Group 01

EVOLUTION OF SIMULATED ANNEALING AND ITS APPLICATIONS

Abstract

Simulated annealing (SA) is a probabilistic optimization technique inspired by the annealing process in metallurgy. This paper explores the evolution of the SA algorithm and its application across various domains, including steel manufacturing, metalworking, jewelry making, automotive cooling systems, semiconductor production, and the production of glass and ceramics. The evolution of the algorithm and how the metropolis criterion is applied for simulated annealing optimization is explained and Central to the study is the introduction of the Markov chain to enhance the efficiency and effectiveness of the algorithm. The Markov chain allows for systematic exploration of the solution space, balancing exploration and exploitation, and improving convergence towards the global optimum. The study involves how the algorithm behaves for various Markov chain lengths and tested for accuracy.

Here the impact of different cooling schedules is analyzed, such as exponential and linear schedules, on the performance of the SA algorithm. The modified algorithm, which incorporates a Markov chain with a fixed length at each temperature iteration, demonstrates superior performance in achieving optimal solutions.

Benchmark functions, both unimodal and multimodal, are utilized to evaluate the reliability and efficiency of the proposed algorithm. Here the algorithm is tested for a selected uni-modal function and a selected multi-modal function and the results show a significant improvement in the convergence rate and accuracy of the SA algorithm with the integration of the Markov chain.

This research provides valuable insights into the optimization capabilities of simulated annealing, offering a robust framework for solving complex optimization problems in various industrial applications.

Introduction

Inspired by the physical annealing process in metallurgy, Simulated Annealing (SA) is a potent probabilistic technique for addressing both bound-constrained and unconstrained optimization problems. This research aims to investigate the development of simulated annealing and its uses in many industries, addressing the need for effective optimization techniques that can traverse vast and intricate solution spaces in order to identify global optima.

Based on the Metropolis-Hastings method from statistical mechanics, simulated annealing was first introduced by Kirkpatrick, Gelatt, and Vecchi in 1983 and has since been widely used in computer science, operations research, and engineering. In SA, the Boltzmann Distribution—a fundamental concept in statistical physics—is essential for estimating the likelihood of accepting suboptimal solutions when exploring the solution space. Markov Chains offer a methodical and adaptable approach to navigating possible solutions by simulating the transitions between states. To improve SA performance, researchers have created and analysed a variety of cooling regimens, including exponential and linear strategies.

The organization of this paper is to give a thorough grasp of simulated annealing and its uses. The theoretical underpinnings, such as the Boltzmann Distribution and Markov Chains, are thoroughly examined at first, and then various cooling regimens are compared and their effects on the algorithm's performance are evaluated. The real-world uses of SA are investigated in sectors like semiconductor manufacturing, jewellery making, steel manufacturing, metalworking, cooling systems in automobiles, and glass and ceramics manufacture. Results from benchmark functions and case studies involving two-variable optimization issues are used to assess the efficacy of SA.

The research shows that simulated annealing is a reliable and adaptable optimization method that can handle a variety of challenging issues. The performance of SA is greatly influenced by various cooling schedules; in general, exponential schedules lead to a faster convergence to optimal solutions than linear schedules. The effectiveness of SA in reaching global optima is demonstrated by its application to benchmark functions and real-world industrial challenges. Markov Chains are introduced, which improves the exploration process and yields more accurate and trustworthy results.

Materials and Methods

The goal of this report is to develop an algorithm for simulated annealing using python programming language to find the equilibrium point with decreasing temperature for different cooling schedules. The primary objectives include creating an effective algorithm, investigating the impact of various cooling schedules, utilizing the Boltzmann distribution to guide state of acceptance, determining the initial temperature through acceptance rate versus temperature plots, representing the solution sequence as a Markov chain, and evaluating the algorithm using benchmark functions. By achieving these objectives, this report aims to provide a comprehensive understanding of the simulated annealing algorithm's functionality and its application to various optimization problems.

2.1 Boltzmann distribution

The Boltzmann distribution is a fundamental concept describing the distribution of particles over various energy states in a system at thermal equilibrium. It provides the probability of a system being in a state with a specific energy E at a given temperature T. In the context of simulated annealing, the Boltzmann distribution is used to probabilistically accept worse solutions during the optimization process. The probability function is given by the following equation

$$f(c) = 4\pi c^2 * \left(\frac{m}{2\pi KT}\right)^{\frac{3}{2}} * \exp\left(\frac{-mc^2}{2KT}\right) \qquad (1)$$

Where f(c) - The probability density function of the speed c particles

m – Mass of the molecule

T – Temperature

K – Boltzmann constant

c – Speed of molecules

By allowing the algorithm to occasionally accept higher-energy (worse) states, the Boltzmann distribution helps the system to escape local minima and explore the solution space more effectively, enhancing the chances of finding the global minimum.

2.2 Pseudo code and dry run process

Initially, we have introduced a simple pseudo code to developed a simulated annealing algorithm involves laying out the foundational steps as below.

```
1. Initialize:
    • Set initial_state = State_i
    • Set initial_temperature = T
    • Define stopping_criterion
2. Repeat until stopping_criterion is met:
        a. Perturbation: Generate new_state = State_j from State_i
        b. Calculate Energy Change (ΔE):
                ΔE = Energy(State_j) - Energy(State_i)
        c. Acceptance Criteria:
                i. If ΔE <= 0:
                        Accept new_state as State_j
                ii. Else:
                        If exp($\frac{-\Delta E}{T}$) > random[0, 1):
                        Accept new_state as State_j
        d. If accepted, update State_i to State_j
3. Update Temperature:
        T = decrease_temperature(T)
4. Check Stopping Criterion:
    If stopping_criterion is met, exit loop
5. Return final_state = State_i as the best solution found
```

After implementing the above pseudo code, we use $y = x^2$ as the energy function where x ranges between [-1, 1], to test its functionality. The algorithm is expected to converge to the global minimum point at y = 0.

2.3 Cooling rate and different cooling schedules

In simulated annealing, the cooling rate refers to the rate at which the temperature T is decreased during the optimization process. This rate affects how quickly or slowly the algorithm transitions from exploring a wide range of solutions (higher temperatures) to focusing on exploiting the best solutions found so far (lower temperatures). Typically, the cooling rate is denoted by a parameter α and it determines the rate of decrease of the temperature T according to the cooling schedule. In our research we used two cooling schedules such as;

- Exponential cooling schedule: $T_{new} = \mathrm{T_{old}} \times \alpha$ where α is a constant between 0 and 1
- Linear cooling schedule: $T_{new} = \mathrm{T_{old}} - \Delta T$ at L trials with a fixed ΔT

The choice of cooling rate depends on the problem at hand and the desired balance between exploration (at higher temperatures) and exploitation (at lower temperatures). Too rapid cooling may lead to premature convergence to suboptimal solutions, while overly slow cooling may result in excessive computational time. Adjusting the cooling rate is often part of fine-tuning the simulated annealing algorithm to achieve optimal performance for specific optimization tasks.

2.4 Acceptance Rate versus Temperature

To determine the initial temperature T_0 in simulated annealing, we can analyze the acceptance rate versus temperature plot. Here's how we can interpret this graph:

1. Understanding the Acceptance Rate:
   - The acceptance rate at a given temperature T indicates the probability of accepting a worse solution compared to the current one.
   - At higher temperatures, the algorithm is more likely to accept worse solutions, allowing for exploration of the solution space.
   - As the temperature decreases, the acceptance rate typically decreases as well, focusing the algorithm on exploitation and convergence towards the global minimum.

2. Finding the Initial Temperature T_0:
   - Plot the acceptance rate against a range of temperatures T starting from an initial guess or a wide range (e.g., logarithmically spaced).
   - Identify the temperature T_0 where the acceptance rate is relatively high, typically aiming for around 80-90%.
   - This temperature T_0 ensures that the algorithm starts with sufficient flexibility to explore different solutions effectively without getting stuck in local minima early in the optimization process.

3. Practical Steps:
   - Implement the simulated annealing algorithm with an initial guess for T_0

- Perform multiple runs of the algorithm, adjusting T_0 based on the acceptance rate observed in each run.
- Refine T_0 until you find a temperature that balances exploration and exploitation effectively, leading to improved convergence towards the global minimum of the objective function $y = x^2$ in our case.

By using this approach, we can effectively determine the initial temperature T_0 in simulated annealing, optimizing the algorithm's performance for finding the global minimum of the objective function within the specified range of x.

2.5 Markov chain Process

In our project, we have implemented simulated annealing with a fixed Markov chain length. This approach allows for a structured exploration of the solution space, balancing between random state transitions for exploration and focusing on more promising solutions for exploitation. By maintaining a fixed Markov chain length, we ensure a consistent and controlled exploration process, which can contribute to more reliable and efficient optimization outcomes across various complex problem domains. So, it is better to modify our algorithm using Markov chain process. The steps of modified algorithm is given below.

Initialization:

- Start with an initial solution i, set the iteration counter k= 0
- Initialize the cooling schedule parameter C_k and the iteration limit L_k.

Main Loop:

- **Repeat** until a stopping condition is met:
  - For i = 0 to L_k:
    - Generate a new solution j uniformly from the solution space S.
    - Evaluate:
      - If f(i) > f(j):
        - Accept j as the new current solution i.
      - Else:
        - Calculate the acceptance probability based on the Boltzmann factor *as* $\exp(\frac{f(i)-f(j)}{C_k})$.
        - Accept j as i if this probability exceeds a uniform random number from [0, 1].
  - Increment k = k+1.
  - Update C_k and L_k
- **Stopping Condition:**
  1. Define a criterion to terminate the algorithm, such as reaching a maximum number of iterations or achieving a convergence threshold.

2.6 Bench Mark Functions

We choose benchmark functions to test our algorithm. Main purpose of using the test function is to evaluate the behavior and performance of the algorithm. For this purpose, the researchers compiled a rich set of benchmark functions for unconstrained optimization problems with diverse properties in terms of modality, separability, and valley landscape.

Test of reliability, efficiency and validation of optimization algorithms is frequently carried out by using a chosen set of common standard benchmarks or test functions.

2.6.1 Types of bench mark functions

A benchmark function is a standard test function used to evaluate and compare the performance of optimization algorithms

1. Unimodal Functions:

   - Have a single global optimum

     Example: Sphere Function

2. Multimodal Functions:

   - Have multiple local optima

     Example: Rastrigin Function

3. Composite Functions:

   - Combine multiple functions to create complex landscapes

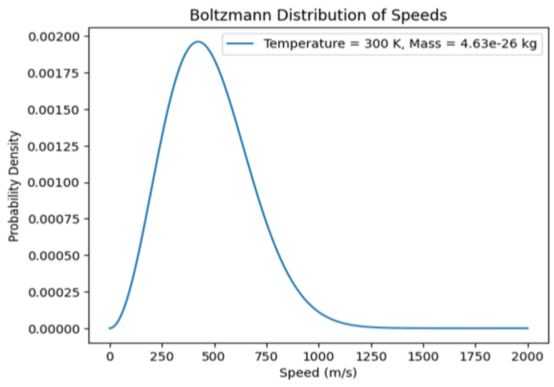
     Example: Schwefel Function

2.6.1 Advantages of choosing bench mark functions

- Standardization: Benchmark problems provide a standardized set of test cases. Researchers can compare different optimization algorithms fairly using the same benchmarks.
- Complexity: Normal functions may not adequately represent real - world optimization challenges. Benchmarks mimic complex landscapes with multiple local optima, allowing evaluation under realistic conditions.
- Known Optima: Benchmarks have known global optima, making it easier to assess an algorithm's convergence. Real-world problems rarely have known optimal solutions.
- Diversity: Benchmarks cover various problem types (unimodal, multimodal, noisy). This diversity helps evaluate algorithm robustness across scenarios.

Results and Discussions

Under this topic we have analyzed the outcomes of our implemented simulated annealing algorithm with a fixed Markov chain length and this section critically examines the effectiveness of our approach, comparing achieved results with theoretical expectations or benchmark solutions where applicable.



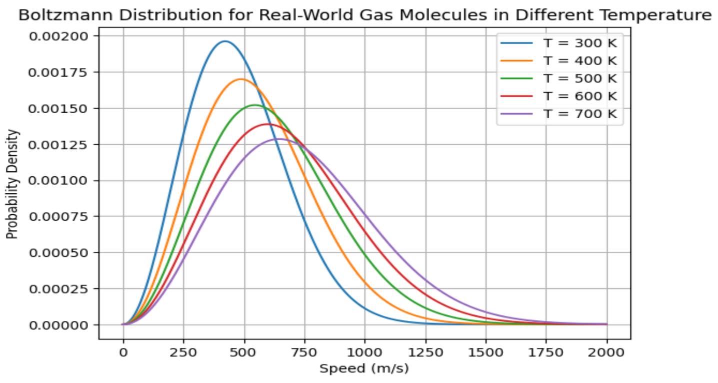


Figure 1: Boltzmann Distribution

Figure 2: Boltzmann Distribution with different temperatures

The Maxwell-Boltzmann distribution is a useful tool for understanding the relationship between temperature and the speed of gas molecules. The graph shows that gas molecules have a range of speeds, and that the distribution of speeds is dependent on temperature. In simulated annealing:

1. Initial Exploration: At high temperatures, the probability of accepting worse solutions is higher, enabling exploration of a broader search space.

2. Gradual Cooling: As the temperature decreases, the acceptance probability for worse solutions diminishes, focusing the search on local improvements.

3. Convergence: Ultimately, the temperature approaches zero, and the algorithm converges to a solution, ideally the global minimum.

The narrowing and left-shifting of the Boltzmann distribution at lower temperatures illustrate how the probability of accepting higher-energy (worse) states decreases, guiding the algorithm toward optimal solutions.

Based on this concept we can find fixed Markov chain length with given temperature. In our analysis we use 1500K as the initial temperature.

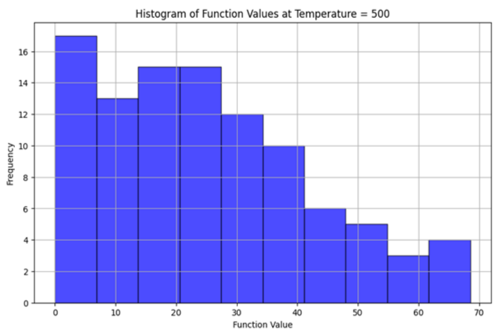


Figure 3

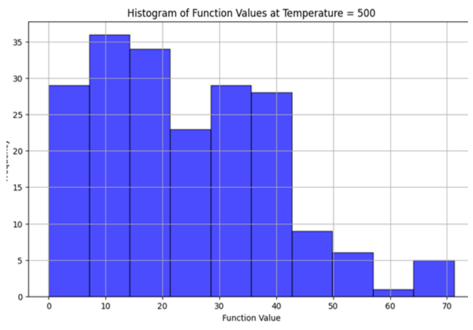


Figure 4

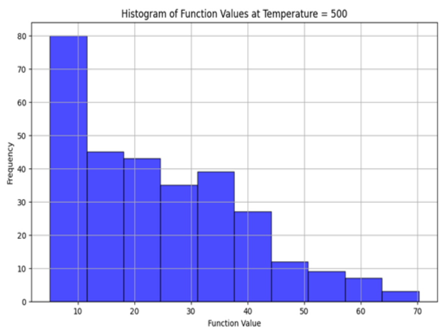


Figure 5

From the above figures 3, 4 and 5 we are selecting 300 as length of fixed Markov chain length because it depict the Boltzmann distribution curve mentioned in figure1.

Then we need to find the suitable initial temperature in order to get the possible global optimum solution. For that, we have use acceptance rate versus temperature graph.

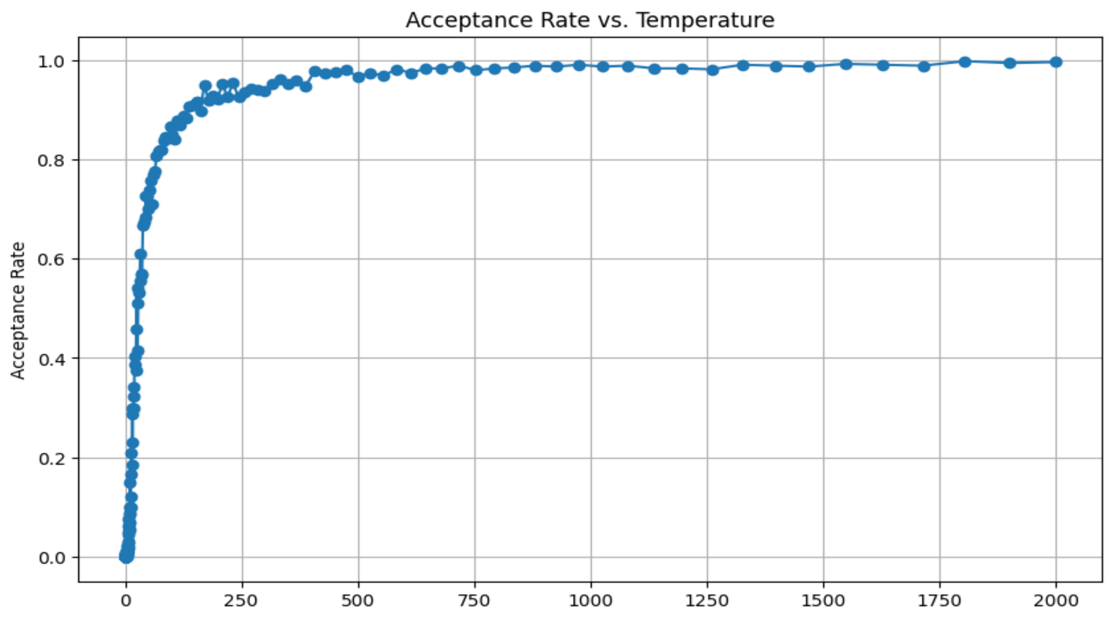
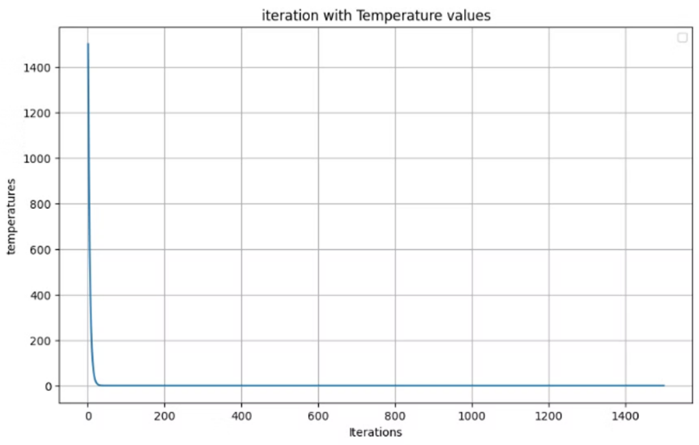


Figure 1: Variation of Acceptance rate versus temperature

As mentioned above in the methodology, we can see that around 200K the acceptance rate is high and between the acceptable ranges of 80-90%.

It is important to select suitable cooling schedules.



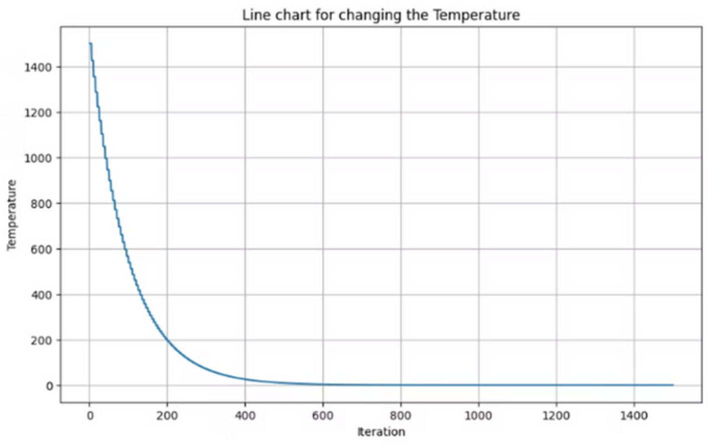


Figure 7: Temperature variation of exponential cooling schedule

Figure 8: Temperature variation of linear cooling schedule

As we know too rapid decrease of temperature may lead to sub optimal solution. So, we have to maintain slow cooling rate. Figure 7 illustrates the variation of temperature with iteration of exponential cooling which shows rapid decrease of temperature. After nearly 20 iterations temperature approaches to zero while figure 8 shows the variation of temperature with iteration of linear cooling schedule in which temperature decrease takes place after every 5 iterations from initial temperature. We can clearly see that approximately after 600 iterations temperature becomes nearly zero and temperature decrease takes place very slowly.

We have used a two variable function given as below to test our modified algorithm and we get desired output according to our modified code.

$$f(x,y) = (x-3)^2 + (y+2)^2 + 5 \tag{3}$$

Where x, y between the range of [-3, 4].

Global Optimum Point: x = 3.0101010101010095, y = -2.0101010101010104,

z = 5.0002040608101215

To get more clear idea we can plot above two graphs labeled as figure 7 and 8 with different iterations by keeping initial temperature as constant and calculate its error from the global optimum solution. Error can be obtained using the equation 2 as below.

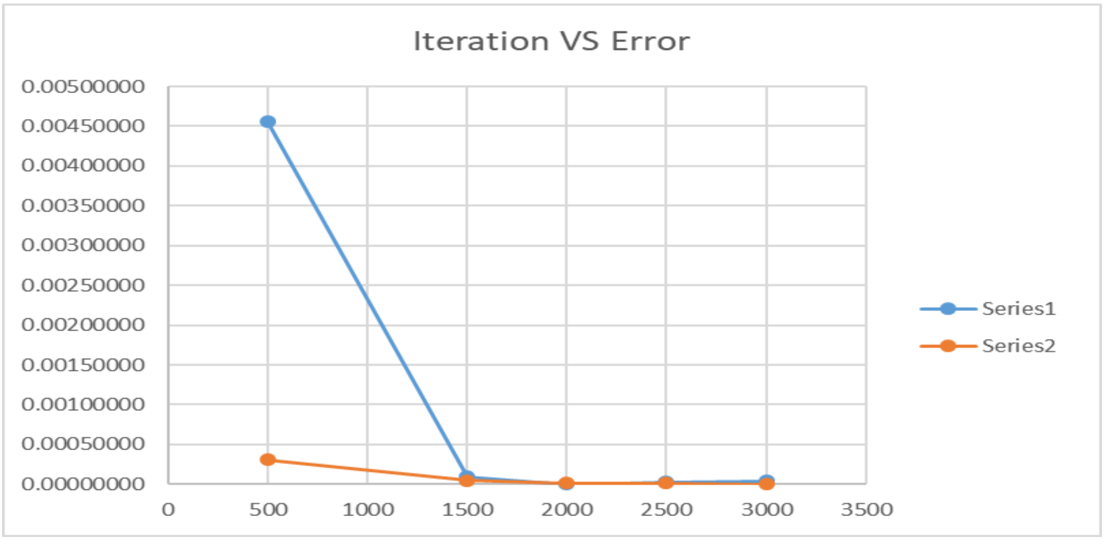
$$error = (f(x) - f^*(x))^2 \tag{2}$$

Table 1: Comparison of error

Iterations	Exponential Cooling(α = 0.95)	Linear cooling	Error1	Error2
500	5.067757399	5.01775322	0.00456345	0.00030797
1500	5.010063789	5.007227785	0.00009721	0.00004933
2000	5.000618402	5.003690734	0.00000017	0.00001216

When we increase the number of iterations, error decreases. Among the above two cooling schedules we select linear cooling schedule by analyzing the error.

Figure 9: Graphical representation of error



This is the graphical representation of error for two cooling schedules where series 1 indicates exponential cooling and series 2 represents linear cooling schedules. Based on this graph error of linear cooling schedule with iteration is less than that of other. So, we select linear cooling schedule.

4.1 Results of test functions

In order to evaluate the performance and efficacy of the simulated annealing algorithm, we conducted a series of tests using standard benchmark functions. These functions are commonly employed in optimization studies due to their well-defined properties and the challenges they present. The results obtained from these tests provide valuable insights into the algorithm's ability to find optimal solutions and its robustness across different types of problem landscapes. The following sections detail the outcomes of these experiments, highlighting both the strengths and limitations observed during the testing process.

Table 2: Results of benchmark test functions

Function Name	Formula	Global optimum solution	Optimum solution using code	Error
Ackley 2 function (Uni modal)	$f(x_1, x_2) = -200e^{-0.02\sqrt{{x_1}^2+{x_2}^2}}$ Subject to -32 ≤ x_i ≤ 32	$x^* = (0, 0)$ $f(x^*) = -200$	X = (0.000677885415, -0.020838515811) f(x) = -199.91662	0.00695

Adjimann function (Multi modal)	$f(x_1, x_2) = \cos(x_1)\sin(x_2) - \frac{x_1}{(x_2^2+1)}$ Subject to $-1 \leq x_1 \leq 2$, $-1 \leq x_2 \leq 1$	$x^* = (2, 0.10578)$ $f(x^*) = -2.02181$	X = (1.999689132690, 0.110514870852) f(x) = -2.02143	0.00000014
Himmelblau function (Multi modal)	$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$ Subject to $-5 \leq x_i \leq 5$	$x^* = (-2, 3)$ $f(x^*) = 0$	X = (-2.804720783304, 3.132321120159) f(x) = 4.65777e-05	2.16948e-09
Camel function – three hump (Multi modal)	$f(x_1, x_2) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$ Subject to $-5 \leq x_i \leq 5$	$x^* = (0, 0)$ $f(x^*) = 0$	X = (0.003687104068, -0.000780339233) f(x) = 2.49210e-05	6.21056e-10
Chichinadze function (Multi modal)	$f(x_1, x_2) = x_1^2 - 12x_1 + 11 + 10\cos\left(\frac{\pi x_1}{2}\right) + 8\sin\left(\frac{5\pi x_1}{2}\right) - \left(\frac{1}{5}\right)^{0.5}\exp(-0.5(x_2 - 0.5)^2)$ Subject to $-30 \leq x_i \leq 30$	$x^* = (5.90133, 0.5)$ $f(x^*) = -43.3159$	X = (6.189963719556, 0.502262107913) f(x) = -42.94438	0.138027
Beale function (Uni modal)	$f(x_1, x_2) = (1.5 - x_1 + x_2)^2 + (2.25 - x_1 + x_1{x_2}^2)^2 + (2.625 - x_1 + x_1{x_2}^3)^2$ Subject to $-4.5 \leq x_i \leq 4.5$	$x^* = (3, 0.5)$ $f(x^*) = 0$	X = (3.023405588903, 0.504625937275) f(x) = 0.00012	1.44e-08

Egg Crate function (Multi modal)	$f(x_1, x_2) = x_1^2 + x_2^2 + 21(sin^2(x_1) + sin^2(x_2))$ Subject to -5 ≤ x_i ≤ 5	x* = (0, 0) f(x*) = 0	X = (0.002699084747, -0.000105668215) f(x) = 0.00019	3.61e-08
Deckkers-Aarts function (Multi modal)	$f(x_1, x_2) = 10^5 x_1^2 + x_2^2 - (x_1^2 + x_2^2)^2 + 10^{-5}(x_1^2 + x_2^2)^2$ Subject to -20 ≤ x_i ≤ 20	x* = (0, ±15) f(x*) = -24777	X = (0.005306801671, 14.947403060997) f(x) = -24773.69279	10.93764
Schwefel 2.36 function (Multi modal)	$f(x_1, x_2) = -x_1 x_2 (72 - 2x_1 - 2x_2)$ Subject to 0 ≤ x_i ≤ 500	x* = (12,….,12) f(x*) = -3456	X = (12.007832576409, 12.026455515647) f(x) = -3455.97674	0.00054
Rotated Ellipse function (Uni modal)	$f(x_1, x_2) = 7x_1^2 - 6\sqrt{3}x_1 x_2 + 13x_2^2$ Subject to -500 ≤ x_i ≤ 500	x* = (0, 0) f(x*) = 0	X = (0.174063842941, 0.369747828424) f(x) = 1.32052	1.74377

The above table 2 presents the results of benchmark test functions, detailing the performance of an optimization algorithm on various functions categorized by their properties (e.g. uni modal and multi modal). Overall, the table illustrates that the algorithm performs well in finding near – optimal solutions for various benchmarks functions, with relatively small error in most cases.

Conclusion

- The simulated annealing process developed for this successfully can be used in optimizing complex functions giving better solutions.
- Inserting the Markov Chain within the algorithm significantly enhanced the algorithm and better solutions, leading to a higher likelihood of achieving the global optimum.

- Multiple cooling schedules, including exponential and linear, were tested and compared, with the exponential schedule proving to be more efficient in reducing the error over iterations.
- The application of benchmark functions, both unimodal and multimodal, provided validation for the simulated annealing algorithm.
- Adjusting the initial temperature and cooling rate parameters demonstrated the sensitivity of the algorithm to these factors, highlighting the importance of parameter tuning for optimal performance.
- The final algorithm showed consistent convergence towards the global minimum, with improvements in both the accuracy and efficiency when compared to initial versions.
- The histogram analysis of function values confirmed the expected Boltzmann distribution, further proving outcome of the simulated annealing process.
- Overall, the project demonstrated the practical applicability of simulated annealing in solving complex optimization problems, laying a foundation for future enhancements.

References

Busetti, F. (2001). Simulated annealing overview. ResearchGate. https://www.researchgate.net/publication/238690391_Simulated_annealing_overview.

Momin Jamil and Xin-She Yang, A literature survey of benchmark functions for global optimization problems, Int. Journal of Mathematical Modelling and Numerical Optimisation, Vol. 4, No. 2, pp. 150–194 (2013). DOI: 10.1504/IJMMNO.2013.055204

Henderson, D., Jacobson, S. H., & Johnson, A. W. (2006). The theory and practice of simulated annealing. In Kluwer Academic Publishers eBooks (pp. 287–319). https://doi.org/10.1007/0-306-48056-5_10

Wang, Z.; Wang, Y.; Xu, H.; Xie, H. Application of Simulated Annealing Algorithm in Core Flow Distribution Optimization. Energies 2022, 15, 8242. https://doi.org/10.3390/en15218242