MT 409: Selected Topics in Applied Operations Research

Academic Year: 2022/2023

Group 05 – Final Project Report

DEVELOPMENT & VALIDATION OF SIMULATED ANNEALING ALGORITHM

by

S/18/813 – Tharindu Kariyawasam

S/18/814 – Nipuni Kaushalya

S/18/819 – Sanduni Ranasinghe

S/18/822 – Ranali Piyatilake

S/18/826 – Niroshan Sampath

S/18/827 – Kumudumali Sandaleka

S/18/836 – Chamod Wijesinghe

Introduction

The Simulated Annealing (SA) algorithm is an optimization technique inspired by the annealing process in metallurgy, effective for solving complex problems. The algorithm was introduced in the combinatorial optimization area by (S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi, 1983). The SA algorithm is designed around the Metropolis Criterion, which probabilistically accepts worse solutions based on the current temperature, facilitating the escape from local optima. The choices of the initial temperature, the number of iterations before reducing the temperature, and the temperature reduction factor play important roles in the successful convergence of the algorithm (Daniel Delahaye, Supatcha Chaimatanan, Marcel Mongeau, 2019).

The primary objective of this study is to develop and validate a Simulated Annealing (SA) algorithm capable of effectively solving complex optimization problems. This involves creating a robust SA algorithm with crucial components such as the Metropolis Criterion and different cooling schedules and evaluating its overall performance using unimodal and multimodal benchmark functions to assess its efficiency and effectiveness (Momin Jamil, Xin-She Yang,, 2013).

In this study, the key parameters such as the cooling schedule and Markov chain length are systematically varied to analyze their influence on the algorithm's effectiveness. Hence, the study compares constant versus dynamic Markov chain lengths to identify the most effective configurations. Moreover, two cooling schedules (linear and exponential) are explored to understand their impact on temperature reduction and convergence. This comprehensive framework thoroughly evaluates the SA algorithm's capabilities and impact on the solution quality, providing insights into optimal parameter settings and practical applicability in solving diverse and challenging optimization problems.

The main results of this study demonstrate that the dynamically adjusted Markov chain length approach provides more efficient use of computational resources, reducing overall computational time while maintaining solution quality. Moreover, the linear cooling schedule performed better in terms of expected error for both unimodal and multimodal functions, though with a higher computational cost. This reflects the importance of selecting appropriate parameters to achieve a tradeoff between computational efficiency and the accuracy of the final solution in Simulated Annealing algorithms.

Materials and Methods

The methodology of this study involves several key steps to develop and evaluate the SA algorithm based on the Metropolis Criterion.

1. Algorithm Development

The SA algorithm was developed using the Metropolis Criterion, which controls the acceptance of new states based on their energy difference and the current temperature. The pseudocode for the Metropolis Criterion is as follows:

```
1. Repeat
    1.i.   Perturbation: State $i$ to state $j$
    1.ii.  If $\Delta E_{ij} \leq 0$ then accept state $j$
           Else if exp $(-\frac{\Delta E_{ij}}{c})$ > random [0, 1) then accept
    1.iii. If accept, update the state $i$ with state $j$
2. Until equilibrium is approached
```

Here, c represents the temperature.

This developed Metropolis criterion was applied to the energy function [1].

$$f(x) = x^2 \tag{1}$$

To explore the search space, the initial temperature was set high, and gradually reduced. The acceptance of states becomes more stable as the initial temperature decreases. The Metropolis criterion, integrated with decreasing temperatures, ensured convergence toward the expected outcome, where the minimum value of function [*1*] is zero, with accepted states (x) approaching zero. Thus, a linear cooling schedule [2], which gradually decrease of the temperature by 50% per iteration, was used to facilitate this process.

$$T_{current} = 0.5 \times T_{current} \tag{2}$$

Then, a dual variable function [3] is chosen and tested to observe the algorithm's performance.

$$f(x_1, x_2) = 3x_1^2 + 5x_2^2 + 2 \tag{3}$$

subject to $-1 \ \leq x \leq \ 1$. The global minimum is located at $x^* = (0, 0),\ f(x^*) = 2$.

The following findings were obtained during the testing of the dual variable function [*3*].

$$\begin{aligned} &Optimal\ Solution\ = \ 2.006343190440746 \\ &Minimum\ points\ = \ (0.0445314946131079, -0.008877256370233066) \\ &Error\ = \ 4.023606496757206\ \times\ 10^{-5} \end{aligned}$$

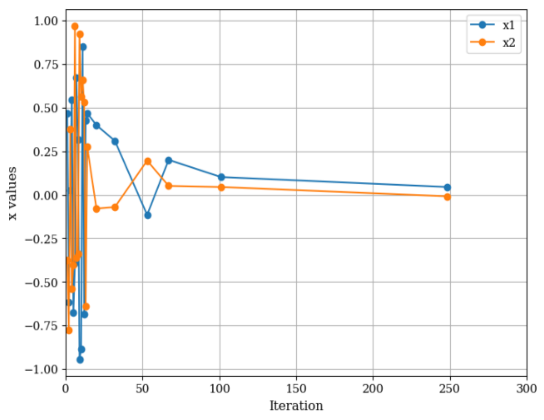


Figure 1 : x values vs Iteration plot for function 3

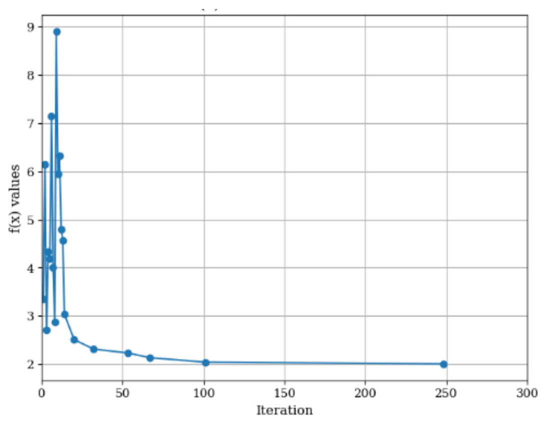


Figure 2 : f(x) values vs Iteration plot for function 3

Figure 1 depicts that x_1 and x_2 values are converging to its corresponding minimum values and Figure 2 depicts the function value converging to its desired minimum value of 2.

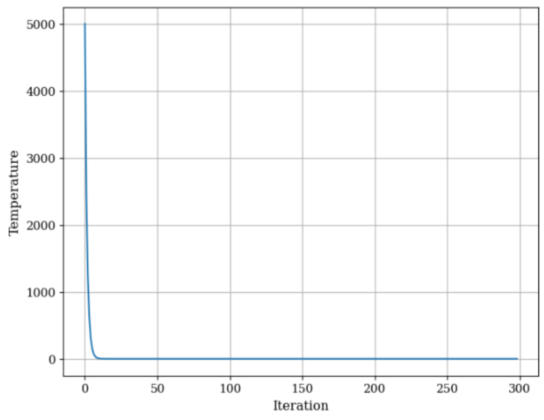


Figure 3 : Temperature vs Iteration Plot

Furthermore, it was noticed that (Figure 3) halving the temperature at each step results in a significant and rapid decrease in temperature. This sudden drop can cause the algorithm to quickly lose its ability to escape local optima, as the probability of accepting worse solutions decreases exponentially with lower temperatures. To address this sudden drop in temperature revealed by the linear cooling schedule, an exponential cooling schedule [4] was adopted.

$$T_{new} = T_{initial} . e^{-0.05i} \qquad 4$$

where i and T represent iteration number and the temperature.

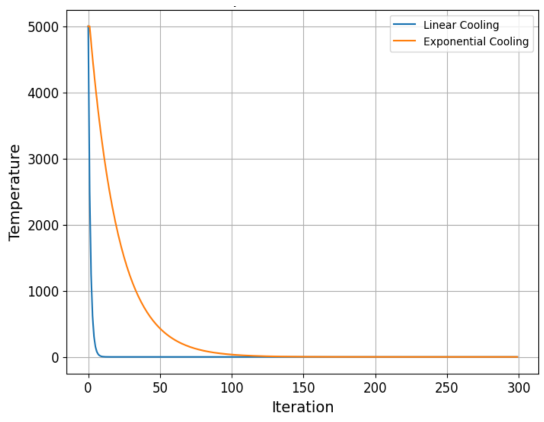


Figure 4 : Temperature vs Iteration Plot Comparison

As shown in Figure 5 (below), the probability of accepting worse solutions decreases with lower temperatures which limits the algorithm's ability to explore the solution space more freely.

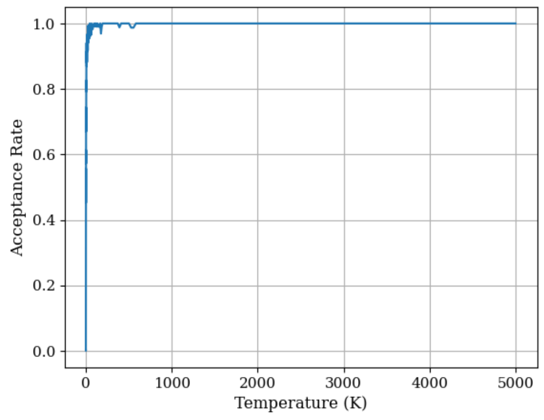


Figure 5 : Variation of Acceptance Rate Against Temperature

Therefore, to expand the search space and reach equilibrium, a dynamically adjusted Markov chain length (L_k) according to equation [5] was introduced by adding another loop to the current algorithm.

$$L_k = L_0 \times 1.01^{\{-\frac{1}{0.05}ln(\frac{T_{new}}{T_{initial}})\}} \qquad 5$$

where L_{k} and L_0 is the current and initial Markov chain length.

As a result, the algorithm was modified to have a dynamic Markov chain length (at each iteration) with an exponential cooling schedule.

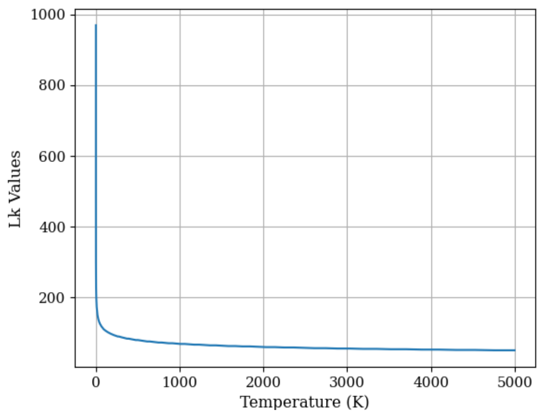


Figure 6 : Temperature vs L_k Values Plot

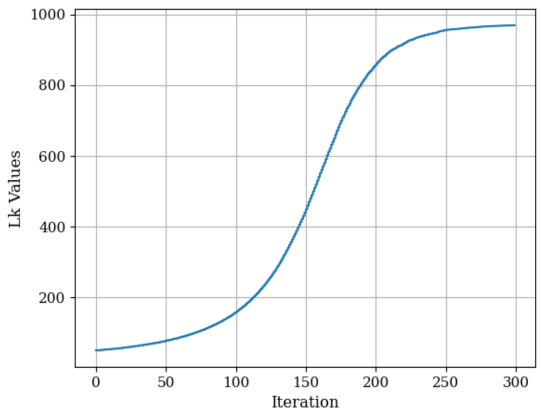
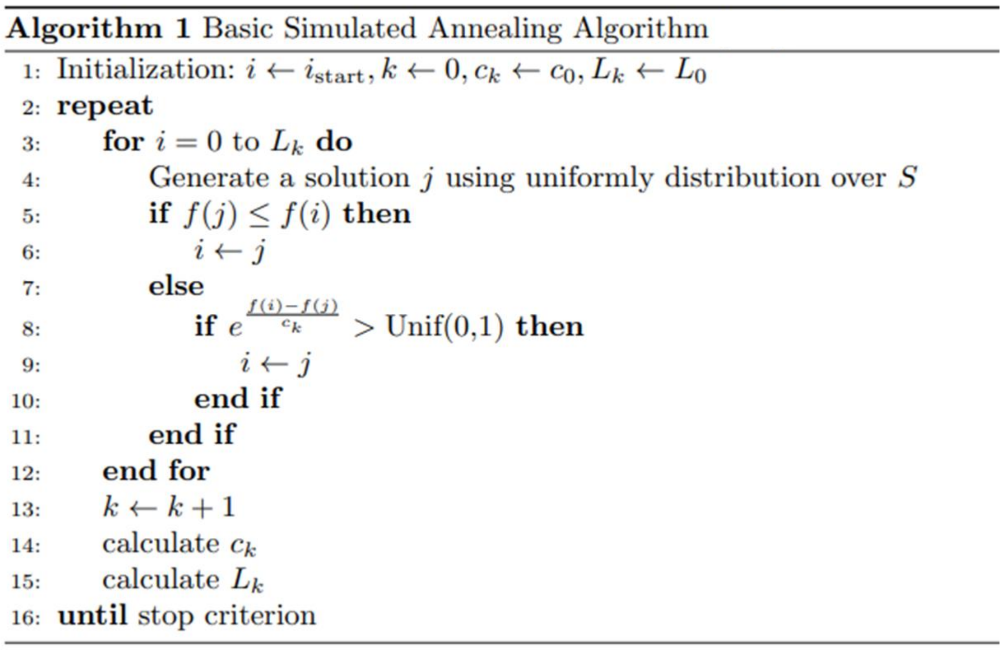


Figure 7 : Iteration vs L_k Values Plot

The relationship between L_k values and temperature is shown in Figure 6. It is observed that L_k values are initially high at lower temperatures and then decrease rapidly, eventually stabilizing at a lower value as the temperature increases. This behavior suggests that at lower temperatures, the algorithm allows for longer Markov chains to thoroughly explore the local minima.

The variation of L_k values with respect to the number of iterations is displayed in Figure 7. It is observed that the L_k values increase gradually at first and then more rapidly, reaching a plateau as the iterations progress. This indicates an adaptive mechanism where the Markov chain length grows as the algorithm progresses, allowing for a more extensive search of the solution space in later iterations.



Algorithm 1 Basic Simulated Annealing Algorithm

```
1: Initialization: $i \leftarrow i_{\text{start}}, k \leftarrow 0, c_k \leftarrow c_0, L_k \leftarrow L_0$
2: repeat
3:     for $i = 0$ to $L_k$ do
4:         Generate a solution $j$ using uniformly distribution over $S$
5:         if $f(j) \leq f(i)$ then
6:             $i \leftarrow j$
7:         else
8:             if $e^{\frac{f(i)-f(j)}{c_k}} >$ Unif(0,1) then
9:                 $i \leftarrow j$
10:            end if
11:        end if
12:    end for
13:    $k \leftarrow k + 1$
14:    calculate $c_k$
15:    calculate $L_k$
16: until stop criterion
```

Figure 8 : Basic Simulated Annealing Algorithm

2. <u>Validation Using Benchmark Functions</u>

Benchmark functions are crucial for evaluating the effectiveness of an optimization algorithm, providing a standardized set of problems to test and compare the algorithm performance. In this study, the effectiveness of the SA algorithm was evaluated using a unimodal and a multimodal benchmark function, named the Ackley 2 Function [6] and Price 2 Function [7]. (Momin Jamil, Xin-She Yang,, 2013)

- *<u>Unimodal Benchmark Function</u>*

Ackley 2 Function: Continuous, Differentiable, Non – Separable, Non – Scalable, Unimodal

$$f_2(x) = -200e^{-0.02\sqrt{x_1^2+x_2^2}} \tag{6}$$

subject to $-32 \leq x \leq 32$. The global minimum is located at $x^* = (0,0),\ f(x^*) = -200$.

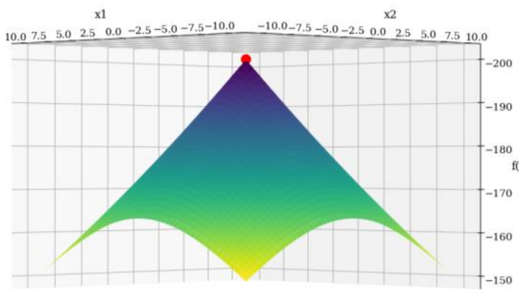


Figure 9 : Surface Plot of the Function 6

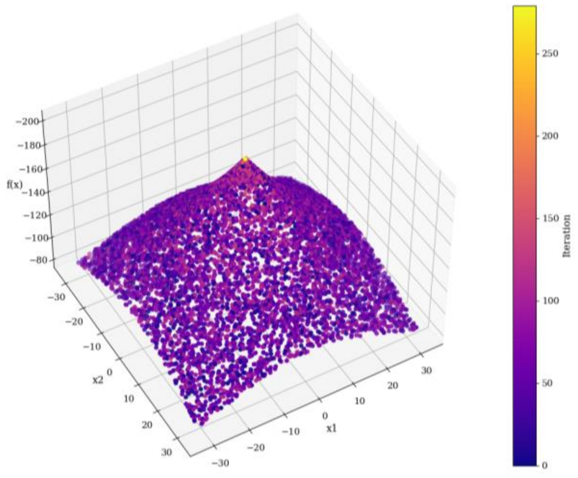


Figure 10 : 3D Scatterplot for Iteration vs f(x) values of the Function 6

$$\begin{aligned} Optimal\ Solution &= -199.76463901919473 \\ Minimum\ points &= (0.05807388240907585, 9.679705542484067 \times 10^{-3}) \\ Error &= 0.05539479128561779 \end{aligned}$$

The above results were observed when testing for the unimodal function [6]. As the number of iterations increased, the function values converged towards the optimal value, reinforcing the algorithm's ability to perform well on unimodal functions. This behavior is illustrated in Figure 10 where the function values become increasingly concentrated around the known minimum as iteration progressed.

- *Multimodal Benchmark Function*

Price 2 Function: Continuous, Differentiable, Non – Separable, Non – Scalable, Multimodal

$$f_{95}(x) = 1 + sin^2\, x_1 + sin^2 x_2 - 0.1e^{-x_1^2 - x_2^2} \qquad 7$$

subject to $-10 \leq x \leq 10$. The global minimum is located at $x^* = f(0,0),\ f(x^*) = 0.9$.

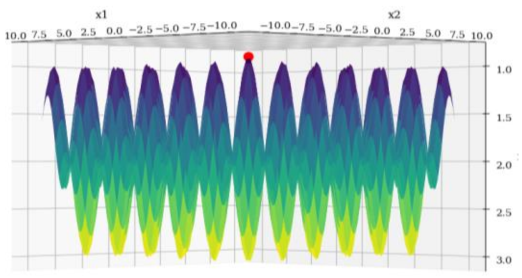


Figure 11 : Surface Plot of the Function 7

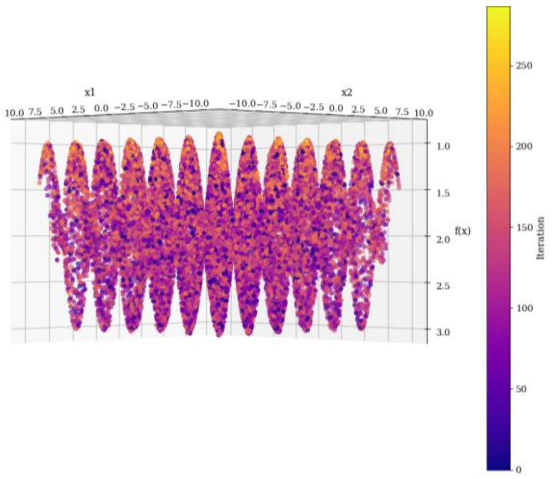


Figure 12 : 3D Scatterplot for Iteration vs f(x) values of the Function 7

$$\begin{aligned} Optimal\ Solution &= 0.9062679636672892 \\ Minimum\ points &= (-0.059569404430890316, 0.04643634268838248) \\ Error &= 3.928736853445762 \times 10^{-5} \end{aligned}$$

The above results were observed when testing for the multimodal function [7]. Similar to the unimodal case, the Figure 12 illustrates that as iterations progressed, the function values increasingly converged to the optimal value. The scattered points progressively clustered around the global minimum, demonstrating the algorithm's capacity to navigate multimodal surfaces effectively.

3. Analysis

To further understand the impact of the key parameters on the performance of the SA algorithm, Markov chain length (L_k) and the cooling schedule were tuned to assess the performance of SA.

Initially, the performance of the algorithm was observed for a fixed Markov chain length which remained constant throughout the iterations against a dynamic L_k value varying according to the equation [5]. Subsequently, the effectiveness of linear and exponential cooling schedules was observed for dynamic L_k scenario.

Furthermore, the expected error and the computation time over 500 runs were recorded for each case to evaluate the performance of the algorithm providing insights into its convergence behavior and solution quality.

Results and Discussion

The following section discuss the observations got for the above mentioned 2 cases.

1. Dynamic L_k vs Constant L_k (for Unimodal and Multimodal function)

Exponential Cooling Schedule [4]				
	Multimodal Benchmark [7]		Unimodal Benchmark [6]	
For 500 runs	Expected Value for Sq. Error	Computation Time (s)	Expected Value for Sq. Error	Computation Time (s)
Dynamic L_k [50 - 989] [5]	4.82×10^{-5}	1.23185	0.29914	0.99263
Constant L_k at 989	3.93×10^{-5}	3.96441	0.18722	3.12878

Table 1 : Performance of Dynamic and Constant Markov Chain Length on Multimodal and Unimodal Benchmarks

The error values for both algorithms are relatively close, indicating that both methods effectively find solutions. The dynamic L_k algorithm has a slightly higher error for both unimodal and multimodal functions compared to the constant L_k algorithm. Moreover, the computational time for the algorithm with a constant L_k is significantly higher than that for the dynamic L_k for both unimodal and multimodal functions (Figure 13 and Figure 14).

The constant L_k algorithm consistently runs the inner loop 989 times, regardless of the temperature or the state of convergence. By running the inner loop a constant number of times, the algorithm ensures a thorough exploration of the solution space at each temperature step. This can result in redundant evaluations when the temperature is high and excessive fine-tuning when the temperature is low, leading to increased computational time without proportional gains in accuracy.

In contrast, the dynamic L_k algorithm starts with a lower value (50) and gradually increases it (to 989), with fewer evaluations at higher temperatures and more as the temperature decreases. This ensures that computational resources are used efficiently, with more effort dedicated to fine-tuning near the optimum, thus reducing overall computational time while maintaining solution quality.

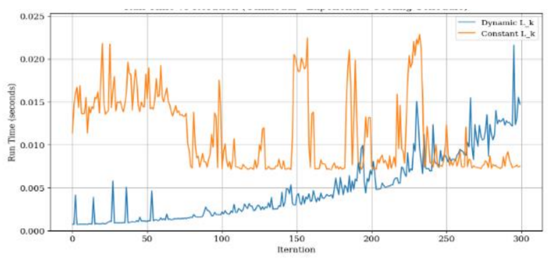


Figure 13 : Run Time vs Iteration (Unimodal - 6) Plot Comparison

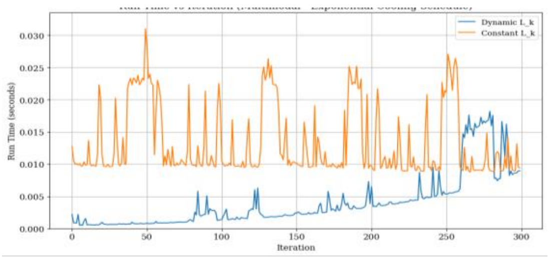


Figure 14 : Run Time vs Iteration (Multimodal - 7) Plot Comparison

2. Linear vs Exponential Cooling Schedule (for Multimodal function & Unimodal function)

Dynamic L_k $[50 - 989]$ [5]				
	Multimodal Benchmark [7]		Unimodal Benchmark [6]	
For 500 runs	Expected Value for Sq. Error	Computation Time (s)	Expected Value for Sq. Error	Computation Time (s)
Exponential [4]	4.82×10^{-5}	1.23185	0.29914	0.99263
Linear [3]	8.78×10^{-7}	2.79462	0.08717	2.31305

Table 2 : Performance of the Exponential and Linear Cooling Schedule on Multimodal and Unimodal Benchmarks

The linear cooling schedule consistently resulted in a lower expected error compared to the exponential cooling schedule for both benchmark functions. Moreover, the linear cooling schedule had a significantly longer computational time compared to the exponential cooling schedule.

The observed differences in performance can be attributed to the exploration and exploitation dynamics of the two cooling schedules. The linear cooling schedule rapidly decreases the temperature, allowing the algorithm to converge quickly to a near-global optimal value. This rapid convergence leads to a shorter exploration phase and a longer exploitation phase, which fine-tunes the solution to a more accurate state. In contrast, the exponential cooling schedule maintains a more gradual temperature decrease, resulting in a more extended exploration phase. This approach allows the algorithm to explore the search space more thoroughly but leaves less time for the exploitation phase, resulting in less accurate final solutions.

The longer exploitation phase of the linear cooling schedule enables it to refine the solution more effectively, leading to lower errors despite the higher computational cost whereas the exponential schedule is computationally less expensive as it does not allocate as much time to fine-tuning the solution, resulting in higher errors.

Conclusion

In conclusion, the evaluation of dynamic versus constant Markov chain length (L_k) demonstrated that dynamically adjusted L_k approach provide a more efficient use of computational resources, significantly reducing overall computational time while maintaining solution quality. For cooling schedules, the linear cooling schedule outperformed the exponential cooling schedule in terms of expected error for both unimodal and multimodal functions, although with a higher computational cost. These findings emphasize the importance of selecting appropriate parameters and cooling schedules to balance computational efficiency and solution accuracy in Simulated Annealing algorithms.

References

- Daniel Delahaye, Supatcha Chaimatanan, Marcel Mongeau. (2019). Simulated Annealing: From Basics to Applications. In *Handbook of metaheuristics* (pp. 1 - 35).
- Jernej Olensek, Tadej Tuma, Janez Puhan, Árpád Burmen. (2011). A new asynchronous parallel global optimization method based on simulated annealing and differential evolution. *ELSEVIER*, 1481 - 1489.
- Juan Frausto-Solis, Ernesto Liñán-García, Guillermo Santamaria-Bonfil. (2014). Tuned Simulated Annealing Based on Boltzmann and Bose-Einstein Distribution Applied to Maxsat Problem. *Journal of Asian Scientific Research, 4*, 14-27.
- Momin Jamil, Xin-She Yang,. (2013). A Literature Survey of Benchmark Functions For Global Optimization Problems. *Int. Journal of Mathematical Modelling and Numerical Optimisation*, 150 - 194. doi:10.1504
- S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi. (1983). Optimization by Simulated Annealing. *SCIENCE*, 671 - 680.