[GA] The difference between two heuristic algorithms for determining the minimum of a function: Hill Climbing & Simulated Annealing

Petru-Gabriel Vamanu

October 2022

1 Abstract

One of the basic problems of combinatorial optimization techniques is the computing globally optimal solutions of multivariable functions. The aim of optimization is the finding of optimum values of the objective function through learning the parameters of the function given in the defined domains.

For this purpose, this paper presents how two local search algorithms, a class of heuristics, approximate an optimal solution for the problem of optimizing multivariable functions: Hill Climbing (Best Improvement, First Improvement and Worst Improvement) and Simulated Annealing. Those are tested on several functions and also on several dimensions (5 10 and 30).

2 Introduction

The following paper will present the difference between different implementations of Hill Climbing Algorithm and Simulated Annealing by calculating the minimum value of different mathematical functions. For the demonstration we will analyse 4 mathematical functions:

- De Jong's 1 function
- Schwefel's function
- Rastrigin's function
- Michalewicz's function

The two methods will be described in section 3, explaining the way that the algorithms works, the changes and improvements that were made. The difference between the two approaches will be seen in the running times of the algorithms / the precision of the results. In this section, the representation of possible solutions (real numbers were represented as bit-strings) is also being presented.

The parameters and the environment in which the experiment was ran will be described in section 4. This section also shows information about the functions (definition domain, graph, minimum point and value). The experiment was ran on a precision of $epsilon = 10^{-5}$. For every function the test for every dimension was ran 30 times on the Release mode from Visual Studio (x64 bit) (compiler and compilation mode were chosen for improving the run time). The number of iterations were 5000, 1000 and 500 for 5, 10 and 30 (Hill Climbing) and 4500, 1200 and 250 for 5, 10 and 30 (Simulated Annealing).

The results(section 5) show that Hill Climbing works better on smaller dimensions and the runtime is $\Delta T_{FirstImprovement} < \Delta T_{BestImprovement} < \Delta T_{WorstImprovement}$. Simulated Annealing works better on more dimensions(the runtime on 30 dimensions is smaller than the runtime of 5 dimensions).

The conclusion(section 6) will debate which algorithm is better and what should be done in practise so the best results can be obtained.

3 Methods

3.1 Representation of solutions

The experiment was performed using bitstring data structure. An interval $[a, b]$ will be divided into $N = (b - a) \cdot 10^d$ equal subintervals. In order to be able to represent the $(b - a) \cdot 10^d$ values, it requires a number $n = ceil(log_2 N)$ of bits. The length of the bit string representing a candidate solution will be the sum of the representation lengths for each parameter of the optimization function. When evaluating the solution (the call of the optimization function) it is necessary to decode each parameter represented as a string of bits in real number, according to the formula: $x_{real} = a + x_{decimal}(x_{binary}) \cdot \frac{b-a}{2^{n-1}}$.
A bitstring's neighbor is another bitstring which has exactly one different bit (Hamming Distance is equal to 1).

3.2 Hill Climbing

This method consists on generating a random point(our candidate) from the function's domain of definition which gets improved until it reaches a local minimum(the minimum value which is reachable starting from a point).

This procedure is repeated a number of times (ITERATIONS) and the best result is chosen from the local minimums. Increasing the number of iterations also increases the probability of having the global minimum in the set of local minimums.

```
t := 0
initialize best
repeat
    local := FALSE
    select a candidate solution (bitstring) vc at random
    evaluate vc
    repeat
        vn := Improve(Neghborhood(vc))
        if eval(vn) is better than eval(vc)
        then vc := vn
        else local := TRUE
    until local
    t := t + 1
    if vc is better than best
    then best := vc
until t = MAX
```

Source: [1]

Improving a point can be done in multiple ways; three of them(and the ones used in the experiment) being described below:

3.2.1 First Improvement

This method of improvement searches the first neighbor which has the best function value(which is also an improvement).
In this experiment, instead of trying to switch every bit in order, the order of bits is randomized so the first improvement returns more variable bitstrings.

3.2.2 Best Improvement

This method of improvement searches through all possible neighbors and returns the one which has the best value (i.e. $f(best\_improvement) < f(neighbor)$ for any neighbor of our candidate).

3.2.3 Worst Improvement

This method of improvement searches the neighbor of the candidate which is an improvement(has a better value than our candidate) but it has the smallest improvement from all such points.

3.3 Simulated Annealing

The simulated annealing algorithm is an optimization method which mimics the slow cooling of metals, and how the metal can be easily modeled when being at a higher temperature. In

a similar way, at each virtual annealing temperature, the simulated annealing algorithm generates a new potential solution (or neighbour of the current state) to the problem considered by altering the current state, according to a predefined criterion.

This algorithm is different from the hill climbing algorithm because at some point there is the temperature-dependent probability of visiting solutions that are weaker than the current one, in order to escape from local optima.

This probability is given by:

$$P = \begin{cases} 1 & iff(candidate) \geq f(neighbor) \\ e^{\frac{-|f(candidate)-f(neighbor)|}{temperature}} & iff(candidate) < f(neighbor) \end{cases}$$

At the beginning of the algorithm, which started from a randomly generated candidate, the higher temperature and higher probability of acceptance of new solutions allowed to explore a wide region of the search space, thus escaping from a minimal location; however, as the temperature was reduced, the probability of acceptance of unfavourable solutions was reduced.

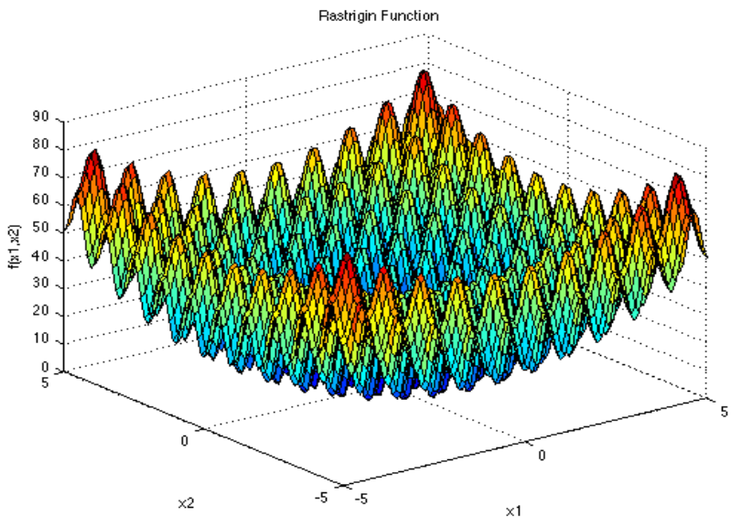
```
t := 0
initialize the temperature T
select a current candidate solution (bitstring) vc at random
evaluate vc
repeat
    repeat
        select at random vn: a neighbor of vc
        if eval(vn) is better than eval(vc)
            then vc := vn
            else if random[0,1) < exp(-|eval(vn) - eval(vc)| / T)
                then vc := vn
    until (termination-condition)
    T := g(T; t)
    t := t + 1
until (halting-criterion)
```

Source: [1]

4 Experimental set-up

4.1 Rastrigint Function

The Rastrigin function has several local minima. It is highly multimodal, but locations of the minima are regularly distributed. It is shown in the plot above in its two-dimensional form.

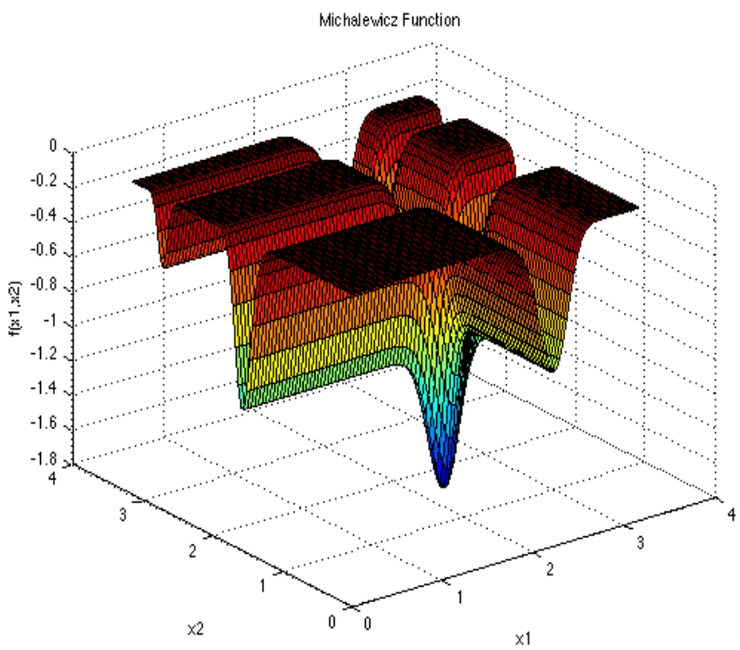


$$f(x) = 10d + \sum_{i=1}^{d} [x_i^2 - 10cos(2\pi x_i)]$$

$$-5.12 <= x_i <= 5.12$$

4.2 Michalewicz's Function

The Michalewicz function has d! local minima, and it is multimodal. The parameter m defines the steepness of they valleys and ridges; a Larger m leads to a more difficult search. The recommended value of m is m = 10.

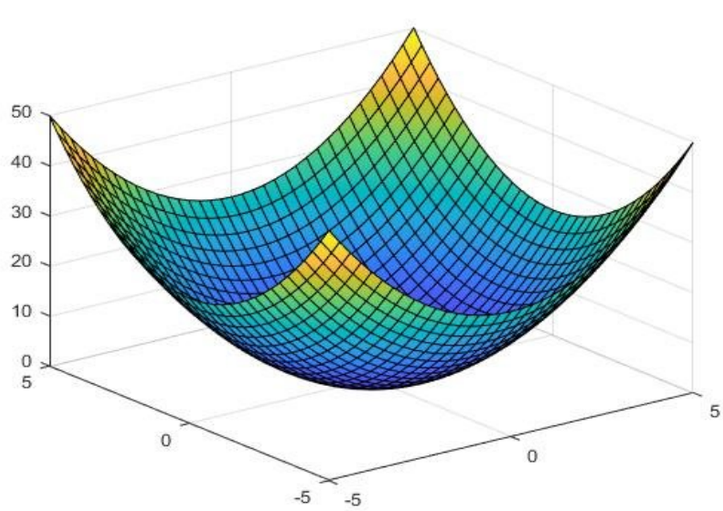


$$f(x) = -\sum_{i=1}^{d}[sin(x_i)sin^{2m}(\frac{ix_i^2}{\pi})]$$

$$0 <= x_i <= \pi$$

4.3 De Jong's Function

The simplest test function is De Jong's function 1. It is also known as sphere model. It is continuos, convex and unimodal.

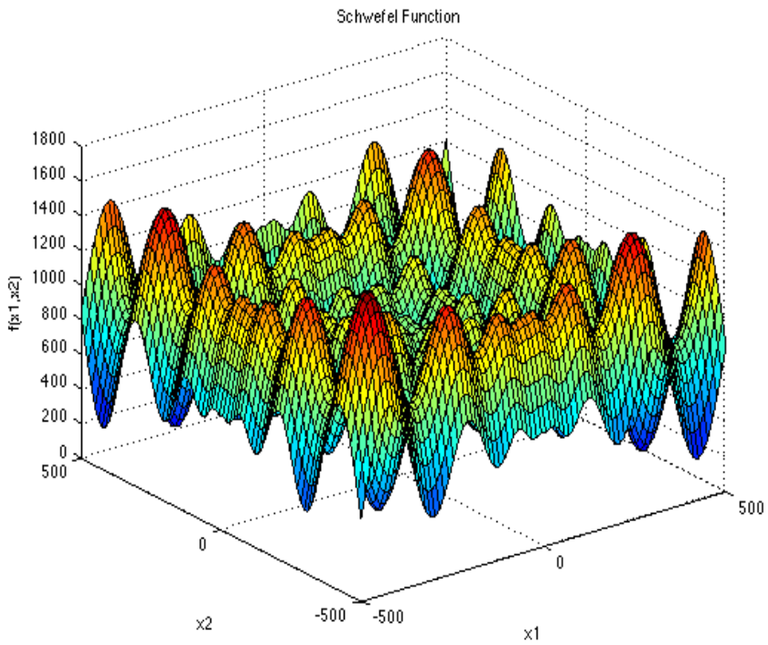


$$f(x) = \sum_{i=1}^{d} x_i^2$$

$$-5.12 <= x_i <= 5.12$$

4.4 Schwefel's Function

Schwefel's function is deceptive in that the global minimum is geometrically distant, over the parameter space, from the next best local minima. Therefore, the search algorithms are potentially prone to convergence in the wrong direction.



$$f(x) = 418.9829 \cdot d - \sum_{i=1}^{d} -x_i \cdot \sin\left(\sqrt{|x_i|}\right)$$

$$-500 <= x_i <= 500$$

4.5 Experiment Description

The dimensions used for running the program were 5, 10 and 30. As early mentioned each function was tested 30 times. The precision chosen was 5 for decimal results. The number of iterations for the Hill Climbing algorithm are:

- 10000 for 5 dimensions
- 5000 for 10 dimensions
- 500 for 30 dimensions

For the Simulated Annealing algorithm the number of iterations are:

- 4500 for 5 dimensions
- 1100 for 10 dimensions
- 300 for 30 dimensions

The initial temperature for the Simulated Annealing algorithm was set at 10000 and the cooling process was set $temperature = temperature \cdot 0.7$, and the halting criterion was $temperature > 0.000000019$ which means about 70 coolings. The termination condition was to make 50 downgrades or to pass a neighbour without improving or downgrading.

5 Results

5.1 Hill Climbing

The following tables show the results of the Hill Climbing method:

5.1.1 Michalewicz's Function

Deterministic algorithm showed that the minimum is found:

$$\text{for 5 dimensions} : f(x) = -4.687658$$

$$\text{for 10 dimensions} : f(x) = -9.66015$$

$$\text{for 30 dimensions} : f(x) = unknown$$

Results for First Improvement:

dimensions	Min.	1st Qu.	Mean	Median	3rd Qu.	Max.
5	-4.6877	-4.6876	-4.6876	-4.6875	-4.6873	-4.6867
10	-9.5841	-9.4539	-9.3891	-9.3927	-9.3053	-9.2103
30	-26.8040	-25.5430	-25.2429	-25.3635	-25.0672	-24.6781

Results for Best Improvement:

dimensions	Min.	1st Qu.	Mean	Median	3rd Qu.	Max.
5	-4.6877	-4.6877	-4.6877	-4.6876	-4.6877	-4.6873
10	-9.6108	-9.5517	-9.4775	-9.4877	-9.4280	-9.3856
30	-27.5216	-27.1260	-26.8478	-26.9001	-26.6725	-26.3817

Results for Worst Improvement:

dimensions	Min.	1st Qu.	Mean	Median	3rd Qu.	Max.
5	-3.9701	-3.4582	-3.2864	-3.3225	-3.1929	-2.9227
10	-5.5069	-4.8493	-4.6106	-4.6366	-4.3942	-4.0814
30	-9.7352	-8.1553	-7.8347	-7.8738	-7.4277	-6.6959

5.1.2 De Jong's Function

The minimum is found on $f(0, 0, ..., 0) = 0$ for any dimension.

Results for First Improvement:

dimensions	Min.	1st Qu.	Mean	Median	3rd Qu.	Max.
5	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
10	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
30	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Results for Best Improvement:

dimensions	Min.	1st Qu.	Mean	Median	3rd Qu.	Max.
5	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
10	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
30	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Results for Worst Improvement:

dimensions	Min.	1st Qu.	Mean	Median	3rd Qu.	Max.
5	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
10	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
30	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

5.1.3 Rastrigin's Function

The minimum is 0 for any dimension.

Results for First Improvement:

dimensions	Min.	1st Qu.	Mean	Median	3rd Qu.	Max.
5	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
10	2.2308	3.9849	4.2258	4.2153	4.6473	6.6974
30	24.344	40.836	43.472	43.271	46.436	51.738

Results for Best Improvement:

dimensions	Min.	1st Qu.	Mean	Median	3rd Qu.	Max.
5	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
10	0.9950	2.2308	2.9874	2.8153	3.2308	4.4616
30	21.637	27.685	29.603	29.047	31.005	34.186

Results for Worst Improvement:

dimensions	Min.	1st Qu.	Mean	Median	3rd Qu.	Max.
5	0.00000	1.00001	1.00001	1.04119	1.00001	2.00002
10	4.0000	5.9999	6.2359	6.2511	7.0001	7.9993
30	40.949	44.998	47.202	47.431	49.665	52.553

5.1.4 Schwefel's Function

The minimum is 0 for any dimension.

Results for First Improvement:

dimensions	Min.	1st Qu.	Mean	Median	3rd Qu.	Max.
5	0.000688	0.001946	0.104368	0.087491	0.104989	0.208646
10	61.431	130.005	153.578	160.036	190.291	272.023
30	1526.50	1732.63	1810.31	1798.82	1887.38	1974.93

Results for Best Improvement:

dimensions	Min.	1st Qu.	Mean	Median	3rd Qu.	Max.
5	0.00007	0.00009	0.00069	0.00061	0.00070	0.00194
10	0.107	0.623	30.975	25.594	34.652	119.345
30	868.69	1214.03	1324.04	1312.11	1434.96	1572.99

Results for Worst Improvement:

dimensions	Min.	1st Qu.	Mean	Median	3rd Qu.	Max.
5	26.884	53.767	53.819	56.098	53.974	80.650
10	188.182	215.065	226.094	226.675	241.947	268.830
30	1661.65	1799.18	1883.11	1862.80	1923.79	1990.34

5.2 Simualted Annealing

5.2.1 Michalewicz's Function

Deterministic algorithm showed that the minimum is found:

$$\text{for 5 dimensions} : f(x) = -4.687658$$
$$\text{for 10 dimensions} : f(x) = -9.66015$$
$$\text{for 30 dimensions} : f(x) = unknown$$

dimensions	Min.	1st Qu.	Mean	Median	3rd Qu.	Max.
5	-4.6877	-4.6876	-4.6873	-4.6873	-4.6870	-4.6863
10	-9.3876	-9.3017	-9.2076	-9.2051	-9.1547	-8.9030
30	-26.1603	-25.4302	-25.1375	-25.2299	-24.9667	-24.6161

5.2.2 De Jong's Function

The minimum is found on $f(0, 0, ..., 0) = 0$ for any dimension.

dimensions	Min.	1st Qu.	Mean	Median	3rd Qu.	Max.
5	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
10	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
30	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

5.2.3 Rastrigin's Function

The minimum is 0 for any dimension.

dimensions	Min.	1st Qu.	Mean	Median	3rd Qu.	Max.
5	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
10	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
30	6.9900	13.9180	14.9800	14.6747	16.6569	18.6155

5.2.4 Schwefel's Function

The minimum is 0 for any dimension.

dimensions	Min.	1st Qu.	Mean	Median	3rd Qu.	Max.
5	0.0011	0.1050	0.2077	1.9902	0.3123	27.0904
10	54.08	205.90	258.36	234.95	281.61	322.26
30	1485.00	1753.22	1883.18	1836.27	1927.10	1996.81

5.3 Comparison

The results show that Simulated Annealing has the best results of all methods. Though Best Improvement matches its performance on some instances.
From the Hill Climbing improvements the one which gives the best results is Best Improvement, but First Improvement is not so far. Worst Improvement on the other hand, gives worse results, but there are some instances when it gets close to the other two improvements.

The mean of the runtimes of all the algorithms are:

Function	De Jong				Rastrigin			
	Best	First	Worst	S.A	Best	First	Worst	S.A
5	78.536	4.9875	19.171	145.88	9.7141	5.9422	56.74	132.81
10	290.64	9.1957	58.836	92.071	29.525	10.051	185.34	97.866
30	1086.1	17.0856	148.2	144.08	65.786	18.4629	420.29	161.58
Function	Michalewicz				Schwefel's			
	Best	First	Worst	S.A	Best	First	Worst	S.A
5	13.271	6.3213	0.35143	500.42	44.282	21.347	300.11	47.135
10	37.205	10.024	0.56119	365.86	133.98	39.601	982.12	35.031
30	80.581	18.452	0.42432	687.41	173.58	17.188	1152.7	68.333

As it is shown in the tables Simulated Annealing works better on more dimensions because it isn't a big difference between the runtime for 5 dimensions and 30 dimensions. Depending on the instance and the HC improvement used, it can be faster than Hill Climber(especially the Worst Improvement).

For Hill Climbing the First Improvement is the fastest one, which also gives decent results. Best Improvement gives better results but it is a bit slower. Worst Improvement is generally the slowest one which also gives the worst results. Even though it may seems inferior to Best and First Improvements, there are instances when it gives better results and faster (De Jong's on 5 dimensions).

6 Conclusions

Simulated Annealing can be considered as a modification of Hill Climbing. Hill Climbing attempts to reach an optimum value by checking if it's current state has the best cost/score in its neighborhood, this makes it prone to getting stuck in local optima. Simulated Annealing attempts to overcome this problem by choosing a "bad" move every once in a while. The probability of choosing of a "bad" move decreases as time moves on, and eventually, Simulated Annealing becomes Hill Climbing.

Also, depending on how the temperature is set, Simulated Annealing can be better than Hill Climbing. When setting a small temperature and a small number of downgrades Simulated Annealing behaves just like the First Improvement method.

Another thing that the report shows is that using same parameters (such as Number of iterations or temperature) on different functions doesn't guarantee that all functions will have good results. For examples, Schwefel's Function has a small runtime on Simulated Annealing compared to the other functions. If it had other parameters, it could have the same runtime and maybe better results. This shows that every function should be parameterized differently, in order to get the best result.

In conclusion, both algorithms are a good way of approximating the minimum of a function, with all of their forms. There isn't a "best algorithm" in general, but for a specific function there is always an algorithm in a specific form with specific parameters will have the best performance in practise and it is just a matter of time and effort until it is found. Another idea could be that we run the algorithm for a certain number of seconds, getting rid of the number of iterations, so each pair algorithm/function can run a different number of iterations, depending on how fast an iteration is computed.

References

[1] GA Laboratory Page
The Algorithms for HC and SA
https://profs.info.uaic.ro/~eugennc/teaching/ga/#Notions02

[2] Rastrigin's Function rendered image.
https://commons.wikimedia.org/wiki/Main_Page

[3] Informations and images of tested functions.
http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.html
http://www.geatbx.com/docu/fcnindex-01.html#P89_3085

[4] Virtual Library of Simulation Experiments:
Test Functions and Datasets
Informations and graphics of implemented functions
https://www.sfu.ca/~ssurjano/

[5] Metaheuristics in Combinatorial Optimization
http://www.lia.deis.unibo.it/~aro/pubs/blum_roli_metaheuristics-preprint.pdf

[6] Overleaf - Learn LaTeX with examples
https://www.overleaf.com/learn/latex/

http://www.malinc.se/math/latex/basiccodeen.php

[7] Table generator tool for LaTeX
https://www.tablesgenerator.com

[8] Nourani, Y., Andresen, B. (1998). A comparison of simulated annealing cooling strategies. Journal of Physics A: Mathematical and General, 31(41), 8373.

https://iopscience.iop.org/article/10.1088/0305-4470/31/41/011/meta

[9] Bertsimas, D., Tsitsiklis, J. (1993). Simulated annealing. Statistical science, 8(1), 10-15.

https://projecteuclid.org/journals/statistical-science/volume-8/issue-1/Simulated-Annealing/10.1214/ss/1177011077.short