Finding the global minimum of a set of given function using Hill-Climbing and Simulated-Annealing algorithms

Sau Ionela Alexandra

November 2022

I. Abstract

The report presents the Hill-Climbing (Best Improvement and First Improvement) and Simulated-Annealing algorithms implemented to find the global minimum of a set of functions: De Jong, Rastrigin, Schwefel and Michalewicz, for 3 different dimensions: 5, 10 and 30. The report contains explanations about algorithms, values used for experiments, tables with results, comparations and conclusions. Hill-Climbing Best Improvement offers the best results, but it also take the longest to find. Hill-Climbing First Improvement is two times faster, but provides the worst answers and Simulated Annealing is way faster than the other two, and also obtain pretty close results to them. Hill-Climbing Algorithms work better with functions that have a small number of local minima, while Simulated-Annealing Algorithm performs better for functions with many local minima.

II. Introduction

1. General introduction

This report presents the differences obtained by implementing 3 non-deterministic algorithms: Hill-Climbing Best Improvement, Hill-Climbing First Improvement and Simulated-Annealing, with the aim of finding the global minimum of the functions: De Jong, Rastrigin, Schwefel and Michalewicz, on 3 dimensions: 5, 10 and 30.

2. Motivation

The implementation and study of non-deterministic algorithms is very important because there are many problems that cannot be solved in a timely manner using deterministic algorithms.

3. Problem description

Using the Hill-Climbing and Simulated-Annealing algorithms, we can solve the global minimum problem for De Jong, Rastrigin, Schwefel and Michalewicz.

III. Methods

1. The algorithms used

<u>Hill-Climbing algorithm</u> is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution

to the problem. It terminates when it reaches a peak value where no neighbor has a higher value.

Hill-Climbing First Improvement examines the neighboring values one by one and select the first neighboring value which optimizes the current cost as the next node.

Hill-Climbing Best Improvement first examines all the neighboring values and then selects the value closest to the solution state as of the next value.

Simulated-Annealing does not examine all the neighboring values before deciding which node to select. It just selects a neighboring value at random (changing one bit at random) and decides (based on the amount of improvement in that neighbor) whether to move to that neighbor or to examine another.

All 3 algorithms use bit strings to represent candidate solution. The neighborhood of the candidate solution is represented by each string of bits obtained by changing a single bit at time.

First, at each 3 it initializes best with INT_MAX and t with 0. Then, it iterates 500 times, and after each iterations it compares candidate solution with best. For Simulated-Annealing, it initializes temperature with 100, and then multiply it by a subunit value (0.99) until the temperature is below 10^-9.

2. Experiments description

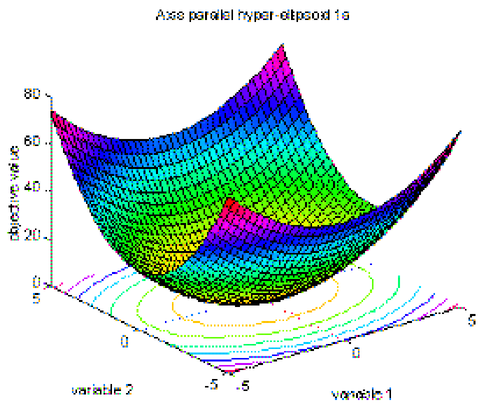
The dimensions for which the algorithms were applied are: 5, 10 and 30. The precision chosen for this experiment is 5 decimal places. The number of repetitions for each algorithm is 500.

IV. Experimental results

The results are presented in the form of tables. In each table it mentions the minimum, the maximum and the average of the solutions, the time and mean time and standard deviations.

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.



Function definition:

$$f_1(x) = \sum_{i=1}^{n} x_i^2 \qquad -5.12 \le x_i \le 5.12$$

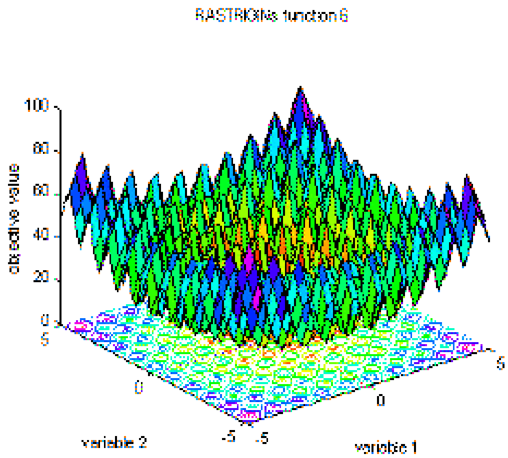
f_1(x)=sum(x(i)^2), i=1:n, -5.12<=x(i)<=5.12.

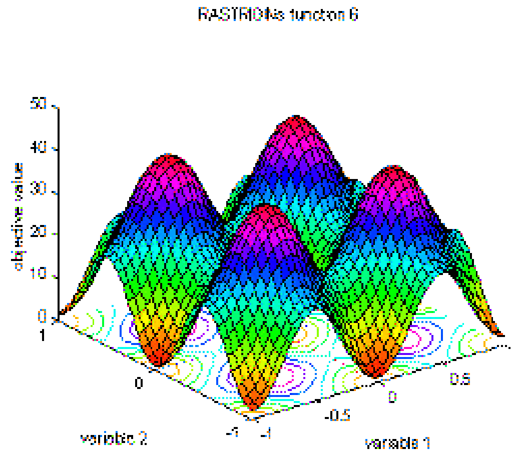
Global minimum: f(x)=0, x(i)=0, i=1:n

Algorithm	Dimension	Minim	Maxim	Average	Time	Mean Time	Deviation
HCB	5	0	0	0	412.976	13.76587	0
HCB	10	0	0	0	2496.618	83.2206	0
HCB	30	0	0	0	42390.2	1413.007	0
HCF	5	0	0	0	230.325	7.6775	0
HCF	10	0	0	0	1426.114	47.53713	0
HCF	30	0	0	0	22147.77	738.2589	0
SA	5	0	0	0	247.671	8.2557	0
SA	10	0	0	0	414.358	13.81193	0
SA	30	0	0	0	705.228	23.5076	0

Rastrigin 's Function

Rastrigin's function is based on function 1 with the addition of cosine modulation to produce many local minima. Thus, the test function is highly multimodal. However, the location of the minima are regularly distributed.





Function definition:

$$f_6(x) = 10 \cdot n + \sum_{i=1}^{n} \left(x_i^2 - 10 \cdot \cos(2 \cdot \pi \cdot x_i)\right) \qquad -5.12 \le x_i \le 5.12$$

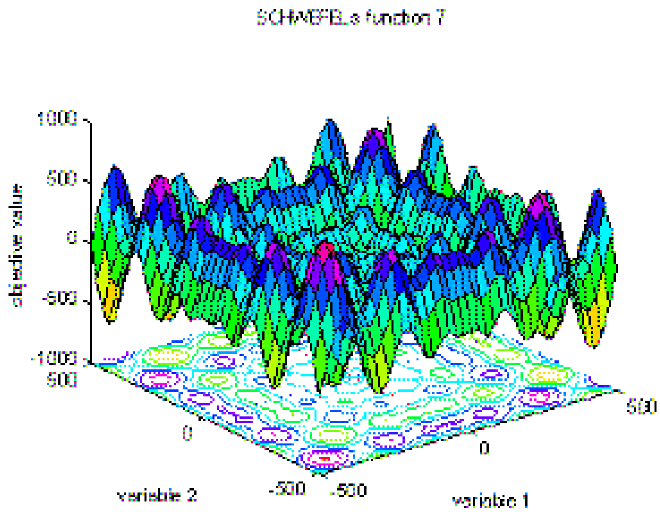
f_6(x)=10·n+sum(x(i)^2-10·cos(2·pi·x(i))), i=1:n; -5.12<=x(i)<=5.12.

Global minimum: f(x)=0; x(i)=0, i=1:n.

Algorithm	Dimension	Minim	Maxim	Average	Time	Mean Time	Deviation
HCB	5	0	1.23582	0.74636	559.559	18.65197	0.453936
HCB	10	2.98993	6.45653	4.72983	3067.66	102.2553	0.84127
HCB	30	20.6256	33.1963	28.53315	35253.47	1174.782	2.93313
HCF	5	0	2.23078	1.28277	212.918	7.09726	0.57705
HCF	10	4.22575	8.68226	6.02050	1287.006	42.9002	1.41440
HCF	30	30.8514	45.9653	38.11441	19790.83	659.6942	3.4729
SA	5	0.00819	3.23624	1.39702	284.286	9.4762	0.74263
SA	10	2.04949	11.8475	6.03797	360.8	12.025667	2.02791
SA	30	17.8664	38.6578	26.27384	765.127	25.50423	6.00089

Schwefel 's Function

Schwefel's function [Sch81] 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.



Function definition:

$$f_7(x) = \sum_{i=1}^{n} -x_i \cdot \sin\left(\sqrt{|x_i|}\right) \qquad -500 \le x_i \le 500$$

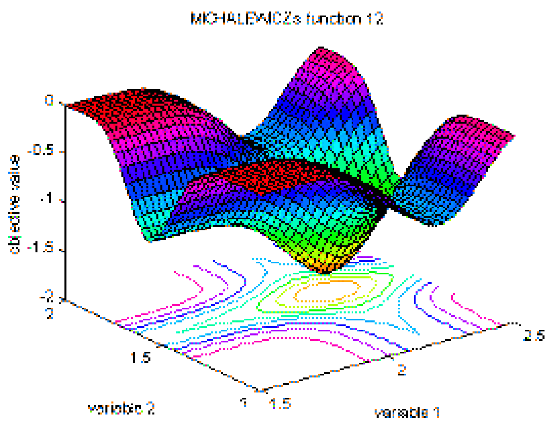
$f_7(x)$=sum(-x(i)·sin(sqrt(abs(x(i))))), i=1:n; -500<=x(i)<=500.

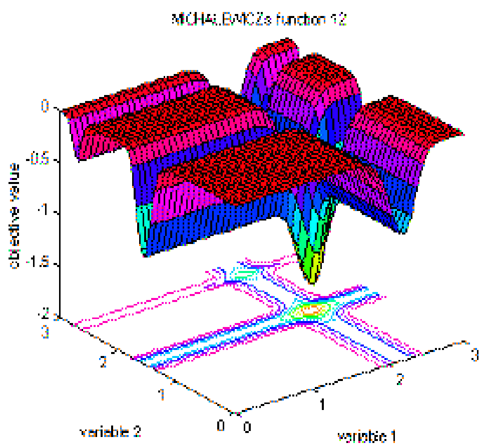
Global minimum: f(x)=-n·418.9829; x(i)=420.9687, i=1:n.

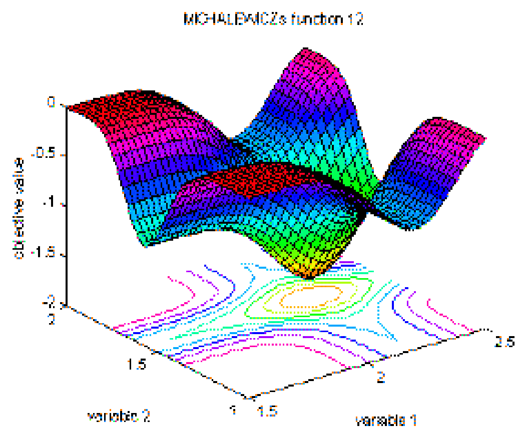
Algorithm	Dimension	Minim	Maxim	Average	Time	Mean time	Deviation
HCB	5	-2094.91	-2094.6	-2094.76	1105.694	36.85647	0.10903
HCB	10	-4155.59	-3914.28	-4031.6	5509.96	183.6653	71.52654
HCB	30	-11561	-10972.8	-11245.6	76807.9	2560.263	149.6116
HCF	5	-2094.81	-1963.58	-2041.3	444.674	14.82247	41.80897
HCF	10	-4023.95	-3706.43	-3861.95	2504.837	83.49457	68.35787
HCF	30	-21516.7	-10513.8	-11092.8	44275.06	1475.835	1941.254
SA	5	-2094.91	-2086.09	-2093.54	256.532	8.55106	2.25694
SA	10	-4189.52	-4049.27	-4161.17	400.209	13.3403	40.48079
SA	30	-12533.7	-11743.9	-12238.3	842.406	28.0802	168.4887

Michalewicz ‘s Function

The Michalewicz function [Mic92] is a multimodal test function (n! local optima). The parameter m defines the "steepness" of the valleys or edges. Larger m leads to more difficult search. For very large m the function behaves like a needle in the haystack (the function values for points in the space outside the narrow peaks give very little information on the location of the global optimum).







Function definition:

$$f_{12}(x) = -\sum_{i=1}^{n} \sin(x_i) \cdot \left(\sin\left(\frac{i \cdot x_i^2}{\pi} \right) \right)^{2 \cdot m} \qquad i = 1:n,\ m = 10,\ 0 \le x_i \le \pi$$

f12(x)=-sum(sin(x(i))·(sin(i·x(i)^2/pi))^(2·m)), i=1:n, m=10, 0<=x(i)<=pi.

Global minimum: f(x)=-4.687 (n=5); x(i)=???, i=1:n.

f(x)=-9.66 (n=10); x(i)=???, i=1:n.

Algorithm	Dimension	Minim	Maxim	Average	Time	Mean time	Deviation
HCB	5	-4.68766	-4.66925	-4.68508	351.051	11.7017	0.00347
HCB	10	-9.48767	-9.07983	-9.29414	2064.508	68.81693	0.10129
HCB	30	-27.4514	-26.3447	-26.7509	31708.8	1056.96	0.28031
HCF	5	-4.68707	-4.64343	-4.67728	195.126	6.5042	0.01099
HCF	10	-9.49299	-8.90714	-9.17366	1224.311	40.81037	0.10630
HCF	30	-26.7597	-25.6211	-26.1949	19790.83	593.4237	0.28769
SA	5	-4.68766	-4.49466	-4.64236	270.479	9.01596	0.04317
SA	10	-9.61093	-9.03486	-9.30006	366.837	12.2279	0.16063
SA	30	-28.4653	-27.1648	-27.9044	799.615	26.65383	0.35712

V. Comparations

It can be seen that Hill-Climbing Best Improvement offers, in the most cases, the best results, but also it's execution takes the longest. Hill-Climbing First Improvement is usually two times faster than Hill-Climbing Best Improvement but it provides worse answers. Simulated-Annealing offers good results but with a larger standard deviation between solutions, because it choose the bits that must be changed randomly, thus having a larger search area than the other two algorithms. Also, Simulated-Annealing has the fastest executions from all three. While the Hill-Climbing Best Improvement took the most time, the other variant, First-Improvement, took half of the time. While the time these two take can be measured in tens of minutes, even hours, Simulated Annealing barely took some minutes. With the chosen conditions, the difference of time between Hill-Climbing algorithms and Simulated Annealing, is immense, and as seen earlier, the difference in results does not really justify the time difference.

VI. Conclusions

From the results of the three selected methods, it can be seen that Simulated Annealing always has a higher standard deviation, because the temperature mechanism allows a larger search range, resulting in more spread results. Thus it can be observed that for functions with many local minima it's better to use a more chaotic method such as Simulated Annealing, while functions with a few local minima works better with a more predictable method, such as Hill-Climbing. In conclusion, Simulated Annealing should be used for more dispersed results, while Hill-Climbing algorithms are better suited for more congregated results.

VII. References

a) http://www.cplusplus.com/reference/cstdlib/rand/
b) http://www.geatbx.com/docu/fcnindex-01.html#P204_10395

c) https://profs.info.uaic.ro/~eugennc/teaching/ga/
d) https://towardsdatascience.com/unit-1-hill-climber-optimization-985d5b79bd5
e) https://www.baeldung.com/cs/hill-climbing-algorithm
f) https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/
g) https://en.wikipedia.org/wiki/Hill_climbing
h) https://www.coursehero.com/file/p2gsdqlh/Difference-between-Simple-Hill-Climbing-and-Steepest-Ascent-Hill-Climbing/
i) https://stackoverflow.com/questions/9878965/rand-between-0-and-1
j) https://www.javatpoint.com/hill-climbing-algorithm-in-ai