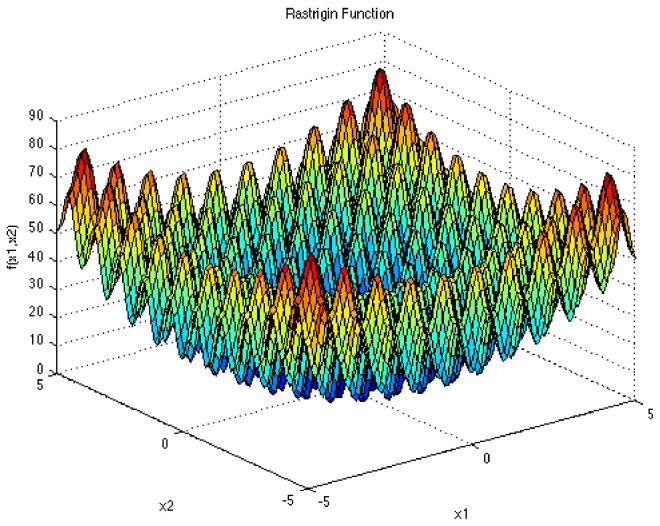
3.2 Schwefel's Function



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

Global Minimum : $f(x) = -n \cdot 418.9829, x(i) = 420.9687, \forall i \in [1; n]$
$n = 5 : f(x) = -2094.91450$
$n = 10 : f(x) = -4189.82900$
$n = 30 : f(x) = -12569.48700$

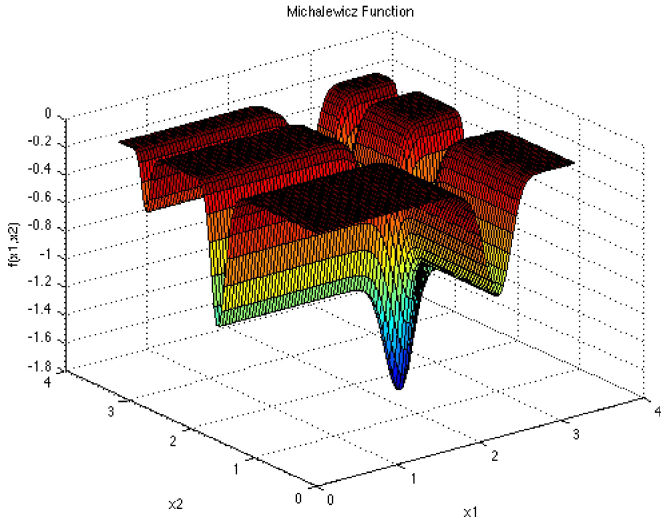
3.3 Rastrigin's Function



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

Global Minimum : $f(x) = 0, \; x_i = 0, \forall i \in [1; n]$

3.4 Michalewicz's Function



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

Global Minimum :

$n = 5 : f(x) = -4.68765$

$n = 10 : f(x) = -9.66015$

$n = 30 : f(x) = -29.63088$

4 Results

4.1 DeJong's Function

DeJong	Time												Value											
	SA			HCB			HCF			HCW			SA			HCB			HCF			HCW		
Dimensions	5	10	30	5	10	30	5	10	30	5	10	30	5	10	30	5	10	30	5	10	30	5	10	30
avg	6.26	10.83	14.1	2.26	6.06	142.06	0.96	5.8	68.3	0.16	5.36	11.26	0	0	0	0	0	0	0	0	0	0.00003	0	
best val/total time	188	325	423	68	182	4262	29	174	2049	5	161	338	0	0	0	0	0	0	0	0	0	0	0	
worst	/	/	/	/	/	/	/	/	/	/	/	/	0	0	0	0	0	0	0	0	0	0.0004	0	
deviation	/	/	/	/	/	/	/	/	/	/	/	/	0	0	0	0	0	0	0	0	0	0.0001	0	

4.2 Schwefel's Function

Schwefel	Time												Value											
	SA			HCB			HCF			HCW			SA			HCB			HCF			HCW		
Dimensions	5	10	30	5	10	30	5	10	30	5	10	30	5	10	30	5	10	30	5	10	30	5	10	30
avg	4.13	9.46	14.2	5.8	30.13	529.66	3.36	16.98	315.66	0.46	1.3	9	-2077.508036	-4133.251349	-12331.40199	-2094.785514	-4070.57417	-11520.47005	-2073.2724	-3893.695517	-10738.19613	-1956.643957	-3174.723202	-8298.402458
best val/total time	124	284	426	174	904	16190	101	509	9470	14	33	270	-2094.90723	-4189.72314	-12568.10547	-2094.91333	-4189.51465	-11571.08105	-2094.91382	-4071.18164	-10920.95312	-2057.36377	-3487.45435	-7129.99658
worst													-1963.58154	-3655.05054	-12177.35547	-2094.70605	-3968.57715	-11112.56348	-2060.47095	-3740.46606	-10505.74707	-1828.23181	-2915.87036	-5819.82861
deviation													39.37015	105.8184459	170.1254537	0.06354	53.00383354	116.0652581	20.13076945	68.18752967	134.6434364	55.54267638	148.4233341	362.0609248

4.3 Rastrigin's Function

Rastrigin	Time												Value											
	SA			HCB			HCF			HCW			SA			HCB			HCF			HCW		
Dimensions	5	10	30	5	10	30	5	10	30	5	10	30	5	10	30	5	10	30	5	10	30	5	10	30
avg	7	10.23	17.8	3.4	10.36	319.83	1.9	10.8	200	0.3	0.83	7.13	3.627692333	6.715594333	23.961427	0.464314667	4.047589	27.10301633	1.018841667	5.385014333	36.372605	4.667482667	30.72753533	218.6813173
best val/total time	210	307	534	102	309	9595	57	324	6000	9	25	214	0.99497	0	14.9031	0	2.2308	22.07722	0	2.99499	19.5638	2.95872	25.00438	208.67458
worst													10.8419	10.90839	35.6733	0.99496	5.46666	32.20154	2.2308	7.21065	40.48875	7.64371	37.5783	239.42552
deviation													2.731256165	2.708162439	6.032807791	0.496373258	0.863401586	3.220810022	0.525276479	1.240302846	3.214083178	1.283874158	3.414121087	8.619378371

4.4 Michalewicz's Function

Michalewicz	Time												Value											
	SA			HCB			HCF			HCW			SA			HCB			HCF			HCW		
Dimensions	5	10	30	5	10	30	5	10	30	5	10	30	5	10	30	5	10	30	5	10	30	5	10	30
avg	15	28.3	96.73	6.73	23.0	7.7.33	4.66	22.66	4.4	0.66	23.46	4.36	-4.576512333	-9.396285	-27.89166633	-4.686416667	-9.362743	-26.94174278	-4.684413667	-9.392723333	39.232892	-4.399259	-9.296757667	-41.74751967
best val/total time	450	849	2962	202	693	2.520	140	680	132	20	704	131	-4.68764	-9.53576	-28.68177	-4.68765	-9.50167	-27.29843	-4.68753	-9.49189	26.21414	-4.47426	-9.46686	28.80259
worst													-4.35693	-8.65503	-26.7411	-4.68266	-9.21711	-26.50074	-4.6704	-9.14019	48.86533	-3.9434	-9.07246	52.41682
deviation													0.151367607	0.294364997	0.452805894	0.001441475	0.083003097	0.216006034	0.004318817	0.093737902	6.239712119	0.144979219	0.105724362	4.914559934

5 Conclusion

After analysing the data obtained from testing we can clearly see that in most scenarios Hill Climbing Best Improvement yields the best results for 5 and 10 dimensions, having fairly accurate results and a reasonable run time, however as it is an iterative algorithm, it no longer becomes an optimal solution for a large number of dimensions, as the execution time increases significantly.
At the same time Simulated Annealing dominates the 30 dimension charts as it proves to be fairly accurate while maintaining a pretty good run time which does not increase nearly as much as Hill Climbing algorithms due to it using the Temperature instead of a simple iterative loop.[**SA-HC**]
Overall the least desirable results were generated by Hill Climbing Worst Improvement, although it's pretty fast as an algorithm, the solutions were very inconsistent.

6 References

References

[1] `http://www.geatbx.com/docu/fcnindex-01.html#P140_6155`

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

[3] `https://en.wikipedia.org/wiki/Simulated_annealing`

[4] `https://en.wikipedia.org/wiki/Hill_climbing`

[5] `https://www.youtube.com/watch?v=7Ch8POTPVrU`