

Study of functions using Hill Climbing and Simulated Annealing algorithms

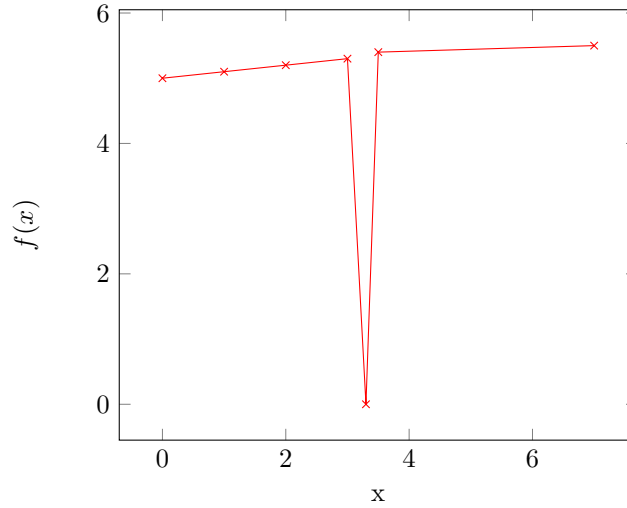
Mihalache Radu-Stefan

November 3, 2021

0.1 Introduction

0.1.1 Motivation

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



0.2 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 accurately represent the function domain.

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

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

Hill Climbing:

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

Simulated annealing:

Select a candidate solution at the start and explore its vicinity. This algorithm better explores the domain of the function by choosing worse vicinities 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 decreases over each iteration based on this formula:

$$temperature = temperature * 0.9$$

0.3 Experiment

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

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

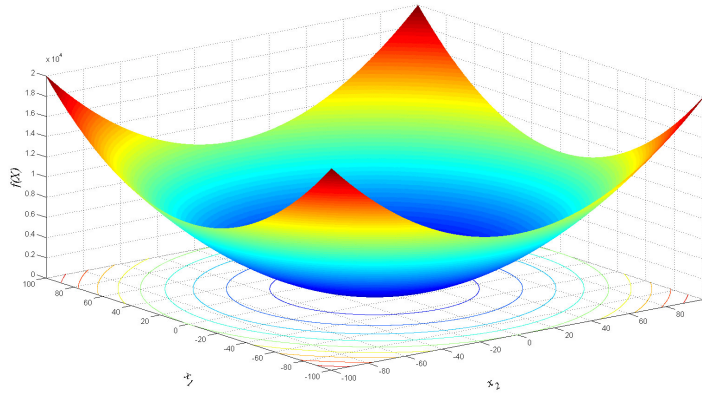


Figure 1: Image De Jong's Function.¹

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

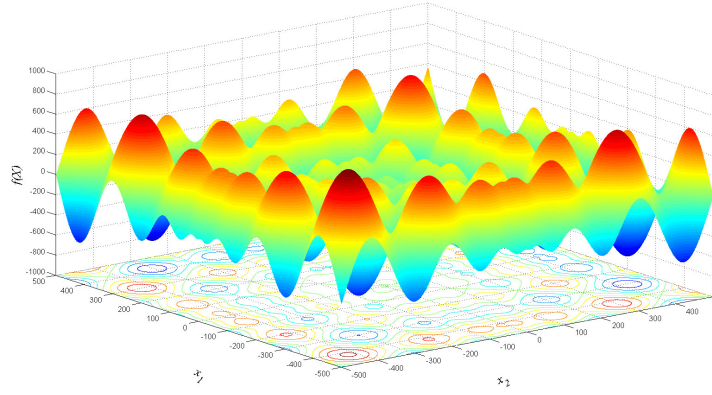


Figure 2: Image Schwefel's Function. ²

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

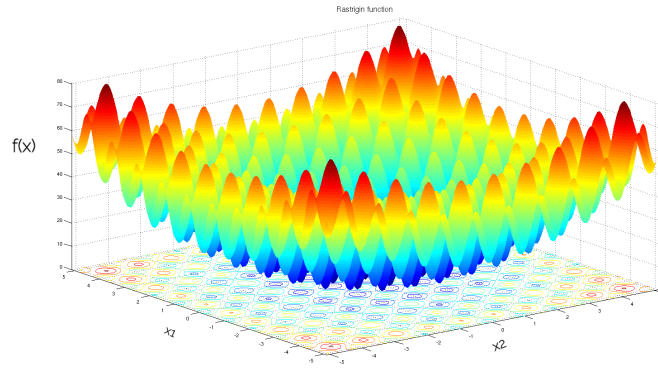


Figure 3: Image Rastrigin's Function. ³

0.4 Results

Hill Climbing first 5D

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

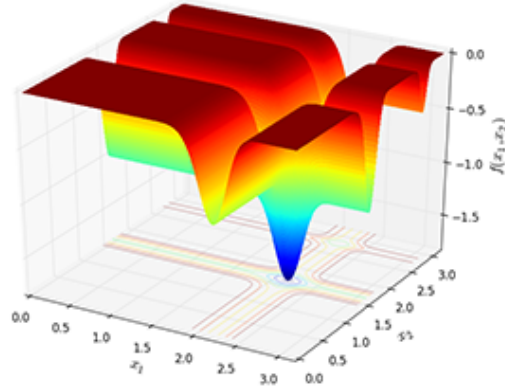


Figure 4: Michalewicz's Function. ⁴

Functions	Solutions			Time		
	Average	Minimum	Maximum	Minimum	Average	Maximum
DeJong 1	0	0	0			
Schwefel	-1939	-1992	-1853			
Rastrigin	2.59497	1.88367	3.69702			
Michalewicz	-4.44109	-4.62793	-3.97123			

Hill Climbing best 5D

Functions	Solutions			Time		
	Average	Minimum	Maximum	Minimum	Average	Maximum
DeJong 1	0	0	0			
Schwefel	-1967	-2094	-1893			
Rastrigin	2.23592	1.579144	2.98702			
Michalewicz	-4.37306	-4.65685	-4.02012			

Simulated Annealing 5D

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

Hill Climbing first 10D

Functions	Solutions			Time		
	Average	Minimum	Maximum	Minimum	Average	Maximum
DeJong 1	0	0	0			
Schwefel	-3652	-3872	-3492			
Rastrigin	7.18790	3.93217	15.19234			
Michalewicz	-8.20880	-9.01029	-6.89132			

Hill Climbing best 10D

Functions	Solutions			Time		
	Average	Minimum	Maximum	Minimum	Average	Maximum
DeJong 1	0	0	0			
Schwefel	-3623	-3772	-3506			
Rastrigin	8.18790	3.67834	14.0925			
Michalewicz	-9.09037	-9.24031	-8.25910			

Simulated Annealing 10D

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

Hill Climbing first 30D

Functions	Solutions			Time		
	Average	Minimum	Maximum	Minimum	Average	Maximum
DeJong 1	0	0	0			
Schwefel	-9453	-9923	-9369			
Rastrigin	23.72184	17.12235	38.63187			
Michalewicz	-20.79776	-22.61219	-17.31592			

Hill Climbing best 30D

Functions	Solutions			Time		
	Average	Minimum	Maximum	Minimum	Average	Maximum
DeJong 1	0	0	0			
Schwefel	-10153	-10453	-10063			
Rastrigin	22.43780	16.97201	37.29053			
Michalewicz	-24.09166	-25.14376	-22.21472			

Simulated Annealing 30D

Functions	Solutions			Time		
	Average	Minimum	Maximum	Minimum	Average	Maximum
DeJong 1	0	0	0			
Schwefel	-9746	-11279	-9365			
Rastrigin	25.18790	17.67834	38.44140			
Michalewicz	-23.66259	-24.29173	-21.53991			

0.5 Conclusions

In my results, Hill Climbing best improvement usually gives slightly better results than first improvement while neither algorithm is clearly better in terms of time.

As expected, with larger input, the time increases significantly, and the difference between the minimum and the maximum is higher, which could symbolize a lower precision.

The results could be improved by using a different random function and the time could be improved by implementing the program in C++ rather than Python.

In my implementation the Simulated Annealing algorithm, gives worse results than Hill Climbing with higher difference between the minimum and maximum, but the time is more than 10 times better, and the increase in input size doesn't increase the time at the same rate as the Hill Climbing.

The algorithm could be improved 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] Geatbx
Function formulas. <http://www.geatbx.com/docu/fcnindex-01.html>
- [5] Course Page. <https://profs.info.uaic.ro/~eugennc/teaching/ga/1>
- [6] Pycharm. <https://www.jetbrains.com/pycharm/>