Comparison of Simulated Annealing and Hill Climbing algorithms performance in finding the global minimum

Sbârcea Ștefan-Vladimir

October 18, 2021

1 Abstract

In this paper, we discuss the differences in performance between Iterated Hill Climbing (best-improvement and first-improvement) and Simulated Annealing algorithms, we report about some experiments and how each algorithm performed in searching for the global minimum of various n-dimensional functions.

2 Introduction

Structural analysis and design usually involve both highly complex procedures and a great number of variables. As a consequence, the solution has to be found iteratively while initial values are set to the variables based mainly on designer's sensitivity and experience. Also, the number of analysis steps is remarkably increased if optimum values are to be found among all possible alternatives. To mathematically describe the physical response of a structure, extreme function values can be found by using optimization techniques.

The great development of structural optimization took place in the early 60's, when programming techniques were used in the minimization of structures weight. From then on, a great diversity of general techniques has been developed and adapted to structural optimization. However, one of the reasons normally attributed to the little application of the optimization techniques to real structural engineering problems consists of the complexity of the mathematic model generated, normally described by non-linear behavior functions and producing a non-convex space of solutions (several points of optimum), problems for which the resolution by traditional mathematical programming methods have proved to be little efficient. For the resolution of these kind of problems the heuristic methods have played an important role, since they involve only values of functions in the process, regardless if there is unimodality or even continuity in their derivatives. Despite the great emphasis in the development of global optimization methods, researchers are even far from the attainment of a method that can be applied with the same efficiency to any class of problems.

Simulated annealing (SA) is a probabilistic technique for approximating the global optimum of a given function.

Hill climbing algorithm 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.

The next sections of this paper present, a description of the optimization methods, the functions used for experiments, some experiments, a comparison between Simulated Annealing and Hill Climbing First-improvement and Best-improvement performance and the conclusions.

3 Methods

3.1 Algorithms description

For every algorithm we used vectors of bits to represent the values from the search space. The vector size is determined with the following formula: $\lceil \log_2 (10^p * (b - a)) \rceil * n$ where p = precision, b = upper boundary, a = lower boundary, n = function dimensions. Every algorithm generates the neighbors diferently, in simulated annealing algorithm negates s bits at random to generate the neighbor,

$$s = 1 + 2 \cdot \frac{vector.size()}{dimensions \cdot precision}$$

, hill climbing first-improvement algorithm changes one random bit and hill climbing best-improvement changes one bit and it searches for the best neighbor jumping over some of them randomly (the iterator is increased with a random value $s \in [0, 2 \cdot precision \cdot dimensions)$.

3.1.1 Simulated Annealing

Basics:

In the early 1980s three IBM researchers, Kirkpatrick, Gelatt and Vecchi,introduced the concepts of annealing in combinatorial optimization. These concepts are based on a strong analogy with the physical annealing of materials. This process involves bringing a solid to a low energy state after raising its temperature. It can be summarized by the following two steps (see Figure 1):

- Bring the solid to a very high temperature until "melting" of the structure;
- Cooling the solid according to a very particular temperature decreasing scheme in order to reach a solid state of minimum energy.

In the liquid phase, the particles are distributed randomly. It is shown that the minimum-energy state is reached provided that the initial temperature is sufficiently high and the cooling time is sufficiently long. If this is not the case, the solid will be found in a metastable state with non-minimal energy; this is referred to as hardening, which consists in the sudden cooling of a solid.

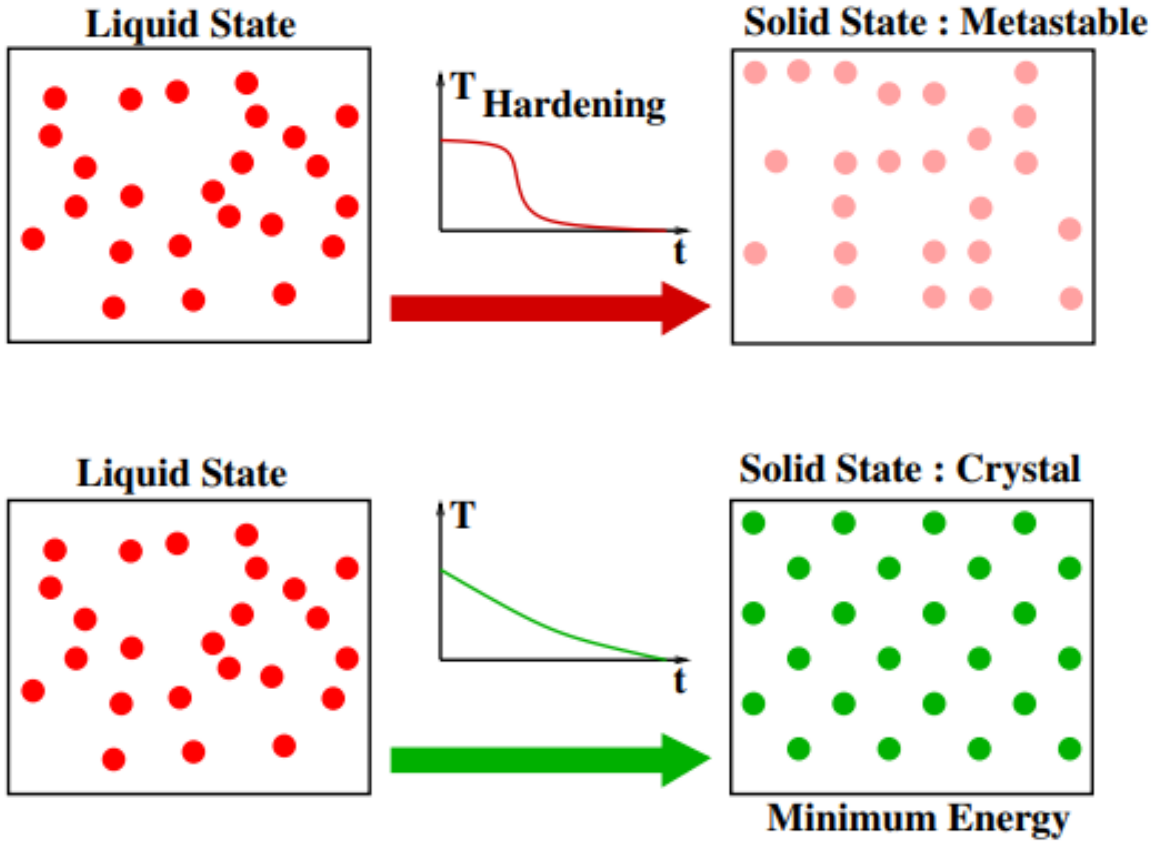

Figure 1: When temperature is high, the material is in a liquid state (left). For a hardening process, the material reaches a solid state with non-minimal energy (metastable state; top right). In this case, the structure of the atoms has no symetry. During a slow annealing process, the material reaches also a solid state but for which atoms are organized with symmetry (crystal; bottom right).[1]

Input- Output:

Input: A function $f : [a, b]^n \to R$ and the values of n, a and b $n \geq 1, a \leq b$
Output: The minimum value found for the function f after a number of iterations

Implementation:

Step 1:Initialize the starting temperature T, $minT$ the lowest value of T and set the number of iterations to 0, $t = 0$, generate a random vector v_c;

Step 2: Select a neighbor, v_n, of v_c. If $f(v_n) < f(v_c)$ interchange the vectors, otherwise generate a random value $s \in [0.1)$ and compare it with

$$exp\left(-\frac{|(f(v_n) - f(v_c)|}{T}\right)$$

if s is lower, then interchange the values of the vectors;

Step 3: Increment $t = t + 1$ and check if t is bigger than MAX ($MAX = 1000$ in our case), if not go to step 1;

Step 4: Lower the temperature $T = \frac{T}{10}$, reset the number of iterations, $t = 0$, and test if temperature is still bigger than minimum temperature $minT$ (in our case $minT = 10^{-8}$) and finish if the answer is yes, otherwise go back to Step 2.

3.1.2 Iterated Hill Climbing

Description:

In numerical analysis, hill climbing is a mathematical optimization technique which belongs to the family of local search. It is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by making an incremental change to the solution. If the change produces a better solution, another incremental change is made to the new solution, and so on until no further improvements can be found.

Input- Output:

Input: A function $f : [a, b]^n \to R$ and the values of n, a and b $n \geq 1, a \leq b$
Output: The minimum value found for the function f after a number of iterations

The difference between First-improvement and Best-improvement

First-improvement differs from best-improvement local search, in the way of selecting the next neighbor in the search process, which is related with the so-called pivot rule. In best-improvement, the entire neighborhood is explored and the best solution is returned, whereas in first-improvement, a solution is selected uniformly at random from the neighborhood.

First-improvement (Nearest ascent) Implementation

Step 1: Set the number of iterations to 0, $t = 0$ and generate a random vector of random bits *best* (in which we will save the the vector with the minimum function value);

Step 2: Generate a random vector v_c and search for the first neighbor v_n of v_c, for which the relation $f(v_c) > f(v_n)$ it's true, if it is interchange v_n with v_c, repeat this step until v_c is the best in it's neighborhood. (search local minimum)

Step 3: Increment $t = t+1$ and check if $f(best) > f(v_c)$ and change *best* with v_c if the answer is yes.

Step 4: Check if $t! = MAX$ where MAX is the maximum number of iterations ($MAX =$ 10000 in our case) and finish if the answer is yes, otherwise go back to Step 2.

Best-improvement (Steepest ascent) Implementation

Step 1: Set the number of iterations to 0, $t = 0$ and generate a random vector of random bits *best* (in which we will save the the vector with the minimum function value);

Step 2: Generate a random vector v_c and search for the best neighbor v_n of v_c, for which the relation $f(v_c) > f(v_n)$ it's true, if it is interchange v_n with v_c, repeat this step until v_c is the best in it's neighborhood. (search local minimum)

Step 3: Increment $t = t+1$ and check if $f(best) > f(v_c)$ and change *best* with v_c if the answer is yes.

Step 4: Check if $t! = MAX$ where MAX is the maximum number of iterations ($MAX = 1000$ in our case) and finish if the answer is yes, otherwise go back to Step 2.

3.2 Test Functions

For the study of the algorithm we used Rastrigin's Function, Michalewicz's Function, De Jong's Function 1 and Schwefel's Function 7.

3.2.1 Rastrigin's Function

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

Figure 2: Rastrigin's Function

Global minimum: $f(x) = 0$; $x_i = 0$, $i = 1 : d$.

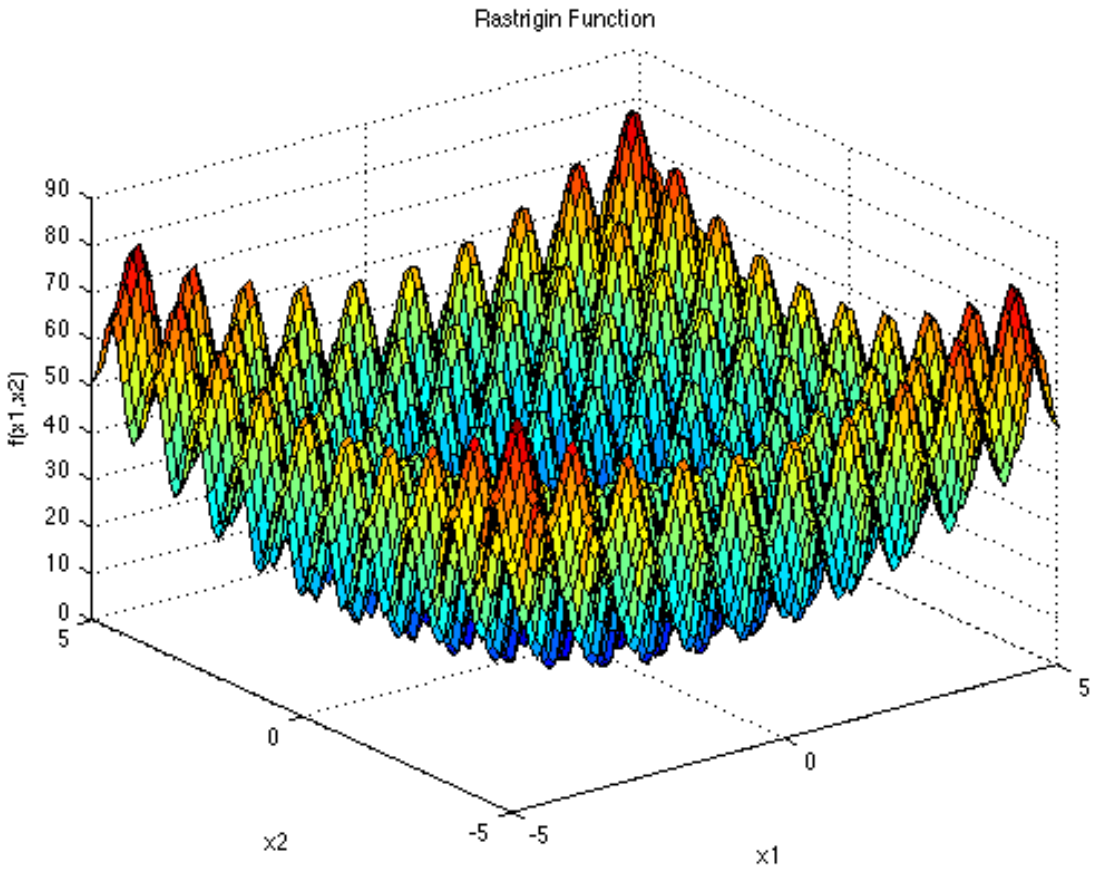

Figure 3: Rastrigin's Function. 2D representation.[2]

3.2.2 Michalewicz's Function

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

Figure 4: Michalewicz's Function

Global minimum: $f(x) = -1.8013$; $(d = 2)$ $x = (2.2, 1.57)$
$f(x) = -4.687$; $(d = 5)$ $x_i =$???, $i = 1 : d$.
$f(x) = -9.66$; $(d = 10)$ $x_i =$???, $i = 1 : d$.

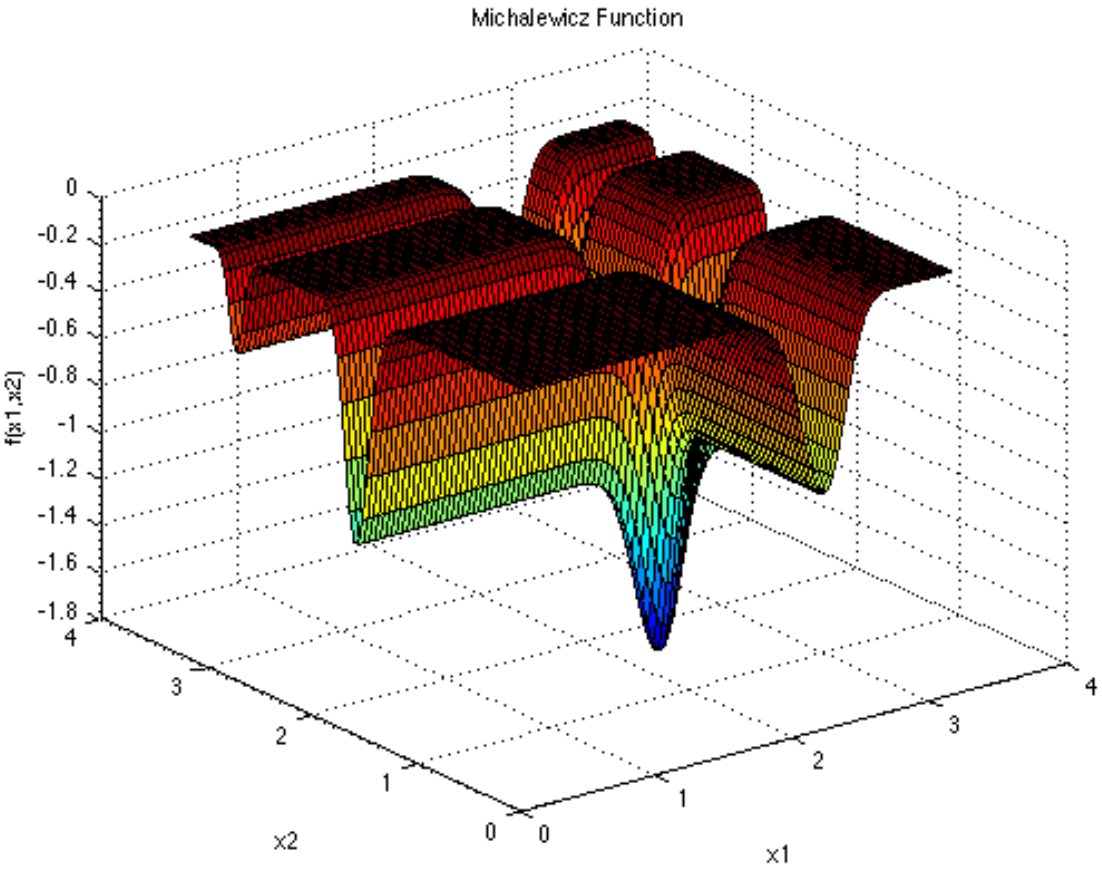

Figure 5: Michalewicz's Function. 2D representation.[3]

3.2.3 De Jong's Function 1

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

Figure 6: De Jong's Function 1

Global minimum: $f(x) = 0$; $x_i = 0$, $i = 1 : d$.

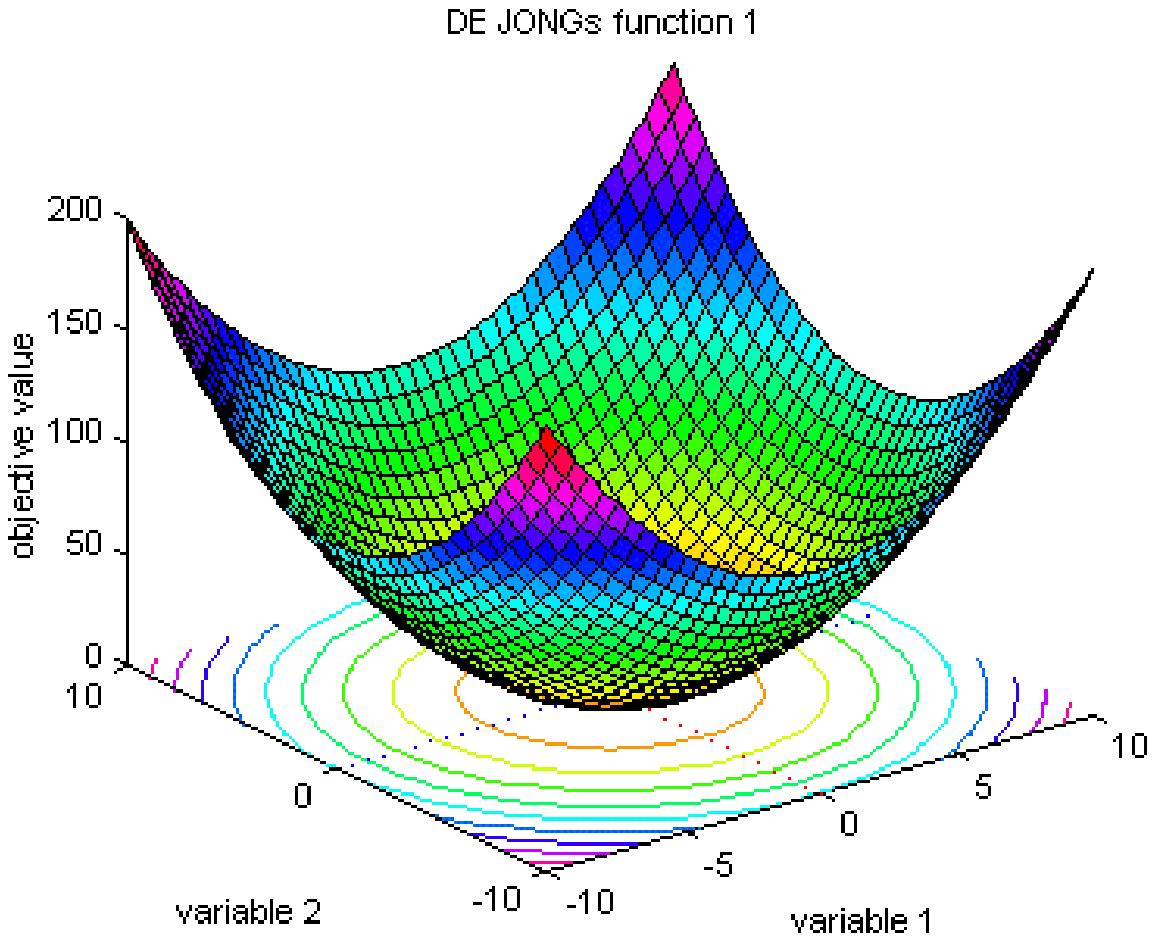

Figure 7: De Jong's Function 1. 2D representation.[4]

3.2.4 Schwefel's Function 7

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

Figure 8: Schwefel's Function 7

Global minimum: $f(x) = -d \cdot 418.9829$; $x_i = 420.9687$, $i = 1 : d$.

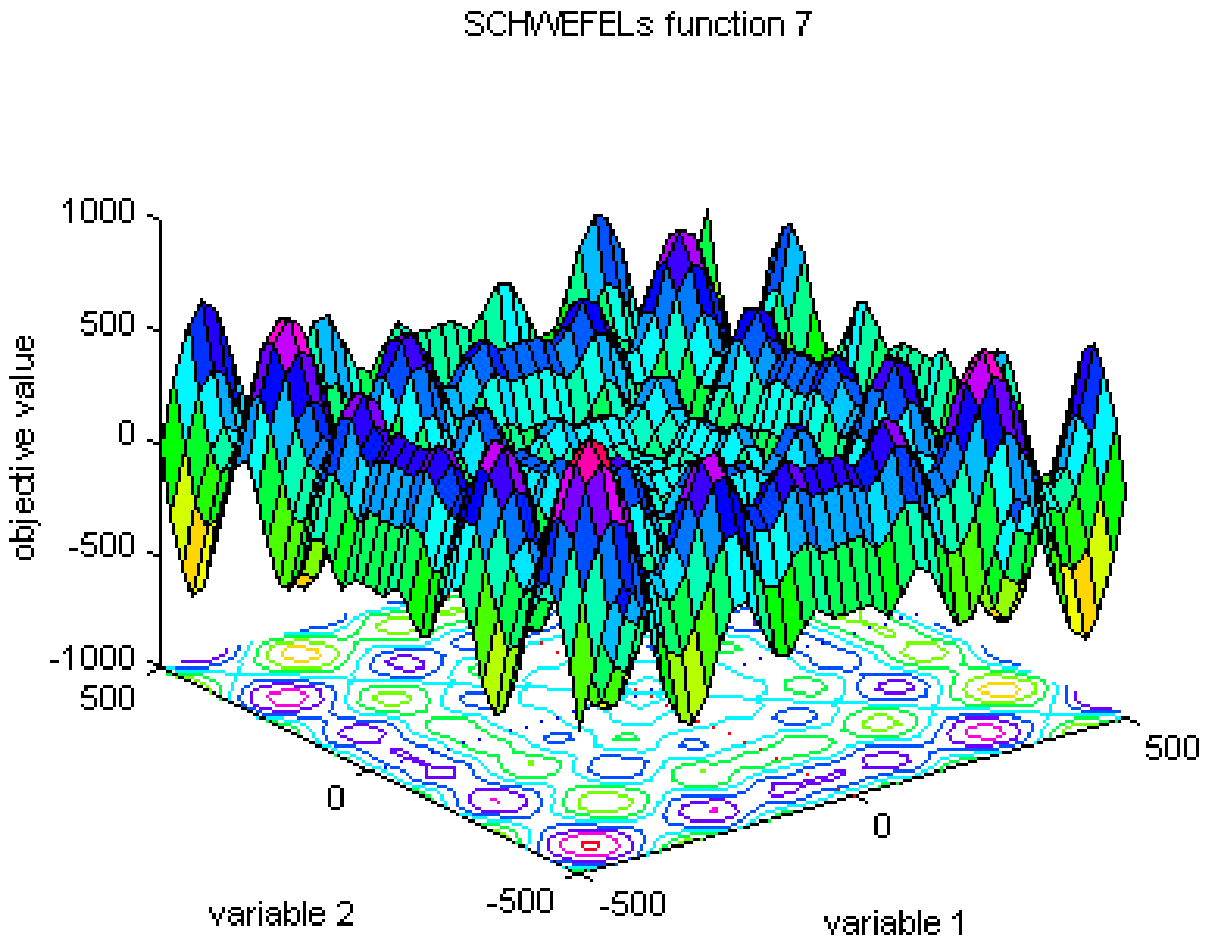

Figure 9: Schwefel's Function 7. 2D representation.[5]

4 Experiment

The experiments were made for 2, 10 and 30 dimensions 30 tests each with precision 5 for every algorithm.
We tested simulated annealing algorithm with $T = t = 1000$ (t iterations for every $T \in \{10^3, 10^2 ... 10^{-7}\}$),

iterated hill climbing first-improvement algorithm with 10000 iterations and iterated hill climbing best-improvement algorithm with 1000 iterations.

4.1 Results Simulated Annealing

Functions	Time			Solution		
	Minimum	Average	Maximum	Best	Worst	Average
Rastrigin's Function	0.23668s	0.24067s	0.24762s	0	0.00037	0.00008
Michalewicz's Function	0.21932s	0.2258s	0.235s	-1.8013	-1.8007	-1.80128
De Jong's Function 1	0.23674s	0.24141s	0.26569s	0	0	0.00001
Schwefel's Function 7	0.30426s	0.31546s	0.34385s	-837.96577	-837.75309	-837.89489

Table 1: Results Simulated Annealing 2-dimensional version

Functions	Time			Solution		
	Minimum	Average	Maximum	Best	Worst	Average
Rastrigin's Function	0.68537s	0.7109s	0.74982s	1.12037	18.91075	8.76412
Michalewicz's Function	0.74677s	0.77124s	0.82955s	-9.38364	-8.20529	-8.9943
De Jong's Function 1	0.66369s	0.6912s	0.76683s	0.00035	0.00189	0.0008
Schwefel's Function 7	0.92899s	0.9423s	0.98034s	-4189.21289	-3952.42035	-4107.11749

Table 2: Results Simulated Annealing 10-dimensional version

Functions	Time			Solution		
	Minimum	Average	Maximum	Best	Worst	Average
Rastrigin's Function	1.98672s	2.03072s	2.11853s	35.603	77.18658	56.22618
Michalewicz's Function	1.89234s	1.96415s	2.06707s	-26.86676	-22.62479	-25.24687
De Jong's Function 1	1.90192s	1.97307s	2.12705s	0.01111	0.0407	0.02139
Schwefel's Function 7	2.65936s	2.7201s	2.87779s	-11546.48915	-10465.19389	-11087.01334

Table 3: Results Simulated Annealing 30-dimensional version

4.2 Results Hill Climbing First-improvement

Functions	Time			Solution		
	Minimum	Average	Maximum	Best	Worst	Average
Rastrigin's Function	2.52301s	2.58313s	2.70454s	0	0.00846	0.00163
Michalewicz's Function	1.53902s	1.5954s	1.70657s	-1.8013	-1.79728	-1.80094
De Jong's Function 1	2.64479s	2.76323s	2.90277s	0	0	0
Schwefel's Function 7	5.18225s	5.41207s	5.70508s	-837.96575	-837.77202	-837.92235

Table 4: Results Hill Climbing First-improvement 2-dimensional version

Functions	Time			Solution		
	Minimum	Average	Maximum	Best	Worst	Average
Rastrigin's Function	8.70524s	9.07639s	9.33594s	18.67729	48.2866	37.60002
Michalewicz's Function	5.28577s	5.45165s	5.58788s	-6.47323	-4.83686	-5.54519
De Jong's Function 1	8.8881s	9.23796s	9.82478s	0.97166	8.64633	4.66985
Schwefel's Function 7	20.26211s	22.22706s	21.30649s	-3459.50779	-2690.37464	-3043.5067

Table 5: Results Hill Climbing First-improvement 10-dimensional version

Functions	Time			Solution		
	Minimum	Average	Maximum	Best	Worst	Average
Rastrigin's Function	30.25309s	31.45512s	32.4299s	228.54124	307.76854	275.74948
Michalewicz's Function	13.829s	14.3538s	15.16273s	-10.87755	-8.94674	-9.82217
De Jong's Function 1	30.89216s	32.21582s	33.68721s	64.84218	98.94504	81.36653
Schwefel's Function 7	73.42588s	76.94095s	81.06224s	-6241.82533	-5262.03371	-5672.81931

Table 6: Results Hill Climbing First-improvement 30-dimensional version

4.3 Results Hill Climbing Best-improvement

Functions	Time			Solution		
	Minimum	Average	Maximum	Best	Worst	Average
Rastrigin's Function	0.78203s	0.8355s	0.88158s	0	0.00152	0.00008
Michalewicz's Function	0.6902s	0.72968s	0.78337s	-1.8013	-1.8013	-1.8013
De Jong's Function 1	0.88403s	0.93618s	1.01416s	0	0	0
Schwefel's Function 7	2.21809s	2.32474s	2.47174s	-837.96577	-837.96412	-837.9655

Table 7: Results Hill Climbing Best-improvement 2-dimensional version

Functions	Time			Solution		
	Minimum	Average	Maximum	Best	Worst	Average
Rastrigin's Function	24.50374s	25.24229s	25.82462s	2.47194	8.20618	5.68509
Michalewicz's Function	26.80481s	27.52416s	28.10938s	-9.46303	-8.89546	-9.23763
De Jong's Function 1	28.23542s	29.00941s	29.64576s	0	0	0
Schwefel's Function 7	68.63846s	70.65765s	72.95799s	-4128.29369	-3880.1473	-3974.99471

Table 8: Results Hill Climbing Best-improvement 10-dimensional version

Functions	Time			Solution		
	Minimum	Average	Maximum	Best	Worst	Average
Rastrigin's Function	293.43564s	302.88384s	311.44777s	26.61194	42.80525	37.14005
Michalewicz's Function	242.04587s	250.89783s	257.82067s	-26.74449	-25.67914	-26.16094
De Jong's Function 1	343.45492s	355.83727s	366.2644s	0	0	0
Schwefel's Function 7	-	-	-	-	-	-

Table 9: Results Hill Climbing Best-improvement 30-dimensional version

4.4 Comparison

This chapter presents comparison of the performances of the Simulated Annealing algorithm, with the two types of Iterated Hill Climbing, first-improvement and best-improvement acting on above given four benchmark functions, namely, Rastrigin's function, Michalewicz's function, De Jong's function 1, and Schwefel's function 7.

Let's compare the 2-dimensional version results first, we can see there is not much of a difference between the solutions found, but there is a difference between the times, we can see that Simulated Annealing runs faster than the other two and the first-improvement hill climbing runs slower than the best-improvement because it iterates 10 times more than the best-improvement. (see Table 1, Table 4 and Table 7)

Moving forward, the solutions are not the same for the 10-dimensional version results. The iterated hill climbing first-improvement gave the worst results and the best-improvement gave slightly better results than simulated annealing, but if we check the times, simulated annealing gave the solution way faster than hill climbing.(see Table 2, Table 5 and Table 8)

Finnaly, after analyzing the 30-dimensional version results, we can conclude that simulated annealing perform faster and gives similar solutions with iterated hill climbing best-improvement and way better solutions then iterated hill climbing first-improvement. (see Table 3, Table 6 and Table 9)

5 Conclusions

Two algorithms: Iterated Hill Climbing and Simulated Annealing are described and compared in this paper. The simulations have been carried out using benchmarking functions, such as Rastrigin's function, Michalewicz's function, De Jong's function 1, and Schwefel's function 7. The computational results have demonstrated that Simulated Annealing algorithm performs better than Iterated Hill Climbing.

6 References

References

[1] Simulated Annealing Basics https://hal-enac.archives-ouvertes.fr/hal-01887543/document

[2] Rastrigin's Function rendered image. https://www.sfu.ca/~ssurjano/rastr.html

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

[4] De Jong's Function 1 rendered image. http://www.geatbx.com/ver_3_3/fcnfun1.html

[5] Schwefel's Function 7 rendered image. http://www.geatbx.com/ver_3_3/fcnfun7.html

[6] Simulated annealing https://en.wikipedia.org/wiki/Simulated_annealing

[7] Hill climbing https://en.wikipedia.org/wiki/Hill_climbing

[8] First-improvement vs. Best-improvement Local Optima Networks of NK Landscapes http://www.cs.stir.ac.uk/~goc/papers/PPSN10FirstImprLON.pdf

[9] Genetic Algorithms https://profs.info.uaic.ro/~eugennc/teaching/ga/

[10] Michalewicz function https://www.sfu.ca/~ssurjano/michal.html

[11] De Jong's function 1 http://www.geatbx.com/docu/fcnindex-01.html#P89_3085

[12] Rastrigin's function 6 http://www.geatbx.com/docu/fcnindex-01.html#P140_6155

[13] Schwefel's function 7 http://www.geatbx.com/docu/fcnindex-01.html#P150_6749

[14] Structural optimization https://www.researchgate.net/publication/315834899_Modified_harmony_search_and_its_application_to_cost_minimization_of_RC_columns

[15] Hill Climbing Algorithm in Artificial Intelligence https://www.javatpoint.com/hill-climbing-algorithm-in-ai