Bitwise Hill Climbing and Simulated Annealing

Popa Andrei, Octavian-Ștefan Regatun

November 29, 2023

Abstract

This document focuses on making a comparative study between the results obtained using Hill Climbing and Simulated Annealing algorithms and the results obtained using a Genetic Algorithm. The objective was to find the global minimum of four distinct mathematical functions: De Jong 1, Schwefel's, Rastrigin's, and Michalewicz's. The results showed us that the Genetic Algorithm can find much better solutions, much closer to the global minimum or sometimes actually find it and it also lowers the time of execution by a lot.

1 Introduction

Following the conclusion from the last paper that the Simulated Annealing seemed that it performed better because of the broader domain of values that we were working on, we will continue to analyse and compare those results with the results of a Genetic Algorithm. Intuitively we think that the Genetic Algorithm will also give us better results because it also allows us to explore more of the domain and not get stuck in a local minimum because of genetic operations such as crossover and mutation.

The objective of this document is to compare the results of of the Hill Climbing and Simulatead Annealing algorithms ran on De Jong 1, Schwefel's, Rastrigin's and Michalewicz's functions methods with the results of a Genetic Algorithm implemented by us the authors.

HCFI = Hill Climbing using First Improvement

HCBI = Hill Climbing using Best Improvement

HCWI = Hill Climbing using Worst Improvement

SA = Simulated Annealing

GA = Genetic Algorithm

2 Methods

2.1 Data Representation

Bitwise representation is a crucial concept in the realms of optimization algorithms like Hill Climbing, Simulated Annealing and Genetic Algorithm. It serves as a foundational framework for encoding input data into a binary format. Understanding and utilizing bitwise

representation is fundamental in enabling these algorithms to efficiently navigate solution spaces for various optimization tasks.

2.1.1 Bitwise Representation

We will use these formulas to get the length of the data in a bitwise representation:

$$10^{\text{precision}} = \text{number of points in an interval}$$

$$10^{\text{precision}} \cdot (b - a) = \text{number of points in the interval} [a, b]$$

$$M = (10^{\text{precision}} \cdot (b - a))^N \quad \text{for} \quad N \text{ dimensions}$$

$$\log_2(M) = \text{number of bits required for representation}$$

The length of an input with N dimensions in bits is given by:

$$N \cdot \lceil \log_2(10^{\text{precision}} \cdot (b - a)) \rceil = L \quad \text{(length of input in bits)}$$

2.1.2 Bitwise Representation to a Number from Function's Domain

We will use this formula to map the bitwise data to a real number from a function domain:

$$\text{minimum\_bound} = \text{a function's minimum domain value}$$
$$\text{maximum\_bound} = \text{a function's maximum domain value}$$

$$\text{bits\_per\_dimension} = \frac{\text{length}}{\text{dimensions}}$$

$$\text{sum} = \sum_{i=0}^{bits\_per\_dimension-1} (2^i \cdot \text{bit}[d \cdot \text{bits\_per\_dimension} + i])$$

$$\text{real\_val} = \text{minimum\_bound} + \frac{\text{sum} \cdot (\text{maximum\_bound} - \text{minimum\_bound})}{2^{\text{bits\_per\_dimension}} - 1}$$

2.2 Hill Climbing

Hill Climbing is a heuristic search algorithm used for mathematical optimization problems. The main idea behind Hill Climbing is analogous to climbing the peak of a hill or mountain. Starting from an arbitrary solution, the algorithm iteratively makes incremental changes, selecting the neighbor with the highest fitness value. The process is repeated until there are no more possible improvements.

Key Features:

- *Local Search*: Hill Climbing operates in the vicinity of the current solution and doesn't require knowledge of the entire search space.
- *Deterministic Nature*: Unlike some other optimization algorithms, Hill Climbing is deterministic, always making the locally optimal choice.
- *Speed*: Due to its local nature and simplicity, Hill Climbing can be faster than more complex algorithms, especially in well-defined search spaces.

Operating Mechanism: The basic steps of the Hill Climbing algorithm are:

1. Start with an arbitrary solution.
2. Determine the neighbors of the current solution (usually solutions that are a small modification away).
3. If any neighbor has a better fitness than the current solution, move to that neighbor.
4. Repeat the process until no better neighbor is found.

Variants: Several variants of Hill Climbing enhance its basic mechanism:

- *Stochastic Hill Climbing*: Chooses at random from the uphill moves; the probability of selection can vary with the steepness of the uphill move.
- *Step-Size Change Hill Climbing*: Varies the size of the steps it takes with each move in the search space.
- *Random-Restart Hill Climbing*: Conducts a series of hill climbing searches from randomly generated initial conditions, leading to better chances of finding a global maximum.

Advantages and Limitations: While Hill Climbing is straightforward and often efficient, it has its limitations. The most significant drawback is its susceptibility to getting stuck in local optima because it uses the Greedy paradigm. Since Hill Climbing only considers the neighboring space of the current solution, it can miss out on better solutions that are further away. This limitation is addressed in its variant, Random-Restart Hill Climbing, which can provide a better chance of finding the global optimum by restarting the search from different positions.

Methods of Implementation

There are several directions which the algorithm can take.

These are the three most popular criterion of choosing the most fitting neighbour::

- *Best Improvement*: The algorithm searches for the most fitting neighbour from the vicinity of the number and continues to use the found value. This is usually the most efficient method.
- *First Improvement*: The algorithm searches for the first value that is more fitting than the current value and then uses the found value.
- *Worst Improvement*: The algorithm searches for the worst-fitting neighbour from the vicinity of the number and continues to use the found value.

2.2.1 Optimization - Improvements

Contorising the number of improvements was implemented to serve the avoidance of local optima, a weak point for the Hill Climbing algorithm. After a fixed maximum number of improvements, the algorithm will halt execution, so it will not get stuck in irrelevant searches of the local result.

2.2.2 Optimization - Mutation Rate

The mutation rate is inspired by genetics, from the term with the same name. This probabilistic optimization is meant to avoid local optima by creating a balance between exploration and exploitation, through the means of a probabilistic approach. The exploration aspect of it is proportional to the probability: if the probability is increased, the algorithm will find various neighbours. Regarding the Bitwise Hill Climbing, the exploration will be done by flipping some bits stochastically.

2.3 Simulated Annealing

Simulated Annealing (SA) is a probabilistic optimization technique inspired by the physical annealing process used in metallurgy. Annealing refers to the heating of a material and then its controlled cooling to enhance its properties. In the context of optimization, SA is used to find an approximation to the global optimum of a function.

Key Features:

- *Exploration and Exploitation*: SA combines the exploration of the solution space with the exploitation of local optima. This balance is controlled by the algorithm's temperature.
- *Variable Probability*: SA accepts worse solutions with a probability that decreases with temperature. This mechanism allows the algorithm to escape local optima in the initial phases and focus on refining the solution in later stages.
- *Flexibility*: SA can be applied to a wide variety of optimization problems, from combinatorial to continuous.

Operating Mechanism: SA starts with an initial solution and a high initial temperature. At each step, a neighboring solution is generated by making a slight modification to the current solution. If the neighboring solution is better, it is accepted as the new current solution. If it is worse, it may still be accepted with a probability that depends on the difference between the qualities of the two solutions and the current temperature. After a fixed number of iterations, the temperature is reduced using a cooling schedule, and the process is repeated until the temperature reaches zero or another stopping criterion is met.

Cooling Schedule: The cooling schedule controls how the temperature decreases over time. A common strategy is geometric cooling, where the temperature is multiplied by a constant at each iteration. The correct choice of cooling schedule is crucial for SA's efficiency.

Advantages and Limitations: SA is a robust technique that can provide quality solutions for a wide range of problems. However, its efficiency depends on the correct setting of parameters, such as the initial temperature, the cooling schedule, and the number of iterations at each temperature. Also, while SA can avoid local optima, there is no guarantee that it will find the global optimum.

2.3.1 Optimization - Hydibrisation

A way of optimizing the algorithm is combining a part of the improvement methods. A good combination that is very efficient is using both first improvement and best improvement. The Simulated Annealing algorithm is derived from Hill Climbing and finds a fitting neighbour, having the option to retrace in case of not finding the wanted value. Then, it compares the found value with the value found using Hill Climbing based on the current value.

2.4 Genetic Algorithms

Genetic Algorithms (GAs) are adaptive heuristic search algorithms based on the evolutionary ideas of natural selection and genetics. They are a subset of evolutionary algorithms used for solving both constrained and unconstrained optimization problems.

2.4.1 Key Features

- *Population-Based Search*: GAs work with a population of solutions, allowing them to explore multiple solutions simultaneously.
- *Use of Genetic Operators*: GAs leverage operators such as selection, crossover, and mutation to evolve the solutions.
- *Robustness*: They are broadly applicable and robust against the landscape of the search space, making them suitable for complex optimization problems.

2.4.2 Operating Mechanism

The basic steps of Genetic Algorithms are:

1. Initialize a population of solutions randomly.
2. Evaluate the fitness of each solution.
3. Select solutions for reproduction based on their fitness.
4. Apply crossover and mutation to generate new offspring.
5. Replace the current population with the new population (offspring).
6. Repeat the process for a set number of generations or until a termination condition is met.

2.4.3 Variants

Several variants of Genetic Algorithms exist, such as:

- *Elitism*: Maintains the best solutions in the population across generations.
- *Steady-State GA*: Replaces only a few members of the population at a time.
- *Multi-Objective GA*: Optimizes more than one objective function simultaneously.

2.4.4 Advantages and Limitations

Genetic Algorithms are highly versatile and can be applied to a wide range of problems. However, they can be computationally intensive and may not guarantee an optimal solution. The randomness in GA can both be an advantage (exploration) and a limitation (may lead to inconsistent results).

2.4.5 Methods of Implementation

In Genetic Algorithms, the fitness function, the selection, crossover, and mutation are key elements:

The Fitness Function

In GAs, a fitness function is used to evaluate how close a given solution is to the optimum solution of the problem being solved. The fitness value is calculated for each individual in the population. The higher the fitness value, the better the solution.

Specific to the task of finding the minimum value of a given function, a more suitable method of calculating the fitness is through the formula:

$$fitness = constant/(function\_value + small\_value)$$

Crossover and Mutation Operations

Crossover and mutation are genetic operators used to vary the programming of chromosomes from one generation to the next.

- *Crossover*: This operator involves combining the genetic information of two parents to generate new offspring. It is hoped that this new offspring will have the best traits of each of its parents. The transfer of the genetic information is determined by a number of n cut points, which are determined heuristically. The value of n is proportional with the rate of exploration of the algorithm.
- *Mutation*: In mutation, the solution is modified to explore the solution space. It introduces genetic diversity by randomly flipping some of the bits in a chromosome.

Selection Methods

- *Roulette Wheel Selection*: Selects individuals based on a probability proportional to their fitness.
- *Tournament Selection*: Selects the best individual from a random subset of the population.
- *Rank-Based Selection*: Selects based on the rank of individuals rather than their fitness value.

2.4.6 Optimization Techniques in Genetic Algorithms

Optimizations in Genetic Algorithms are crucial for enhancing their performance and solution quality. Here are some specific optimizations:

- *Elitism*: Elitism is a technique where the best-performing individuals from the current generation are directly carried over to the next generation. This ensures that the quality of the population does not decrease and helps maintain the best solutions found during the search process.
- *Gray Code Representation*: In Gray Code representation, consecutive values differ by only one bit, which helps in reducing the Hamming distance between successive chromosomes. This is beneficial in genetic algorithms as it can lead to a smoother and more progressive search, especially in optimization problems where small changes in the chromosome can lead to significant changes in the solution space.
- *Uniform Crossover*: Uniform Crossover is a crossover technique where each gene in the offspring is chosen independently from one of the corresponding genes of the parent chromosomes. Unlike single-point or multi-point crossover, uniform crossover does not rely on a fixed crossover point; instead, it treats each gene position individually, potentially leading to a more diverse set of offspring. This can increase the explorative power of the genetic algorithm.

3 Function Descriptions

In our study, we focused on four mathematical functions: De Jong 1, Schwefel's, Rastrigin's, and Michalewicz's. Each of these functions has its unique characteristics and challenges for optimization algorithms. Here, we provide a more detailed description of each function and its significance in optimization studies.

3.1 De Jong 1

De Jong's first function, often referred to as the Sphere function, is a simple quadratic function. It is defined as:

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

Where x is a vector in n-dimensional space.

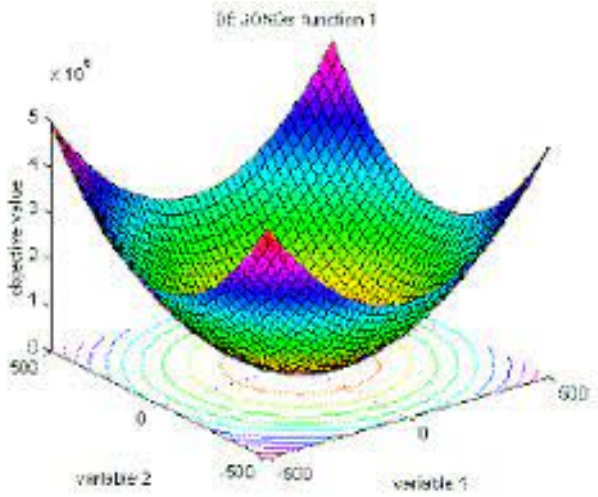

Figure 1: DeJong1

Characteristics:

- The function is continuous and convex.
- The global minimum is at the origin, where $f(x) = 0$.
- It serves as a basic test problem for optimization algorithms due to its simplicity.

3.2 Schwefel's Function

Schwefel's function is a complex, multimodal function with many local minima, making it challenging for optimization algorithms. It is defined as:

$$f(x) = 418.9829n - \sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|})$$

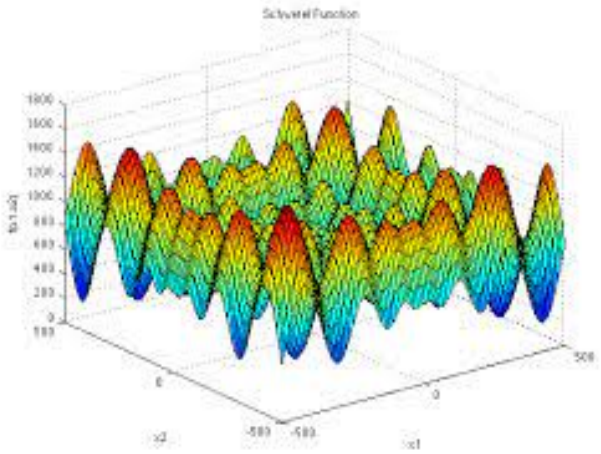

Figure 2: Schwefel

Characteristics:

- The function has a high number of local minima.
- The global minimum is at $x_i = 420.9687$ for all i, where $f(x) = 0$.
- It is often used to test the performance of algorithms in escaping local minima.

3.3 Rastrigin's Function

Rastrigin's function is a typical example of non-linear, multimodal functions. It is defined as:

$$f(x) = An + \sum_{i=1}^{n} \left[x_i^2 - A\cos(2\pi x_i)\right]$$

Where $A = 10$ is a constant.

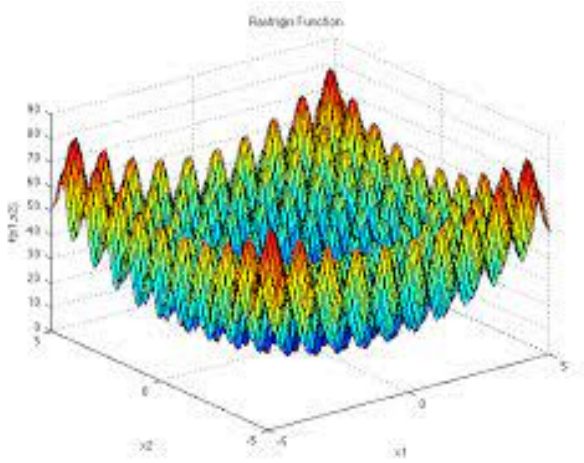

Figure 3: Rastrigin

Characteristics:

- The function has a regular pattern of local minima.
- The global minimum is at the origin, where $f(x) = 0$.
- It is commonly used to test the performance of algorithms in multimodal optimization tasks.

3.4 Michalewicz's Function

Michalewicz's function is designed to test optimization algorithms on multimodal functions with steep ridges and valleys. It is defined as:

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

Where m is a constant that determines the steepness of the valleys and ridges.

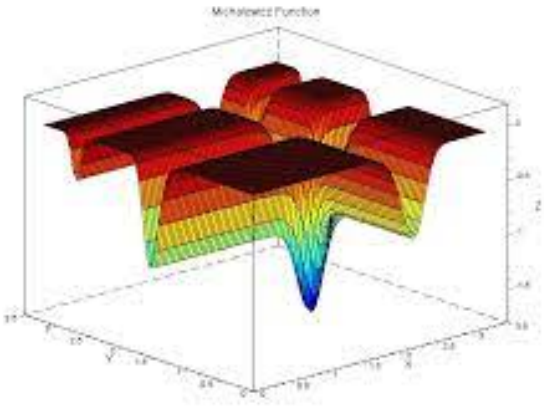

Figure 4: Michalewicz

Characteristics:

- The function becomes more difficult to optimize as m increases.
- It is used to test the ability of algorithms to navigate complex landscapes with steep ridges and valleys.

4 Applications of the Functions

Each of these functions serves as a benchmark in optimization studies. While they are mathematical constructs, they can represent real-world problems in various domains, from engineering design to financial modeling. By understanding the behavior of optimization algorithms on these functions, researchers can gain insights into how these algorithms might perform on real-world problems with similar characteristics.

5 Experiment

5.1 Function Values (5 Dimensions)

Rastrigin - 5D	HCFI	HCBI	HCWI	SA	GA
	Value	Value	Value	Value	Value
Best	0	0	0	0	0.00000002
Average	0	0	0	0	0.00000002
Worst	0	0	0	0	0.00000002
Standard Deviation	0	0	0	0	0
Median Time	81.7	33.6	1.9	66.5	7.12598

Michalewicz - 5D	HCFI	HCBI	HCWI	SA	GA
	Value	Value	Value	Value	Value
Best	-4.687658	-4.687658	-0.060601	-4.687658	-4.68766000
Average	-4.687658	-4.687658	0	-4.687658	-4.68766000
Worst	-4.687658	-4.687658	0	-4.687658	-4.68766000
Standard Deviation	0	0	0.026255	0	0.2321
Median Time	109.5	52.1	2.5	92.7	7.12598

Schwefel - 5D	HCFI	HCBI	HCWI	SA	GA
	Value	Value	Value	Value	Value
Best	-4.839293	0	0	-4.960689	0
Average	-4.602191	0	0	-4.956196	0
Worst	-4.558086	0	2137.680267	-4.84085	0
Standard Deviation	0.1151565417	0	35.68954128	0.05331361337	0
Median Time	93	0	3.4	97.9	7.12598

De Jong - 5D	HCFI	HCBI	HCWI	SA	GA
	Value	Value	Value	Value	Value
Best	0	0	0.064487	0	6.84953
Average	0	0	0.277988	0	7.10891
Worst	0	0	0.429912	0	7.60322
Standard Deviation	0	0	0.1351923526	0	0.2321
Median Time	72.3	33.6	2.1	66.8	7.12598

Rastrigin - 10D	HCFI	HCBI	HCWI	SA	GA
	Value	Value	Value	Value	Value
Best	1.23582	0.99665	0.47561	0.24769	0.00000005
Average	2.23390	1.03947	1.99644	1.00428	0.00000005
Worst	3.47165	1.23785	2.277	1.07137	0.00000005
Standard Deviation	6.08728	2.92284	0.04472	1.68671	0.00000000
Median Time	125.2	153.4	12.6	198.6	14.3328

Michalewicz - 10D	HCFI	HCBI	HCWI	SA	GA
	Value	Value	Value	Value	Value
Best	-9.653261	-9.652015	-0.540338	-9.645489	-9.65524000
Average	-9.588102	-9.621623	-0.467639	-9.631049	-9.65072680
Worst	-9.562992	-9.607992	-0.115563	-9.604407	-9.61763000
Standard Deviation	0.03797998	0.01901086	0.17943471	0.01705667	0.01234589
Median Time	190.4	219.1	6.6	286.4	13.5639

Schwefel - 10D	HCFI	HCBI	HCWI	SA	GA
	Value	Value	Value	Value	Value
Best	23.4061	-9.116129	3880.96369	-9.020024	0.00012728
Average	120.445029	-8.919016	4222.216032	-8.59476	0.00012728
Worst	142.065298	-8.866232	4395.975583	-8.329221	0.00012728
Standard Deviation	46.36080917	0.10925953	217.0061208	0.28151268	0.00000000
Median Time	152.9	228.5	10.5	326.1	18.807

De Jong - 10D	HCFI	HCBI	HCWI	SA	GA
	Value	Value	Value	Value	Value
Best	0	0	5.817876	0	0.00000000
Average	0	0	6.471136	0	0.00000000
Worst	0	0	7.305963	0	0.00000000
Standard Deviation	/	/	0.62089513	/	0.00000000
Median Time	106.6	106.6	5.5	206.7	13.8474

Rastrigin - 30D	HCFI	HCBI	HCWI	SA	GA
	Value	Value	Value	Value	Value
Best	34.28834	25.48046	260.87720	28.6126	0.00000057
Average	35.99085	27.9785	293.34450	30.46100	0.02474556
Worst	39.26313	30.05344	302.34923	31.17775	1.23725000
Standard Deviation	3.49399	0.99146	0.78421	2.74372	0.17497349
Median Time	162.3	1778.2	234.4	1425.5	41.6816

Michalewicz - 30D	HCFI	HCBI	HCWI	SA	GA
	Value	Value	Value	Value	Value
Best	-27.343628	-3.801449	-4.252796	-26.798798	-29.18320000
Average	-26.819411	-3.232143	-4.116171	-26.157758	-29.00547600
Worst	-26.482812	-2.141482	-2.447032	-26.088068	-27.93320000
Standard Deviation	0.31718746	0.60369931	0.82323110	0.37291550	0.25985783
Median Time	354.4	63.9	44.8	1793.7	38.1317

Schwefel - 30D	HCFI	HCBI	HCWI	SA	GA
	Value	Value	Value	Value	Value
Best	2069.019227	853.274205	12105.56047	963.821133	0.00038185
Average	2525.312332	874.929595	12967.47401	1073.283964	0.32212541
Worst	2757.021887	977.034059	13117.71871	1162.558345	0.93845000
Standard Deviation	297.2399946	50.10977461	415.6170331	85.24056691	0.26080078
Median Time	168.2	736.8	76.2	2354.1	55.4348

De Jong - 30D	HCFI	HCBI	HCWI	SA	GA
	Value	Value	Value	Value	Value
Best	0	0	5.817876	0	0.00000000
Average	0	0	6.471136	0	0.00000000
Worst	0	0	7.305963	0	0.00000000
Standard Deviation	0	0	0.62089513	0	0.00000000
Median Time	106.6	106.6	5.5	206.7	40.4762

5.2 Parameters and their Significance

In the provided code for Bitwise Hill Climbing , Simulated Annealing and Genetic Algorithm, several parameters play a crucial role in determining the algorithm's performance. Properly understanding and tuning these parameters is essential for achieving optimal results.

5.2.1 Bitwise Hill Climbing

From the code, the primary parameters for Bitwise Hill Climbing include:

Number of Bits: This determines the length of the binary string representation of a solution. The choice of the number of bits can influence the granularity of the solution space. A larger number of bits allows for a more detailed representation but increases the search space's size.

Mutation Rate: This parameter determines the likelihood of flipping a bit during the mutation phase. A higher mutation rate can increase the exploration of the solution space but might also introduce too much randomness.

Recommendation: It's essential to balance exploration and exploitation. Starting with a moderate mutation rate and adjusting based on preliminary results can be beneficial.

5.2.2 Simulated Annealing

For Simulated Annealing, the code suggests the following key parameters:

Initial Temperature: This sets the starting temperature for the annealing process. A higher value allows for more exploration initially, while a lower value makes the algorithm more conservative.

Cooling Rate: Determines how quickly the temperature decreases. A value closer to 1 means a slower cooling process, allowing more iterations at each temperature level.

Stopping Criterion: This could be a specific temperature threshold or a number of iterations without improvement. It ensures the algorithm doesn't run indefinitely.

Recommendation: The choice of parameters should be based on the problem's nature and the desired balance between exploration and exploitation. Running tests with different parameter combinations can help identify the optimal settings.

5.2.3 Genetic Algorithm

Number of Generations: These parameters characterize the evolutionary aspect of the Genetic Algorithm.

Cooling Rate: Determines how quickly the temperature decreases. A value closer to 1 means a slower cooling process, allowing more iterations at each temperature level.

Stopping Criterion: This could be a specific temperature threshold or a number of iterations without improvement. It ensures the algorithm doesn't run indefinitely.

6 Conclusions

The comprehensive study explored the comparative effectiveness of Genetic Algorithms, Hill Climbing, and Simulated Annealing in finding the global minima of four mathematical functions: De Jong 1, Schwefel's, Rastrigin's, and Michalewicz's. This analysis, spanning various dimensions (5, 10, and 30), offers critical insights into the strengths and limitations of these algorithms in optimization challenges. Key observations from the study include:

1. **Dimensional Variability in Performance**: Each algorithm exhibited different levels of efficiency across varying dimensions. Genetic Algorithms showed remarkable adaptability in higher dimensions, while Hill Climbing and Simulated Annealing had varied success rates depending on the problem's complexity. This finding emphasizes the necessity of choosing the right algorithm based on the dimensionality and nature of the optimization task.

2. **Genetic Algorithms' Edge**: A notable aspect was the consistent performance of Genetic Algorithms, particularly in higher-dimensional spaces, where their ability to balance exploration and exploitation shone through. This highlights the advantage of evolutionary algorithms in complex optimization scenarios.

3. **Comparative Algorithmic Analysis**: The study revealed the distinctive features of Hill Climbing and Simulated Annealing in comparison to Genetic Algorithms. While the former excel in specific scenarios, Genetic Algorithms generally provide more robust and consistent solutions across various test cases and dimensions.

4. **Impact of Algorithmic Parameters**: The influence of key parameters like mutation rate in Genetic Algorithms, and temperature in Simulated Annealing was evident. These parameters critically determine the algorithms' search behavior and effectiveness, underlining the importance of careful parameter tuning in optimization tasks.

5. **Future Research Opportunities**: The study opens avenues for further exploration in areas such as:

    - *Enhanced Hybrid Models*: Investigating combinations of Genetic Algorithms with elements of Hill Climbing or Simulated Annealing could lead to more sophisticated and efficient optimization strategies.
    - *Dynamic Parameter Adjustment*: Developing adaptive algorithms that can modify their parameters in real-time based on performance feedback might enhance efficacy and adaptability.

In summary, this research provides valuable insights into the comparative performance of Genetic Algorithms, Hill Climbing, and Simulated Annealing across different dimensions and mathematical functions. The findings emphasize the significance of choosing appropriate algorithms and fine-tuning parameters for optimal results. As the optimization field progresses, such comparative studies will be crucial in advancing more efficient and effective algorithmic solutions.

[1] Bäck, T., Schütz, M. (1996). *Intelligent mutation rate control in canonical genetic algorithms*. In: Raś, Z.W., Michalewicz, M. (eds) Foundations of Intelligent Systems. ISMIS 1996. Lecture Notes in Computer Science, vol 1079. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-61286-6_141

[2] SFU. *Rastrigin Function*. Image source: https://www.sfu.ca/~ssurjano/rastr.png.

[3] SFU. *Michalewicz Function*. Image source: https://www.sfu.ca/~ssurjano/michal.png.

[4] ESA. *Schwefel Function*. Image source: https://esa.github.io/pagmo2/_images/schwefel.png.

[5] GEATbx. *De Jong Function*. Image source: http://www.geatbx.com/docu/fcnindex-msh_f1_500-2.gif.

[6] *Hill Climbing Optimization Algorithm - Simply Explained*. Towards Data Science. https://towardsdatascience.com/hill-climbing-optimization-algorithm-simply-ex

[7] *Understanding Hill Climbing in AI*. Section Engineering Education. https://www.section.io/engineering-education/understanding-hill-climbing-in-ai/

[8] *LaTeX Tutorial*. https://www.overleaf.com/learn/latex/Tutorials/

[9] MIT OpenCourseWare. *Gradient Descent, Step Size, Rate of Convergence*. YouTube.

[10] GeeksforGeeks. *ML - Stochastic Gradient Descent (SGD)*.

[11] James Brookhouse and Alex Freitas. *Fair Feature Selection: A Comparison of Multi-Objective Genetic Algorithms*. (2023). https://arxiv.org/pdf/2310.02752.

[12] Krutika Sarode and Shashidhar Reddy Javaji. *Hybrid Genetic Algorithm and Hill Climbing Optimization for the Neural Network*. (2023). https://arxiv.org/pdf/2308.13099.

[13] Alexander E. I. Brownlee and others. *Enhancing Genetic Improvement Mutations Using Large Language Models*. (2023). https://arxiv.org/pdf/2310.19813.

[14] Tarek Faycal and Claudio Zito. *Direct Mutation and Crossover in Genetic Algorithms Applied to Reinforcement Learning Tasks*. (2022). https://arxiv.org/pdf/2201.04815.

[15] Endah Rokhmati Merdika Putri and others. *A Deep-Genetic Algorithm (Deep-GA) Approach for High-Dimensional Nonlinear Parabolic Partial Differential Equations*. (2023). https://arxiv.org/pdf/2311.11558.

[16] *Genetic Algorithm Course on Udemy*. https://www.udemy.com/course/geneticalgorithm/?utm_source=adwords&utm_medium=udemyads&utm_campaign=LongTail-New_la.EN_cc.ROWMTA-B&utm_content=deal4584&utm_term=_._ag_101378276820_._ad_533999945410_._kw__._de_c_._dm__._pl__._ti_dsa-1007766171032_._li_1011828_._pd__._&matchtype=&gad_source=1&gclid=CjwKCAiAvJarBhA1EiwBwE. (Accessed: November 25, 2023).

[17] *Genetic Algorithms Course Website*. https://profs.info.uaic.ro/~eugennc/teaching/ga/.

[18] *A Comparative Study between Hill Climbing and Simulated Annealing*. https://github.com/octavian-regatun/h1/latex.pdf.