Hill Climbing vs. Simulated Annealing Function Minimization

Grumăzescu George, Romanescu Adia

November 1st, 2023

1 Abstract

In this study, we employ heuristic algorithms, specifically leveraging the methodologies of **Hill Climbing** and **Simulated Annealing**. These algorithms are strategically applied to the challenging task of **estimating the global minimum** values for complex functions, including well-known benchmarks such as **Rastrigin**, **Michalwicz**, **De Jong I** and **Schwefel**. Through the judicious use of heuristic techniques, we aim to explore the intricate landscapes of these functions and uncover the elusive global minima.

2 Introduction

This paper delves into the application of heuristic algorithms, specifically Hill Climbing and Simulated Annealing, to estimate the global minimum of well-known functions, namely Rastrigin, Michalewicz, De Jong I and Schwefel. These heuristic techniques are renowned for their ability to escape local optima and navigate complex landscapes, making them valuable tools for global optimization challenges.

The primary focus of this paper is to demonstrate the effectiveness of these heuristic algorithms in efficiently approximating the global minima of these functions. We begin by introducing the algorithms and then proceed to apply them to the benchmark functions. Our aim is to provide practical insights into how heuristic algorithms can be employed to tackle complex optimization problems, shedding light on their strengths and limitations in the pursuit of global optimization.

The paper is structured to present an overview of the algorithms, an examination of the benchmark functions, experimental results and discussions on the

implications of our findings. Ultimately, this work contributes to the understanding of heuristic optimization techniques and their role in addressing global optimization challenges.

3 Methods

In this study, we have implemented two well-known algorithms for finding the global minimum point of a function, namely Hill Climbing and Simulated Annealing. We opted to explore both methods to highlight the differences between them and to compare the results obtained from the experiments conducted. The two algorithms represent methods for local optima search, utilizing constant feedback to aid in solution generation.

For both Hill Climbing and Simulated Annealing, the same solution representation and initialization variants were used, as well as the methods for solution generation and neighbor selection.

The solutions are represented as arrays of bits, each with a length determined by the formula:

$$\text{length} = \text{dim} \cdot \lceil \log\left((\text{upper} - \text{lower}) \cdot 10^p\right) \rceil$$

Here:

- *dim* represents the dimension being tested.
- *upper* is the upper bound of the domain.
- *lower* is the lower bound of the domain.
- 10^p is the reciprocal of the precision at which the search space is discretized.

Once the solution representation method is set, the initialization procedure involves generating a **random bitstring** of the previously specified length, which will be the value of the initial solution.

In what concerns the generation of neighbors, they are generated by flipping exactly one bit in the current solution. Therefore, we can say that the neighborhood represents the set of all solutions at a 1-Hamming distance from the original solution (the Hamming distance between two strings of the same length is equal to the number of positions at which the corresponding symbols are different).

There is a multitude of ways in which neighbors can be selected. However, for the purpose of this study, we employed three of them:

1. **Steepest Descent (Best Improvement):** The algorithm evaluates all potential paths from the current solution and selects the one that yields the most significant improvement — the neighbor that leads to the optimal solution.

2. **First Descent (First Improvement):** The algorithm randomly selects a path and opts for it if it leads to any improvement, irrespective of whether it is the optimal one — the first neighbor that leads to a better solution.

3. **Worst Descent (Worst Improvement):** The algorithm evaluates all available paths from the current solution and selects the one that produces the least improvement — the neighbor that leads to an immediately better solution.

3.1 Hill Climbing

The **Hill Climbing** algorithm, as the name suggests, simulates the process of climbing a hill to find the maximum point. However, it is equally effective in solving the problem of finding the minimum point because it is an equivalent problem to the former, with the only difference being the direction taken when seeking a better solution — instead of climbing, it descends. Therefore, we could also call it **Hill Descent**. The focus of the study was to find the global minimum; hence, we will refer to the specified algorithm in this context.

The general flow of a Hill Climbing Algorithm, illustrated in Figure 1 just for a single iteration, is as follows:

1. An initial solution is generated, which, at this moment, is the current best solution.

2. A neighbor of the current best solution is selected based on the neighbor selection method.

3. If the solution generated by the neighbor is better than the current best solution (i.e., it produces a smaller value), the neighbor becomes the new optimal solution.

4. Repeat the two steps above until no solution generated by neighbors is better than the current optimal solution, reaching a local optimum.

5. Repeat all of the above steps a fixed number of times to increase the exploration of the search space and generate multiple local optima, thereby increasing the possibility of encountering or approaching the global optima.

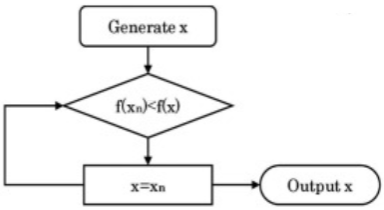


Figure 1: Illustration of the Hill Climbing Algorithm (one iteration)[15]

3.2 Simulated Annealing

Simulated Annealing (SA) represents a random-search technique which comes from the analogy of annealing in metallurgy, which involves a heating and a controlled cooling of a material in order to alter its physical properties. Simulated annealing approaches the global optimization problem.

The algorithm has a fairly high probability of finding a global optimum due to the fact that it explores parameters through large jumps and makes probabilistic decisions based on the Metropolis-Hastings algorithm.

The **Metropolis-Hastings** algorithm brings a critical element of randomness to the optimization process, much like the controlled heating and cooling in metallurgy that inspired simulated annealing. By introducing probabilistic decisions during parameter exploration, it enables the algorithm to escape local optima and traverse the solution space more effectively. This probabilistic nature allows simulated annealing to balance the trade-off between exploration and exploitation.

The general flow of a Simulated Annealing Algorithm, illustrated in Figure 2 just for a single iteration, is as follows:

1. An initial solution is generated, which is considered the current best solution at this moment, and set the starting temperature.
2. Select a neighbor of the current best solution using the Steepest-Ascent neighbor selection method.
3. If the neighbor solution is better (i.e., it yields a smaller value), it becomes the new optimal solution.
4. If the neighbor solution is not better, apply the Metropolis acceptance rule to determine whether to accept it.
5. If the Metropolis acceptance rule is satisfied, repeat the above three steps; otherwise, go to step 6.

6. Adjust the temperature and check the temperature halting condition. If it is still satisfied, return to step 2.

7. Repeat all of the above steps a fixed number of times to increase the exploration of the search space and generate multiple local optima, thereby increasing the possibility of encountering or approaching the global optima.

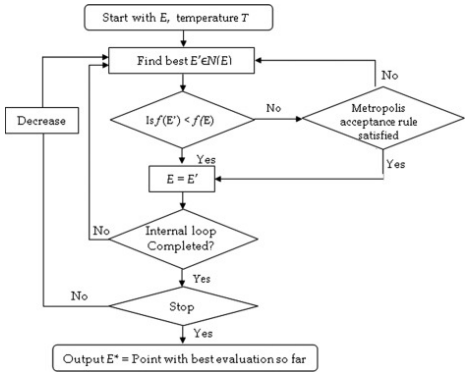


Figure 2: Illustration of the Simulated Annealing Algorithm (one iteration)[16]

4 Experimental Setup Description

The experiments involve running the Hill Climbing algorithms (*Best Improvement*, *First Improvement* and *Worst Improvement* variants) and Simulated Annealing (*Best Improvement* variant only) on four benchmark functions — **Rastrigin**, **Michalewicz**, **De Jong I** and **Schwefel** —, also known as artificial landscapes. These test functions are useful for evaluating the characteristics of optimization algorithms, including precision, convergence rate, robustness and general performance.

The chosen dimensions for conducting the experiments on the four previously mentioned functions are the **5-dimensional**, **10-dimensional** and **30-dimensional** ones. For each function and each dimension, the experiments were run for a total of **30 repetitions** to obtain statistically relevant results. Consequently, we chose to analyze the *average*, *minimum*, *maximum*, and *standard deviation* of the values produced from the repetitions, as well as the *average*, *minimum* and *maximum* time it took to reach these values. All of these results were represented with a precision of **5 decimal** places.

Based on our findings and with the aim of achieving optimal runtime durations, we set the number of iterations (i.e., the count of generated local optima)

to the specific algorithm in use, the nature of the evaluated function and its dimensionality.

Firstly, in the context of the Hill Climbing algorithm, we set the iteration counts as follows:

- For the Rastrigin function, we opted for **10,000**, **1,000**, and **1,000** iterations.
- For the Michalewicz function, we opted for **10,000**, **1,000**, and **100** iterations.
- For both De Jong I and Schwefel functions, the choices were **1,000**, **100** and **10**.

Secondly, in the context of the Simulated Annealing algorithm, we set the iteration counts as follows:

- For the Rastrigin function, we set **100**, **10**, and **10** iterations.
- For the De Jong function, all iterations were set to **1**.
- For Michalewicz and Schwefel, we set **1,000**, **100**, and **10**

The iterations are across the 5-dimensional, the 10-dimensional and the 30-dimensional versions, in this order.

As for the precision at which we discretize the domain, it was established at $\mathbf{10^{-3}}$ for all the aforementioned functions.

The tests were conducted using the **Python** programming language and executed on a robust octa-core computer system featuring an AMD RYZEN 7 5000 processor with 16GB of RAM.

5 Functions

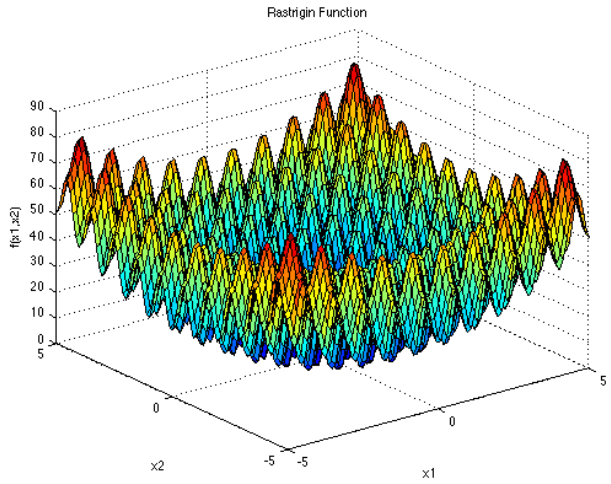


Figure 3: Rastrigin's Function[7]
$f_n(x) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i))$
$-5.12 \leq x_i \leq 5.12$

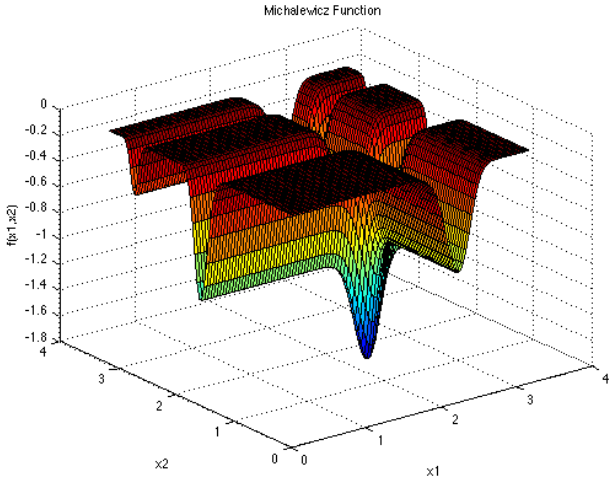


Figure 4: Michalewicz's Function[8]
$f_n(x) = -\sum_{i=1}^{n}\sin(x_i)\left(\sin\left(\frac{ix_i^2}{\pi}\right)\right)^{20}$
$0 \leq x_i \leq \pi$

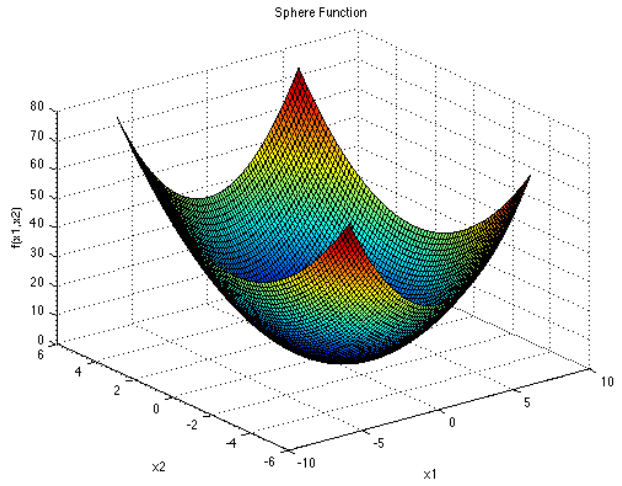


Figure 5: De Jong I's Function[9]
$f_n(x) = \sum_{i=1}^{n} x_i^2$
$-5.12 \leq x_i \leq 5.12$

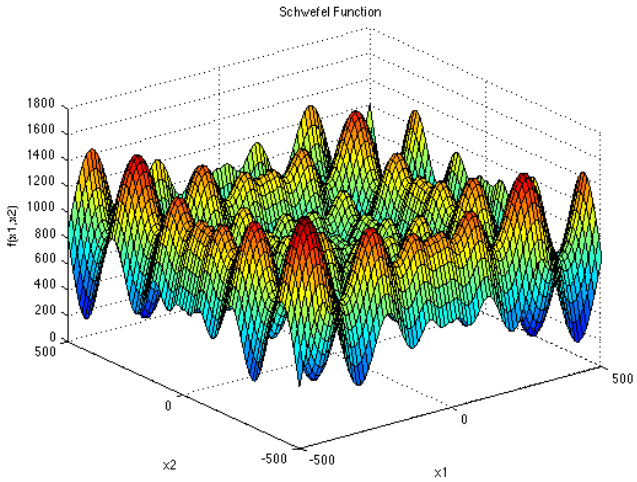


Figure 6: Schwefel's Function[10]
$f_n(x) = -\sum_{i=1}^{n} x_i \sin\left(\sqrt{|x_i|}\right)$
$-500 \leq x_i \leq 500$

6 Experimental Results

6.1 Results' Tables

6.1.1 Rastrigin's Function

Hill Climbing - Best Improvement							
Dim	**Min**	**Avg**	**Max**	**St Dev**	**Min Time**	**Avg Time**	**Max Time**
5	0.00010	0.00010	0.00010	0.00000	169.53s	170.33s	171.51s
10	0.99514	3.89552	6.22584	1.03292	127.08s	130.24s	160.72s
30	22.84275	28.43078	33.30702	2.33338	3383.95s	3396.62s	3426.53s

Figure 7: Table presenting the results of the Rastrigin function across 30 tests conducted using the Hill Climbing - Best Improvement method.

Hill Climbing - First Improvement							
Dim	**Min**	**Avg**	**Max**	**St Dev**	**Min Time**	**Avg Time**	**Max Time**
5	0.00010	0.00010	0.00010	0.00000	44.05s	44.44s	45.51s
10	0.00019	5.90149	8.16365	1.55982	19.11s	19.54s	20.43s
30	34.91447	41.72606	49.50220	2.97928	200.86s	210.70s	220.25s

Figure 8: Table presenting the results of the Rastrigin function across 30 tests conducted using the Hill Climbing - First Improvement method.

Hill Climbing - Worst Improvement							
Dim	**Min**	**Avg**	**Max**	**St Dev**	**Min Time**	**Avg Time**	**Max Time**
5	0.00010	1.08501	1.99580	0.58697	811.01s	843.02s	843.02s
10	5.46352	9.01768	12.16611	1.59975	613.22s	621.94s	633.91s
30	35.69975	48.96891	57.63856	4.85133	16701.76s	17141.38s	17631.91s

Figure 9: Table presenting the results of the Rastrigin function across 30 tests conducted using the Hill Climbing - Worst Improvement method.

Simulated Annealing							
Dim	**Min**	**Avg**	**Max**	**St Dev**	**Min Time**	**Avg Time**	**Max Time**
5	0.00010	0.00010	0.00010	0.00000	12.05s	12.87s	13.95s
10	0.00019	2.96437	6.16792	1.94954	12.12s	14.05s	16.28s
30	4.95755	15.84911	22.22148	4.62482	700.32s	810.03s	905.79s

Figure 10: Table presenting the results of the Rastrigin function across 30 tests conducted using the Simulated Annealing method.

6.1.2 Michalewicz's Function

Hill Climbing - Best Improvement							
Dim	**Min**	**Avg**	**Max**	**St Dev**	**Min Time**	**Avg Time**	**Max Time**
5	-4.68761	-4.68759	-4.68729	0.00007	301.91s	307.16s	319.13s
10	-9.64786	-9.37927	-9.23506	0.09205	211.71s	217.22s	232.66s
30	-27.25796	-26.38198	-25.69986	0.41077	470.54s	478.27s	485.27s

Figure 11: Table presenting the results of the Michalewicz function across 30 tests conducted using the Hill Climbing - Best Improvement method.

Hill Climbing - First Improvement							
Dim	**Min**	**Avg**	**Max**	**St Dev**	**Min Time**	**Avg Time**	**Max Time**
5	-4.68761	-4.68734	-4.68669	0.00032	70.82s	71.65s	72.26s
10	-9.44471	-9.15291	-8.99773	0.11359	30.92s	31.66s	32.74s
30	-25.63137	-24.77787	-24.09906	0.37485	32.76s	33.89s	36.18s

Figure 12: Table presenting the results of the Michalewicz function across 30 tests conducted using the Hill Climbing - First Improvement method.

Hill Climbing - Worst Improvement							
Dim	**Min**	**Avg**	**Max**	**St Dev**	**Min Time**	**Avg Time**	**Max Time**
5	-4.68707	-4.67827	-4.66927	0.00750	869.05s	876.96s	892.08s
10	-9.33155	-8.87582	-8.56137	0.18141	654.26s	662.56s	674.29s
30	-23.93519	-22.72707	-21.74598	0.55913	1569.86s	1680.68s	1795.61s

Figure 13: Table presenting the results of the Michalewicz function across 30 tests conducted using the Hill Climbing - Worst Improvement method.

Simulated Annealing							
Dim	**Min**	**Avg**	**Max**	**St Dev**	**Min Time**	**Avg Time**	**Max Time**
5	-4.68761	-4.68761	-4.68761	0.0000	193.54s	197.91s	203.11
10	-9.64786	-9.48603	-9.33948	0.07661	229.12s	250.53s	300.51s
30	-28.47928	-27.99250	-27.42105	0.30132	1663.58s	1916.02s	2208.24s

Figure 14: Table presenting the results of the Michalewicz function across 30 tests conducted using the Simulated Annealing method.

6.1.3 De Jong I's Function

Hill Climbing - Best Improvement							
Dim	**Min**	**Avg**	**Max**	**St Dev**	**Min Time**	**Avg Time**	**Max Time**
5	0.00000	0.00000	0.00000	0.00000	21.93s	22.24s	22.73s
10	0.00000	0.00000	0.00000	0.00000	16.57s	16.93s	17.53s
30	0.00000	0.00000	0.00000	0.00000	44.28s	46.73s	55.27s

Figure 15: Table presenting the results of the De Jong I function across 30 tests conducted using the Hill Climbing - Best Improvement method.

Hill Climbing - First Improvement							
Dim	**Min**	**Avg**	**Max**	**St Dev**	**Min Time**	**Avg Time**	**Max Time**
5	0.00000	0.00000	0.00000	0.00000	3.63s	3.73s	4.11s
10	0.00000	0.00000	0.00000	0.00000	1.55s	1.61s	1.66s
30	0.00000	0.00000	0.00000	0.00000	1.67s	1.84s	2.17s

Figure 16: Table presenting the results of the De Jong I function across 30 tests conducted using the Hill Climbing - First Improvement method.

Hill Climbing - Worst Improvement							
Dim	**Min**	**Avg**	**Max**	**St Dev**	**Min Time**	**Avg Time**	**Max Time**
5	0.00000	0.00000	0.00000	0.00000	42.03s	43.48s	47.35s
10	0.00000	0.00000	0.00000	0.00000	32.37s	33.13s	35.11s
30	0.00000	0.00000	0.00000	0.00000	83.57s	89.03s	95.19s

Figure 17: Table presenting the results of the De Jong I function across 30 tests conducted using the Hill Climbing - Worst Improvement method.

Simulated Annealing							
Dim	**Min**	**Avg**	**Max**	**St Dev**	**Min Time**	**Avg Time**	**Max Time**
5	0.00000	0.00000	0.00000	0.00000	0.06s	0.07s	0.09s
10	0.00000	0.00000	0.00000	0.00000	0.52s	0.55s	0.59s
30	0.00000	0.00000	0.00000	0.00000	14.35s	15.24s	16.01s

Figure 18: Table presenting the results of the De Jong I function across 30 tests conducted using the Simulated Annealing method.

6.1.4 Schwefel's Function

Hill Climbing - Best Improvement							
Dim	**Min**	**Avg**	**Max**	**St Dev**	**Min Time**	**Avg Time**	**Max Time**
5	-2094.91441	-2094.82688	-2094.70606	0.07593	54.46s	54.89s	55.51s
10	-4189.30827	-3938.59267	-3768.43431	85.70114	40.86s	41.41s	42.31s
30	-11243.90581	-10566.57020	-10099.88615	285.95308	110.21s	113.85s	117.01s

Figure 19: Table presenting the results of the Schwefel function across 30 tests conducted using the Hill Climbing - Best Improvement method.

Hill Climbing - First Improvement							
Dim	**Min**	**Avg**	**Max**	**St Dev**	**Min Time**	**Avg Time**	**Max Time**
5	-2094.91408	-2072.69326	-2033.58360	18.04857	9.72s	9.87s	10.03s
10	-3921.43595	-3748.21071	-3622.21861	78.00288	4.11s	4.26s	4.51s
30	-10789.36123	-10008.18734	-9392.23870	289.08306	4.48s	4.83s	5.25s

Figure 20: Table presenting the results of the Schwefel function across 30 tests conducted using the Hill Climbing - First Improvement method.

Hill Climbing - Worst Improvement							
Dim	**Min**	**Avg**	**Max**	**St Dev**	**Min Time**	**Avg Time**	**Max Time**
5	-2041.14676	-2008.85269	-1987.38082	16.10686	320.31s	330.18s	338.85s
10	-3994.28876	-3777.74776	-3632.97880	92.30484	240.61s	254.97s	266.51s
30	-10491.52228	-9933.12033	-9516.44543	247.40521	661.85s	700.32s	752.81s

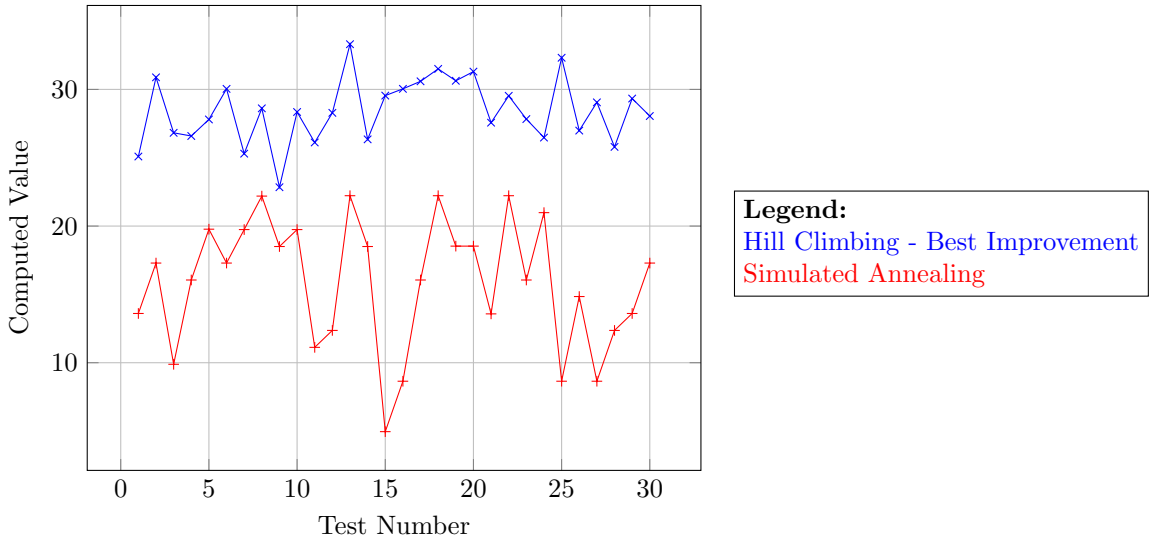
Figure 21: Table presenting the results of the Schwefel function across 30 tests conducted using the Hill Climbing - Worst Improvement method.

Simulated Annealing							
Dim	**Min**	**Avg**	**Max**	**St Dev**	**Min Time**	**Avg Time**	**Max Time**
5	-2094.91441	-2094.89486	-2094.80869	0.04113	47.23s	52.31s	58.75s
10	-4189.72124	-4189.59731	-4189.51412	0.05583	589.85s	612.58s	638.35s
30	-11941.02592	-11525.53900	-11155.25234	225.92438	1235.49s	1397.51s	1555.48s

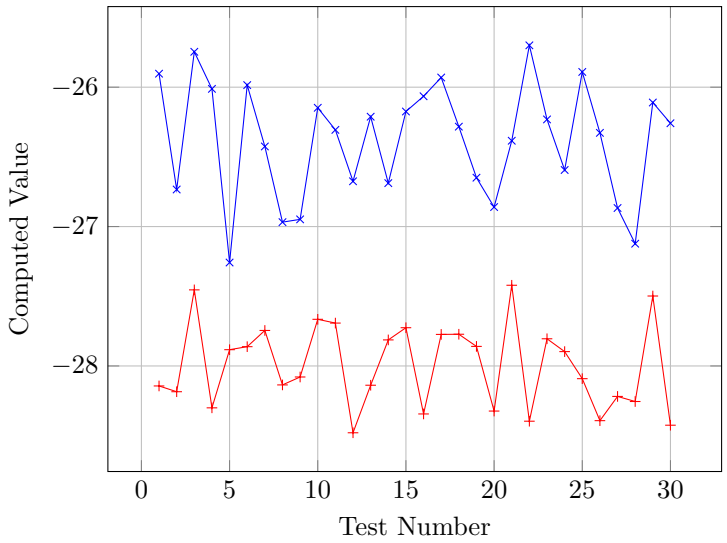
Figure 22: Table presenting the results of the Schwefel function across 30 tests conducted using the Simulated Annealing method.

6.2 Relevant Plots

Rastrigin Function 30D: Computed Values vs. Test Number

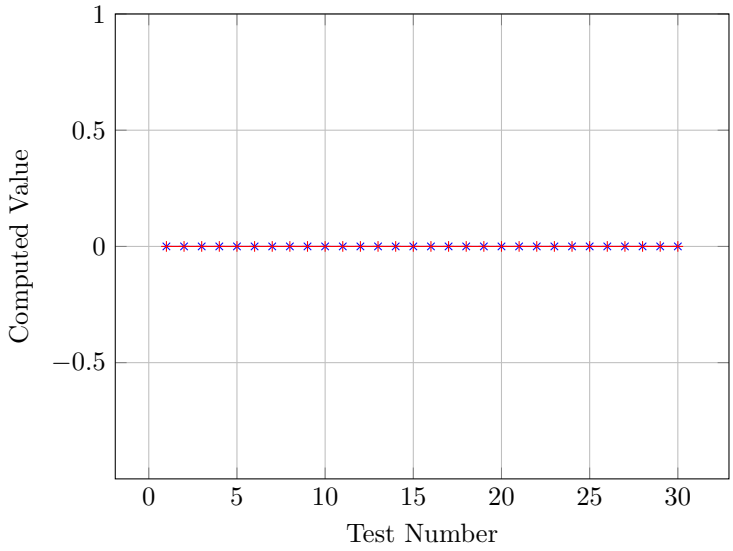


Michalewicz Function 30D: Computed Values vs. Test Number

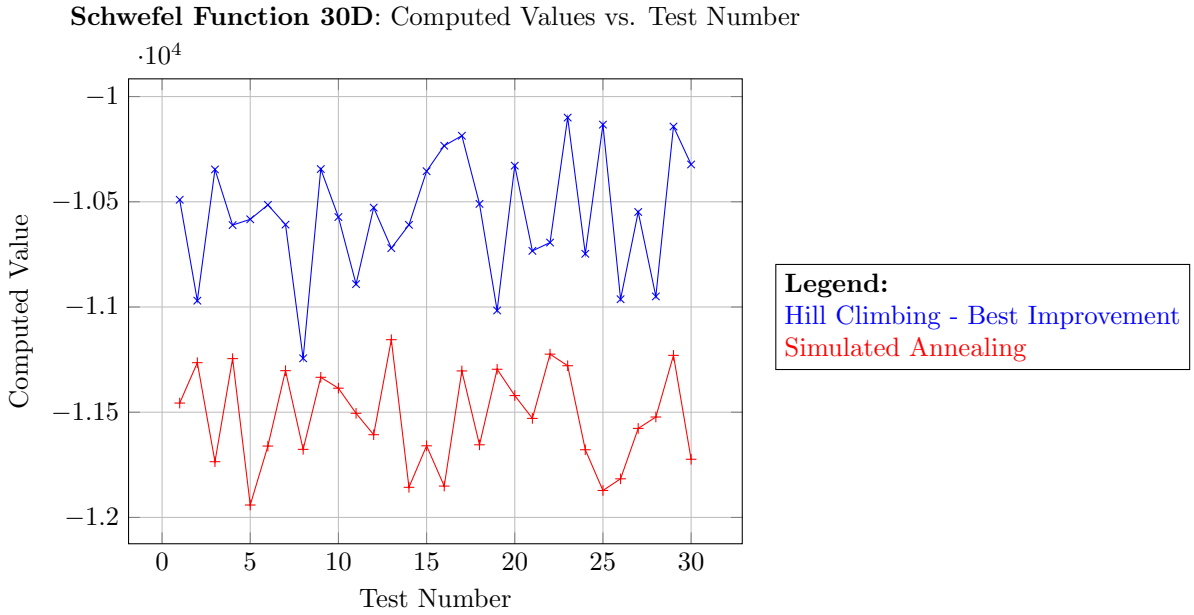


Legend:
Hill Climbing - Best Improvement
Simulated Annealing

De Jong I Function 30D: Computed Values vs. Test Number



Legend:
Hill Climbing - Best Improvement
Simulated Annealing



6.3 The Influence of Parameters on the Algorithms

While in the case of the Hill Climbing algorithm, the solution is relatively straightforward, adjusted primarily through the number of iterations and the precision at which the search space is discretized (factors that influence the runtime and the search space), in the case of the Simulated Annealing algorithm, there are various strategies that can be employed using parameters such as temperature and cooling ratio.

Initial Temperature

We have decided to set the **initial temperature** value at **1000**. This choice strikes a balance between exploration and exploitation, allowing the algorithm to begin with a relatively high temperature. It provides the algorithm with the flexibility to initially explore a wide range of solutions, which can be beneficial for escaping local optima and searching for global solutions.

Importance of the Cooling Schedule

The cooling schedule is a critical component of the Simulated Annealing algorithm as it defines the manner in which the temperature is decreased during the optimization process. The choice of the cooling schedule plays a pivotal

role in the algorithm's success, and it can greatly impact the efficiency and effectiveness of the search.

A very low cooling schedule can lead to an excessive number of iterations required to reach the global minimum. If there's a limit on the total number of iterations, this can result in an unsuccessful search. Conversely, an overly fast cooling schedule can risk trapping the algorithm in a local minimum or any smooth region of the error surface.

We have decided to use the **geometric cooling schedule** - which, in practice, is often employed -, defined as:

$$T_k = \alpha^k \cdot T_0$$

where:

$$\begin{aligned} T_k &\text{ is the temperature at iteration } k \\ \alpha &\text{ is the cooling ratio} \\ T_0 &\text{ is the initial temperature} \end{aligned}$$

We have chosen to set the cooling ratio, α, to **0.99**. This selection is made to prevent an excessively rapid cooling process, which can be detrimental to the algorithm's performance.

Stopping Criterion

The stopping criterion in Simulated Annealing refers to a set of conditions that determine when the algorithm should conclude its search for an optimal solution. In our specific implementation, we've established two halting conditions:

1. The first condition relates to the reduction of temperature to an extremely low value (in our case, **0.0001**). Once the temperature reaches or drops below this threshold, the algorithm terminates, indicating that the system has undergone sufficient cooling.
2. The second condition is centered on the potential to escape local optima by accepting worse neighbor solutions. If the algorithm perceives no further opportunities for such escapes, it ceases its search, recognizing that it has exhaustively explored the solution space.

7 Comparison and Interpretation

Our extensive experimentation has yielded valuable insights into the comparative performance of Simulated Annealing and Hill Climbing in the field of optimization. Our findings demonstrate that Simulated Annealing consistently outperforms Hill Climbing when it comes to finding optimal solutions, even when the number of iterations is lower. If the number of iterations were to be equal, Simulated Annealing would indeed yield even better results in optimization. However, it is essential to note that this enhanced performance comes at the cost of increased computational time. Simulated Annealing, owing to its probabilistic nature and the necessity for temperature reduction, operates at a slower pace than Hill Climbing, a greedy, local search method. Among all the Hill Climbing improvement methods, it is obvious that the Best Improvement method consistently yields the best results, outperforming both the First Improvement and Worst Improvement approaches. However, it's worth noting that the First Improvement method excels in terms of speed, making it a resource-efficient choice and, in certain scenarios, it can even compete with the Best Improvement method. On the other hand, the Worst Improvement method not only delivers the poorest results, but also consumes the most time.

We've established a fixed number of **100.000 evaluations** to meticulously observe how all algorithms perform to each other after an equivalent number of comparisons on every test function. This evaluation is conducted exclusively within the **30-dimensional** context, and the resulting value represents the mean of 30 distinct results.

RASTRIGIN				
Method	**HC BEST**	**HC FIRST**	**HC WORST**	**SA**
Value	54.53840	54.32682	503.00942	58.81216

Figure 23: Table presenting the results of the Rastrigin function after 100.000 evaluations using different optimization methods

MICHALEWICZ				
Method	**HC BEST**	**HC FIRST**	**HC WORST**	**SA**
Value	-24.58955	-24.34347	-11.25494	-24.45756

Figure 24: Table presenting the results of the Michalewicz function after 100.000 evaluations using different optimization methods

DE JONG I				
Method	**HC BEST**	**HC FIRST**	**HC WORST**	**SA**
Value	0.00000	0.00000	163.70747	0.00000

Figure 25: Table presenting the results of the De Jong I function after 100.000 evaluations using different optimization methods

SCHWEFEL				
Method	**HC BEST**	**HC FIRST**	**HC WORST**	**SA**
Value	-9755.39838	-10259.88760	6.38254	-9958.12843

Figure 26: Table presenting the results of the Schwefel function after 100.000 evaluations using different optimization methods

The results obtained after 100,000 evaluations showcase a remarkable similarity in performance across the various optimization algorithms, except for the "Worst Improvement" method, which consistently delivers notably suboptimal solutions. In the initial phase, when the optimization landscape is analogous to a high-temperature environment, "Simulated Annealing" provides similar values to "Hill Climbing." However, as the algorithm progresses, and the temperature gradually decreases, "Simulated Annealing" exhibits a notable transition, moving progressively closer to the true optimal solutions. This behavior underscores the dynamic and adaptive nature of Simulated Annealing, which adapts to the changing characteristics of the optimization problem as it progresses towards convergence.

Simulated Annealing and Hill Climbing both use **fitness based selection** — a new search solution only replaces the current solution if it is fitter. If the current solution is a local optimum, further progress can depend on having to "hop" to another (near) local optimum. If each search term has to be "near", then the probability of finding such a search solution can decrease exponentially with increasing dimensionality of the search space. Simulated Annealing and Hill Climbing both show **decreasing performanc**e on the tested functions as **dimensionality increases**, and this is consistent with the expected effects of the Curse of Dimensionality. Simulated Annealing can perform deteriorating steps by accepting search solutions with worse fitness. However, the probability of these steps being accepted still depends on the fitness difference and, as the time passes, Simulated Annealing eventually becomes Hill Climbing. Thus, the deteriorating steps that are most likely to be accepted are small deteriorating steps in the same attraction basin, or hops to near local optima in other attraction basins. Small deteriorating steps in the same attraction basin do not meaningfully change the search trajectory of Simulated Annealing, and a small

expansion of the area around a local optimum that will lead to a successful hop is still overwhelmed by the effects of the **Curse of Dimensionality**.

In what concerns the stopping criteria, Hill Climbing comes to an end after a certain number of iterations/local optimums achieved, while Simulated Annealing either ends when the temperature hits a predetermined level, either when it is certain that it cannot generate a better solution from that point onward. This takes us back to the idea that Hill Climbing has no tuning parameters, while Simulated Annealing has a **beginning temperature**, a **cooling schedule** and an **acceptance probability**.

8 Conclusion

Our experiments demonstrate that **Simulated Annealing consistently outperforms Hill Climbing** in function minimization, providing superior results, but with increased computational time. Hill Climbing, notably the Best Improvement method, offers a trade-off by delivering competitive outcomes swiftly. The choice between these algorithms should be made considering the balance between solution quality and runtime, emphasizing the significance of matching algorithm selection to problem-specific needs.

Using Simulated Annealing or Hill Climbing is a good choice because of its **ease of implementation**. There are also other methods such as **Genetic Algorithms** and **Hyper Heuristics** techniques that probably produce better results, but due to their implementation details, they should be used only when the mentioned algorithms do not provide good enough results. However, they still use Simulated Annealing or Hill Climbing at some point in their algorithm; either to generate the initial population or as a higher level heuristic.

References

[1] https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/

[2] https://www.geeksforgeeks.org/difference-between-hill-climbing-and-simulated-annealing-algorithm/

[3] https://towardsdatascience.com/hill-climbing-optimization-algorithm-simply-explained-dbf1e1e3cf6c

[4] https://en.wikipedia.org/wiki/Test_functions_for_optimization

[5] https://en.wikipedia.org/wiki/Hamming_distance

[6] https://ai.stackexchange.com/questions/8983/how-is-simulated-annealing-better-than-hill-climbing-methods

[7] https://www.sfu.ca/~ssurjano/rastr.html

[8] https://www.sfu.ca/~ssurjano/michal.html

[9] https://www.sfu.ca/~ssurjano/spheref.html

[10] https://www.sfu.ca/~ssurjano/schwef.html

[11] https://en.wikipedia.org/wiki/Metropolis%E2%80%93Hastings_algorithm

[12] https://www.researchgate.net/publication/238690391_Simulated_annealing_overview

[13] https://profs.info.uaic.ro/~eugennc/teaching/ga/

[14] "Simulated Annealing" — Rafael E. Banchs

[15] "Intelligent Nanotechnology", Chapter 4 - *Nanophotonic devices based on optimization algorithms* — Cuicui Lu, Hongyi Yuan, Nianen Zhang (2023).

[16] "LPWAN Technologies for IoT and M2M Applications", Chapter 10 - *Energy optimization in low-power wide area networks by using heuristic techniques*, Pages 199-223 — Zeinab E. Ahmed, Rashid A. Saeed, Amitava Mukherjee, Sheetal N. Ghorpade (2020)

[17] "Simulated Annealing vs. Hill Climbing in Continuous Domain Search Spaces" — Stephen Chen, Shehnaz Islam, Shanshan Lao (2021).

[18] "Mathematical and Computer Modelling" - *Simulated annealing: Practice versus theory*, Pages 29-57 — L. Ingber (1993)

[19] "Comparison of simulated annealing and hill climbing in the course timetabling problem" — Kenekayoro Patrick (2012).