Differences between Hill Climbing Algorithm (first improvement, best improvement) and Simulated Annealing Algorithm in finding Global Minimum of numeric functions

George Buțco

October 27, 2021

Abstract

In this paper, I compare the time and the minimum value found by a generic Hill Climbing Algorithm (using both first improvement and best improvement) with a simple Simulated Annealing Algorithm.

The comparison will be done using De Jong 1 function[1], Schwefel's function[2], Rastrigin's function[3], Michalewicz's function[4] in 5, 10, and 30 dimensions.

I try to prove with examples the following hypotheses:

1. Simulated Annealing is faster than Hill Climbing, at least for higher dimensions, but it has a higher error.
2. Hill Climbing "first improvement" is faster compared to "best improvement", but the minimum value found by "best improvement" is closer to the global minimum.
3. The only type of function that benefits from using Simulated Annealing over Hill Climbing are wave functions.

1 Introduction

Hill climbing method is an optimization technique that is able to build a search trajectory in the search space until reaching the local optima. It only accepts the uphill movement which leads it to easily get stuck in local optima.[5]

Simulated annealing is a well-studied local search meta-heuristic used to address discrete and, to a lesser extent, continuous optimization problems. The key feature of simulated annealing is that it provides a mechanism to escape local optima by allowing hill-climbing moves (i.e., moves which worsen the objective function value) in hopes of finding a global optimum.[6]

As for the **quantification** of how good an algorithm is, I will choose the best optima found and the time it was found. The sample size for each test is 30.

2 Implementation

The function *useData(time, value)* is the way the main program receives information about the state of the algorithm.

The function *eval(vector)* is a multi-variable numeric function with a single real number as output.

The function *getCurrentTime()* returns the time passed since the start of the algorithm in seconds as a real number.

The function *neighborhood(vector)* returns all the successors of the vector.

2.1 Hill Climbing

The function *improve(vectors)* returns the first successor better than the current candidate (first improvement) or the best successor among all the vectors (best improvement).

Algorithm 1 Hill Climbing

procedure HC(`useData`, `eval`)
    $t \leftarrow 0$
    $best \leftarrow$ `eval`(random candidate)
    repeat
        $lower \leftarrow false$
        $v_c \leftarrow$ random candidate
        repeat
            $v_n \leftarrow$ `improve(neighborhood(`v_c`))`
            if `eval(`v_n`)` is better than `eval(`v_c`)` **then**
                $v_c \leftarrow v_n$
            else
                $lower \leftarrow true$
            end if
        until $lower$
        if `eval(`v_c`)` is better than $best$ **then**
            $best \leftarrow$ `eval(`v_c`)`
            `useData( getCurrentTime(),` $best$ `)`
        end if
    until $t < MAX$
end procedure

2.2 Simulated Annealing

Algorithm 2 Simulated Annealing

```
procedure SA(useData, eval)
    initialize the temperature T
    v_c ← random candidate
    repeat
        for each v_n ∈ shuffle( neighborhood(v_c) ) do
            if eval(v_n) is better than eval(v_c) then
                v_c ← v_n
                useData( getCurrentTime(), eval(v_c) )
            else
                if random[0,1) < e^(-|eval(v_n)-eval(v_c)|/T) then
                    v_c ← v_n
                    useData( getCurrentTime(), eval(v_c) )
                end if
            end if
        end for
        T ← T / (1+T·α)
    until T is small enough
end procedure
```

2.3 Vectors

A vector with d dimensions is an array of $n \cdot d$ bytes. Every dimension has n bytes that represent an unsigned integer. v_{max} is an unsigned integer where all the bits are 1.

The conversion from an unsigned integer x to a real number r, using b_l as the lower bound and b_u as the upper bound of the function:

$$r = \frac{x}{v_{max}}(b_u - b_l) + b_l$$

The precision of a vector(10^{-p}):

$$p = \log_{10}\left(\frac{2^{n \cdot 8}}{b_u - b_l}\right)$$

3 Comparisons

3.1 De Jong 1 Function

Function Definition:

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

$$min(f(x)) = 0$$

`Precision used for testing` $= 10^{-8.62}$

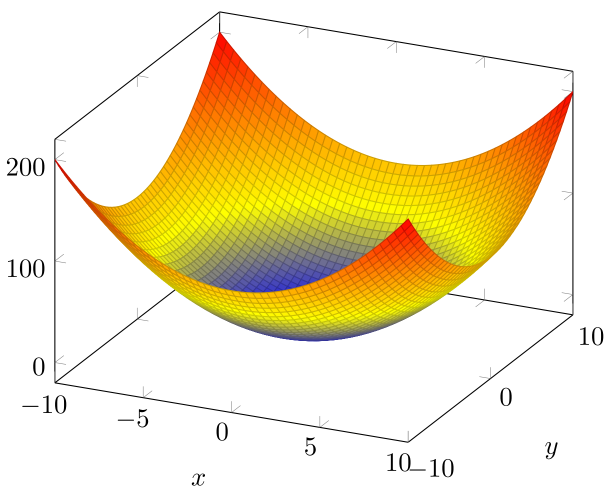

Figure 1: Schwefel's Function Graphic

Comparison:

Dimension	5					
Algorithm	HC FI		HC BI		SA	
Time (s) \Minima	Time	Minima	Time	Minima	Time	Minima
Best	0.09116	0.00000	0.00359	0.00000	0.16880	0.00068
Worst	1.78641	0.00000	0.00555	0.00000	1.05987	0.00431
Average	1.02012	0.00000	0.00427	0.00000	0.76646	0.00213

Dimension	10					
Algorithm	HC FI		HC BI		SA	
Time (s) \Minima	Time	Minima	Time	Minima	Time	Minima
Best	0.16019	0.00000	0.01376	0.00000	0.88551	0.01190
Worst	6.11886	0.00000	0.02351	0.00000	2.07062	0.03498
Average	3.18844	0.00000	0.01811	0.00000	1.63137	0.02196

Dimension	30					
Algorithm	HC FI		HC BI		SA	
Time (s) \Minima	Time	Minima	Time	Minima	Time	Minima
Best	0.50992	0.00000	0.18687	0.00000	4.48161	0.15152
Worst	61.24491	0.00000	0.29486	0.00000	8.16109	0.32255
Average	26.92410	0.00000	0.21709	0.00000	5.93557	0.22453

3.2 Schwefel's Function

Function Definition:

$$f(x) = \sum_{i=1}^{n}(-x_i sin(\sqrt{|x_i|})) \qquad -500 \leq x_i \leq 500$$

$$min(f(x)) = -n \cdot 418.9829$$

`Precision used for testing` $= 10^{-6.63}$

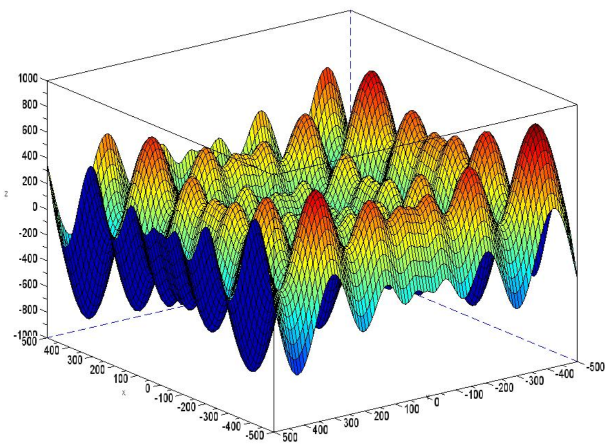

Figure 2: Schwefel's Function Graphic[7]

Dimension	5					
Algorithm	HC FI		HC BI		SA	
Time (s) \Minima	Time	Minima	Time	Minima	Time	Minima
Best	0.070228	-2094.809489	0.089141	-2094.914410	0.147038	-1875.884561
Worst	3.977570	-2060.367917	4.243352	-2094.602810	1.080576	-1134.486647
Average	1.409107	-2073.465037	1.982799	-2094.820178	0.835157	-1506.567675

Dimension	10					
Algorithm	HC FI		HC BI		SA	
Time (s) \Minima	Time	Minima	Time	Minima	Time	Minima
Best	0.800135	-4070.692393	1.450350	-4189.103207	1.341476	-3597.873265
Worst	13.416435	-3750.173074	24.727243	-3998.896753	2.581541	-2492.105525
Average	6.729373	-3928.126840	10.565069	-4068.862745	1.992218	-3067.723202

Dimension	30					
Algorithm	HC FI		HC BI		SA	
Time (s) \Minima	Time	Minima	Time	Minima	Time	Minima
Best	1.418859	-11039.503027	0.395691	-11592.659353	4.974412	-10337.482177
Worst	138.074417	-10610.823983	378.023560	-11126.868731	6.123193	-8261.315698
Average	66.605833	-10847.627730	198.344112	-11378.970798	5.743173	-9422.084350

3.3 Michalewicz's Function

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

$min(f(x)) = -4.687 \qquad n = 5$

$min(f(x)) = -9.66 \qquad n = 10$

`Precision used for testing` $= 10^{-9.13}$

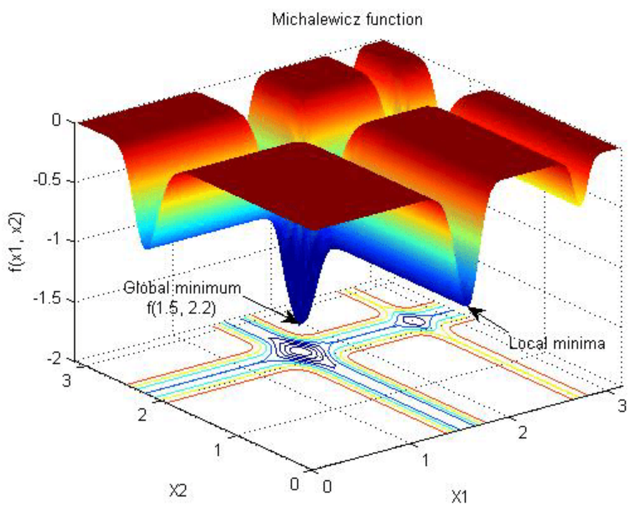

Figure 3: Michalewicz's Function Graphic[8]

Dimension	5					
Algorithm	HC FI		HC BI		SA	
Time (s) \Minima	Time	Minima	Time	Minima	Time	Minima
Best	0.032946	-4.687648	0.012815	-4.687658	0.063353	-4.687046
Worst	1.858042	-4.652766	7.005613	-4.682666	0.987068	-4.374004
Average	1.069647	-4.681145	3.265170	-4.686512	0.606711	-4.583025

Dimension	10					
Algorithm	HC FI		HC BI		SA	
Time (s) \Minima	Time	Minima	Time	Minima	Time	Minima
Best	0.148167	-9.468502	0.836543	-9.520684	0.301580	-9.364155
Worst	8.555991	-8.873322	42.695731	-9.203955	2.129440	-8.547526
Average	4.466485	-9.214113	22.206963	-9.385788	1.344480	-9.089312

Dimension	30					
Algorithm	HC FI		HC BI		SA	
Time (s) \Minima	Time	Minima	Time	Minima	Time	Minima
Best	1.069913	-26.657838	28.435671	-27.928919	3.719086	-27.709873
Worst	83.424811	-25.246337	922.223990	-26.502955	6.441927	-25.271548
Average	48.286689	-25.831170	435.228857	-27.076157	5.655194	-26.907374

3.4 Rastrigin's Function

$$f(x) = 10 \cdot n + \sum_{i=1}^{n}(x_i^2 - 10 \cdot cos(2 \cdot \pi \cdot x_i)) \qquad -5.12 \leq x_i \leq 5.12$$

$min(f(x)) = 0$

`Precision used for testing` $= 10^{-8.62}$

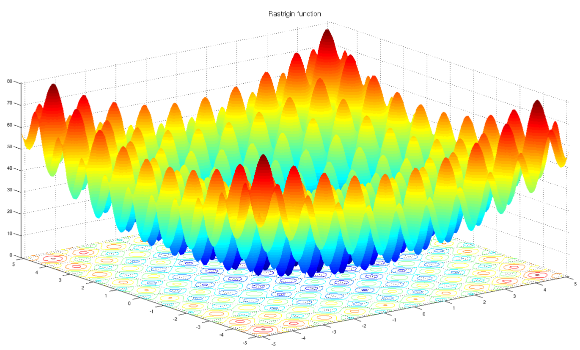

Figure 4: Rastrigin's Function Graphic[9]

Dimension	5					
Algorithm	HC FI		HC BI		SA	
Time (s) \Minima	Time	Minima	Time	Minima	Time	Minima
Best	0.018603	0.000000	0.038870	0.000000	0.162043	1.239108
Worst	2.731234	1.994971	3.952659	1.000000	1.043209	24.256068
Average	1.304426	1.045690	1.692400	0.530813	0.666320	11.050256

Dimension	10					
Algorithm	HC FI		HC BI		SA	
Time (s) \Minima	Time	Minima	Time	Minima	Time	Minima
Best	0.417285	2.989766	1.338017	0.994959	1.076592	13.679208
Worst	10.525889	8.200483	18.573040	5.461455	2.095559	40.873165
Average	5.183360	5.801246	11.051008	3.856569	1.805863	26.502426

Dimension	30					
Algorithm	HC FI		HC BI		SA	
Time (s) \Minima	Time	Minima	Time	Minima	Time	Minima
Best	2.773310	34.775586	3.339413	22.394448	4.534281	52.169252
Worst	92.607008	48.102214	301.882618	34.277449	6.288821	115.992046
Average	42.184061	39.989620	148.235979	28.633780	5.699633	79.151416

4 Conclusion

1. *"Simulated Annealing is faster than Hill Climbing, at least for higher dimensions, but it has a higher error."*

   The average time of Simulated Annealing is better than Hill Climbing for both Wave Functions and paraboloid types of functions. The difference in average time between algorithms is higher as the number of dimensions increases.

   Conclusion: The Hypothesis is true

2. *"Hill Climbing "first improvement" is faster compared to "best improvement", but the minimum value found by "best improvement" is closer to the global minimum."*

   "first improvement" is faster than "best improvement" just when it comes to wave functions. De Jong's Function works better with "best improvement". The average minima found by "best improvement" is better for all functions.

   Conclusion: The Hypothesis is false.

   Correction: "best improvement" is faster for paraboloid types of functions.

3. *"The only type of function that benefits from using Simulated Annealing over Hill Climbing are wave functions."*

   For paraboloid types of functions, Hill Climbing works better than Simulated Annealing with a small difference in speed. The difference in speed is noticeable for wave functions and as the number of dimensions increases, but the minimum is not as good as using Hill Climbing.

   Conclusion: The Hypothesis is false for minima, but true for time.

References

[1] K. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," 01 1975.

[2] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimisation problems," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 2, pp. 150–194, 2013.

[3] L. A. Rastrigin, *Systems of extremal control.* Mir, Moscow, 1974.

[4] T. Bäck, D. B. Fogel, and Z. Michalewicz, "Handbook of evolutionary computation," *Release*, vol. 97, no. 1, p. B1, 1997.

[5] M. A. Al-Betar, "β-hill climbing: an exploratory local search," *Neural Computing and Applications*, vol. 28, no. 1, pp. 153–168, 2017.

[6] M. Gendreau, J.-Y. Potvin, *et al.*, *Handbook of metaheuristics*, vol. 2. Springer, 2010.

[7] J. D. McCaffrey, "Plotting schwefel's function with scilab."

[8] D. Moore, "A 2d test function known as the michalewicz function (michalewicz 1998)."

[9] Wikipedia, "Rastrigin function."