Study of functions using Hill Climbing and Simulated Annealing algorithms

Mihalache Radu-Stefan

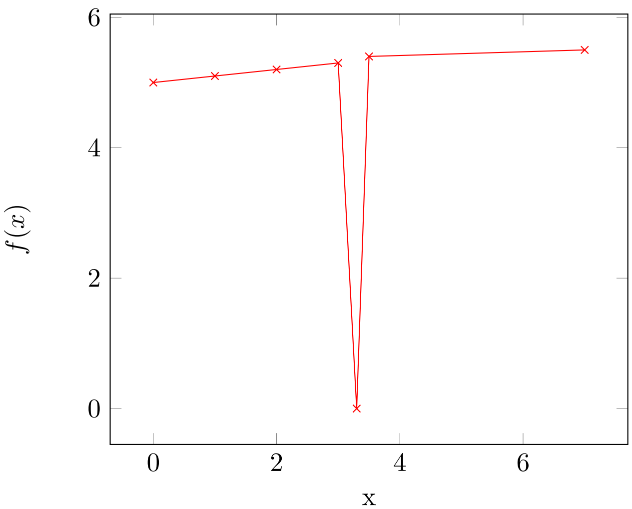
0.1 Abstarct

Analyzing Simulated Annealing and Hill Climbing algorithms to evaluate mathematical functions on multiple dimensions and find the minimum point on specific intervals.

0.2 Introduction

0.2.1 Motivation

The problem of exploring the values of functions and finding the global minimum of said function for a specified domain has useful aplications, yet it is dificult to solve with a deterministic algorithm. That is because some functions have a very steep path to the minimum, like in the example below.



0.3 Method

Nondeterministic algorithms can be used to overcome this problem, as they have a better chance to explore the function and find the minimum .
The representation of the input variables will be a string of n bits such that they can accuately represent the function domain.

$$x = a + decimal_{represenation}(bit_{str}) \cdot (b - a)/(2^n - 1), x \in [a, b]$$

Using this represenattion, a random input called candidate solution can be generated, and its vecinity can be explored by negating one bit, such that the hamming distance between the candidate solution and the vecinity is one. This leads to the following aproaches:

Hill Climbing:

Select a candidate solution for each iteration and try to improove it using either the first better vecinity or the best vecinity. This algorithm finds the minimumm by exploring the basin of the candidate solution.

Simulated annealing:

Select a candidate solution at the start and explore its vecinity. This algorithm better explores the domain of the function by choosing worse vecinities base on the probability given by this expression:

$$random.uniform(0, 1) < math.exp(-abs((evaln - evalc)/temperature))$$

This algorithm makes use of the hot iron concept. At the begining the temperature is high and the chance to choose a worse solution is high but it decreeses over each iteration based on this formula:

$$temperature = temperature * 0.9$$

0.4 Experiment

For this experiment, a python program will analyse theese functions on 5, 10 and 30 dimensions with 100 iterations and 10^{-5} precissionn. Each test is run 30 times to ensure consistancy

[1]https://al-roomi.org/benchmarks/unconstrained/n-dimensions/
[2]https://al-roomi.org/component/tags/tag/schwefel-function
[3]https://commons.wikimedia.org/wiki/MainPage
[4]https://www.sfu.ca/ ssurjano/michal.html

$$f(x) = \sum_{i=1}^{n} \left[x_i^2 \right], x_i \in [-5.12, 5.15]$$

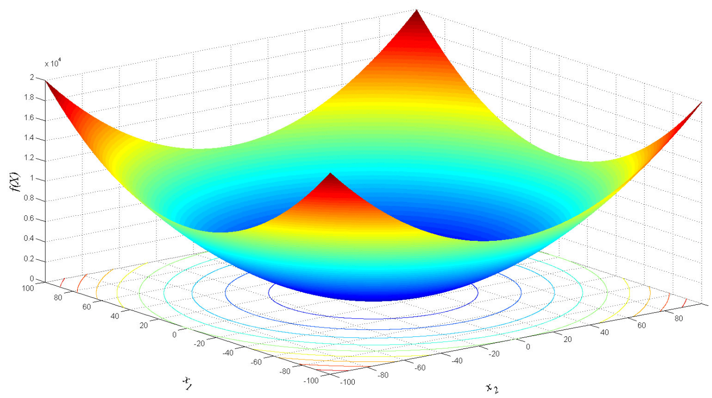


Figure 1: Image De Jong's Function.[1]

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

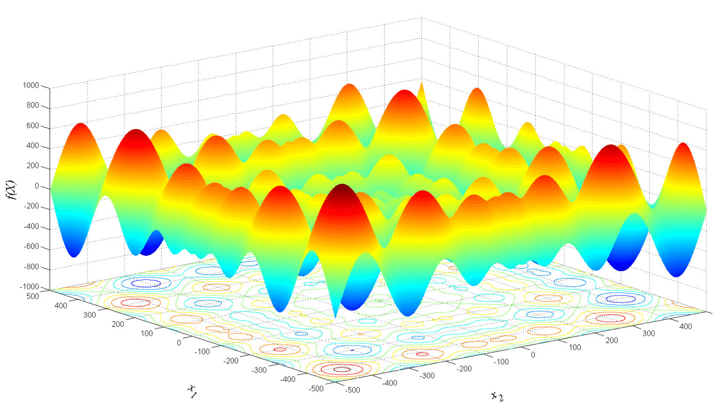


Figure 2: Image Schwefel's Function. [2]

$$f(x) = A \cdot n + \sum_{i=1}^{n} \left[x_i^2 - A \cdot cos(2\pi x_i) \right], A = 10, x_i \in [-5.12, 5.15]$$

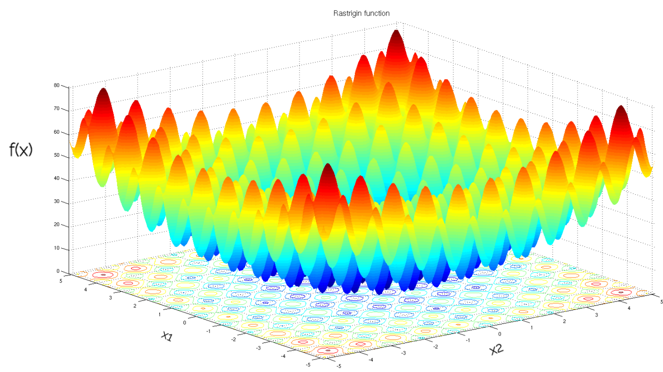


Figure 3: Image Rastrigin's Function. [3]

$$f(x) = -\sum_{i=1}^{n} \left[sin(x_i) \cdot \left(sin \left(\frac{i \cdot x_i^2}{\pi} \right) \right)^{2 \cdot m} \right], x_i \in [0, \pi], m = 10$$

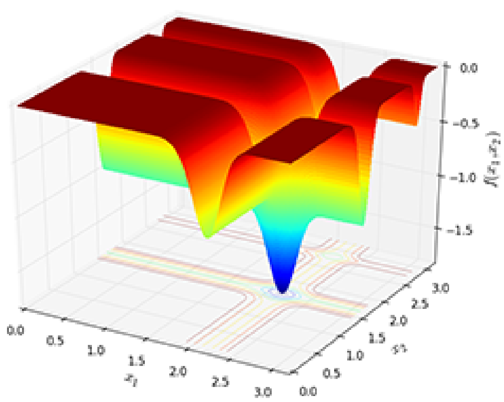


Figure 4: Michalewicz's Function. [4]

0.5 Results

Hill Climbing first 5D

Functions	Solutions			Time Averege
	Averege	Minimum	Maximum	
DeJong 1	0	0	0	14
Schwefel	-1939	-1992	-1853	22
Rastrigin	2.29497	1.58367	3.39702	26
Michalewicz	-4.44109	-4.62793	-3.97123	22

Hill Climbing best 5D

Functions	Solutions			Time Averege
	Averege	Minimum	Maximum	
DeJong 1	0	0	0	28
Schwefel	-1967	-2094	-1893	20
Rastrigin	1.93592	1.279144	1.68702	28
Michalewicz	-4.37306	-4.65685	-4.02012	20

Simulated Annealing 5D

Functions	Solutions			Time Averege
	Averege	Minimum	Maximum	
DeJong 1	0	0	0	4
Schwefel	-1825	-1963	-1776	6
Rastrigin	2.23086	1.71005	3.102193	6
Michalewicz	-4.28916	-4.6328	-3.26917	4

Hill Climbing first 10D

Functions	Solutions			Time Averege
	Averege	Minimum	Maximum	
DeJong 1	0	0	0	112
Schwefel	-3652	-3872	-3492	262
Rastrigin	7.18790	3.93217	15.19234	322
Michalewicz	-8.20880	-9.01029	-6.89132	214

Hill Climbing best 10D

Functions	Solutions			Time Averege
	Averege	Minimum	Maximum	
DeJong 1	0	0	0	216
Schwefel	-3623	-3772	-3506	166
Rastrigin	8.18790	3.67834	14.0925	242
Michalewicz	-9.09037	-9.24031	-8.25910	158

Simulated Annealing 10D

Functions	Solutions			Time Averege
	Averege	Minimum	Maximum	
DeJong 1	0	0	0	8
Schwefel	-3642	-3905	-3246	14
Rastrigin	9.18790	3.19593	15.2394	14
Michalewicz	-8.70903	-9.22690	-6.15737	12

Hill Climbing first 30D

Functions	Solutions			Time Averege
	Averege	Minimum	Maximum	
DeJong 1	0	0	0	1946
Schwefel	-9453	-9923	-9369	3051
Rastrigin	33.72184	27.12235	48.63187	3234
Michalewicz	-20.79776	-22.61219	-17.31592	1260

Hill Climbing best 30D

Functions	Solutions			Time Averege
	Averege	Minimum	Maximum	
DeJong 1	0	0	0	2570
Schwefel	-10153	-10453	-10063	3022
Rastrigin	32.43780	26.97201	47.29053	2784
Michalewicz	-24.09166	-25.14376	-22.21472	846

Simulated Annealing 30D

Functions	Solutions			Time Averege
	Averege	Minimum	Maximum	
DeJong 1	0	0	0	20
Schwefel	-9746	-11279	-9365	40
Rastrigin	35.18790	27.67834	56.44140	42
Michalewicz	-23.66259	-24.29173	-21.53991	30

0.6 Conclusions

In my resaults, Hill Climbing best improovement usualy gives slightly better resaults than first improovement while neither algorithm is clearly better in terms of time.
As expected, with larger input, the time increses significantly, and the difference between the minimum and the maximum is higher, which could simbolize a lower precission.
The resaults could be improoved by using a different random function and the time could be improoved by implmenting the program in C++ rather than Python.
In my implementation the Simulated Annealing algorithm, gives worse resaults than Hill Climbing with higher difference between the minimum and maximum,

but the time is more than 10 times better, and the increese in input size doesn't increese the time at the same rate as the Hill Climbing.
The algorithm could be improoved by using a better temperature function, and perhaps finding a way to make it more efficient at low temperatures.

As demonstrated in the experiment, nondeterministic algorithms can be used to find the global minimum of functions.

Bibliography

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

[2] Al-roomi
De Jong's Function rendered image. https://al-roomi.org/benchmarks/unconstrained/n-dimensions/ Al-roomi
Schwefel's Function rendered image. https://al-roomi.org/component/tags/tag/schwefel-function

[3] Sfu
Michalewicz's Function rendered image. https://www.sfu.ca/~ssurjano/michal.html

[4] Geatbxi
Function formulas. http://www.geatbx.com/docu/fcnindex-01.html

[5] Course Page. https://profs.info.uaic.ro/~eugennc/teaching/ga/l

[6] Pycharm. https://www.jetbrains.com/pycharm/