Simulated Annealing and Hill Climbing

Group Members :

1) **Ansh Rastogi 2021A7PS0515P**

2) **Hardik Gupta 2021A7PS2421P**

3) **Aadeesh Garg 2021A7PS0446P**

Table of Contents

Overview

The assignment on improvements in simulated annealing and hill climbing algorithms will explore the developments and enhancements made to these two popular optimization algorithms. Simulated annealing and hill climbing algorithms are used to solve a wide range of optimization problems, from logistics to engineering and beyond.

The assignment will begin by introducing the basic concepts and workings of simulated annealing and hill climbing algorithms. It will describe the algorithmic framework of each and highlight their strengths and weaknesses.

Next, the assignment will delve into recent improvements made to these algorithms. This may include modifications to the algorithms themselves, such as the incorporation of hybrid techniques that combine multiple optimization methods, as well as the development of novel algorithms that build upon the foundation of simulated annealing and hill climbing.

Additionally, the assignment will discuss improvements to the underlying data structures and techniques used to implement these algorithms, such as the use modifying the existing algorithms and giving them extra functionalities.

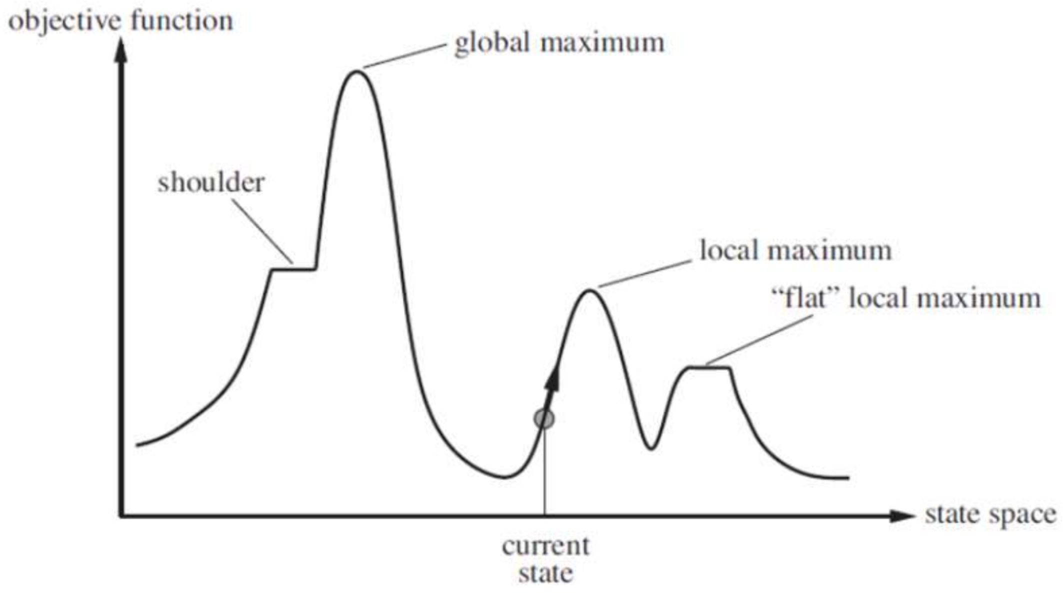
Furthermore, the assignment will analyse the practical applications of these improved algorithms in various fields. It will discuss real-world case studies that demonstrate the effectiveness of these algorithms in solving complex optimization problems.

Finally, the assignment will conclude by summarising the key takeaways and implications of the developments made to simulated annealing and hill climbing algorithms. It will reflect on the potential future directions of these algorithms and the opportunities they present for advancing optimization techniques in various fields.

Background

Hill climbing algorithm

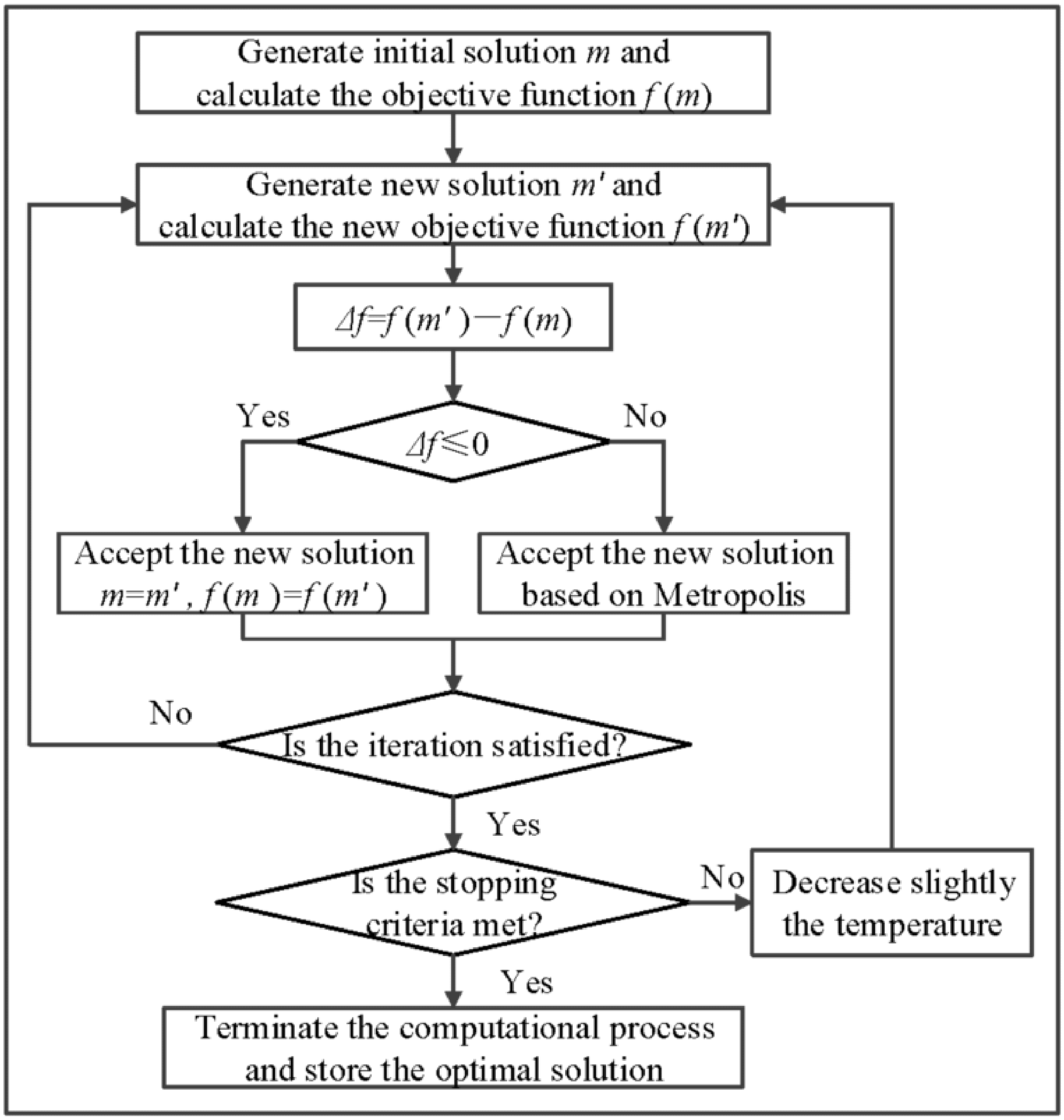
The hill climbing algorithm is an algorithm used to find the maxima(or alternatively, the minima) in optimization problems. The algorithm is created to tackle a situation which can be thought of by imagining a robot which needs to find the highest peak in a mountain range, but is only able to view the points in its immediate vicinity. The solution to this problem can be found by starting at an arbitrary point and moving on to larger values near the point by looking at the values of surrounding points. Once the robot identifies a larger point, it moves to this newly identified position and repeats the process over and over again until it finds a point which does not have a larger value in its immediate vicinity. This allows it to find a point which is a local maxima. Unfortunately, it is impossible to tell if this point is in fact the largest value in the given dataset (the highest peak in the mountain range in our example) or merely a value larger than all its neighbours. Other limitations include getting stuck on plateaus and even diagonal ridges.



Simulated Annealing

Simulated annealing is a stochastic optimization algorithm inspired by the annealing process in metallurgy, where a metal is slowly cooled down to achieve its lowest energy state. The idea was to apply the same principles to optimization problems, where the objective is to find the global minimum of a function. The simulated annealing algorithm uses a probabilistic approach to explore the search space, allowing it to escape local minima and find the global maxima. Over the years, simulated annealing has been widely used in various fields, including computer science, engineering, operations research, and physics, to solve a wide range of optimization problems. Despite the emergence of many new optimization algorithms, simulated annealing

remains a popular and effective tool for solving complex optimization problems. It stresses over having a purely random walk to select the optimal starting point.
The limitation however that it has is quite similar to the Hill Climbing algorithm, it does not guarantee the result is a global maxima.



Generate initial solution m and calculate the objective function $f(m)$
Generate new solution m' and calculate the new objective function $f(m')$
$\Delta f=f(m')-f(m)$
Yes
$\Delta f\leqslant 0$
No
Accept the new solution $m=m', f(m)=f(m')$
Accept the new solution based on Metropolis
No
Is the iteration satisfied?
Yes
Is the stopping criteria met?
No
Decrease slightly the temperature
Yes
Terminate the computational process and store the optimal solution

Literature Review

Theoretical Foundations

- Metropolis-Hastings algorithm: Simulated annealing is based on the Metropolis-Hastings algorithm, which is a Markov chain Monte Carlo method. This algorithm uses a probability distribution to guide the search for the optimal solution.
- Temperature: The key parameter in simulated annealing is the temperature, which controls the acceptance probability of worse solutions. The algorithm starts with a high temperature, allowing it to explore a large solution space, and gradually decreases the temperature to converge to a near-optimal solution.
- Cooling schedules: There are various cooling schedules that can be used to control the temperature, including linear cooling, geometric cooling, and logarithmic cooling.

Applications

- Travelling salesman problem: Simulated annealing has been applied to the travelling salesman problem, which involves finding the shortest possible route that visits a set of cities and returns to the starting point.
- Vehicle routing problem: Simulated annealing has also been applied to the vehicle routing problem, which involves finding the optimal routes for a fleet of vehicles to deliver goods to a set of customers.
- Graph colouring problem: The graph colouring problem involves assigning colours to the vertices of a graph in such a way that adjacent vertices have different colours. Simulated annealing has been used to find near-optimal solutions to this problem.
- Quadratic assignment problem: Simulated annealing has also been used to solve the quadratic assignment problem, which involves assigning n facilities to n locations in such a way that the sum of the distances between facilities is minimised.

Variants

- Fast annealing algorithm: The fast annealing algorithm is a variant of simulated annealing that uses a faster cooling schedule to converge more quickly to a near-optimal solution.
- Adaptive simulated annealing algorithm: The adaptive simulated annealing algorithm is a variant that adjusts the cooling schedule based on the progress of the search, allowing it to converge more quickly.
- Parallel tempering algorithm: The parallel tempering algorithm is a variant that uses multiple chains with different temperatures to explore the solution space more effectively.

Comparison with other optimization algorithms

- Genetic algorithms: Genetic algorithms are a popular optimization algorithm that uses the principles of natural selection and evolution to find near-optimal solutions. Simulated annealing has been shown to perform better than genetic algorithms on problems with rugged landscapes and a large number of local optima.
- Particle swarm optimization: Particle swarm optimization is another optimization algorithm that is inspired by the behaviour of swarms of birds or insects. Simulated annealing has been shown to outperform particle swarm optimization on some problems.
- Ant colony optimization: Ant colony optimization is an optimization algorithm that is inspired by the behaviour of ant colonies. Simulated annealing has been shown to perform better than ant colony optimization on some problems.

Limitations

- Convergence speed: Simulated annealing requires a large number of iterations to converge to a near-optimal solution, which can make it computationally expensive.
- Local optima: Simulated annealing can get stuck in local optima, which can prevent it from finding the global optimal solution.

- Temperature schedule: Selecting the right temperature schedule can be challenging, and a poorly chosen schedule can prevent simulated annealing from finding a near-optimal solution.

Assignment problem statement and its details

Problem Statement

Optimisation of the Simulated Annealing Algorithm.

Details

We were aware of the various limitations of the simulated annealing and hill climbing algorithms. The motivation behind this was to tackle as many limitations as possible by modifying the existing algorithms and even adding in more functionalities to these algorithms.

We generate random data sets and apply our modified algorithms on them to extract the result sets and perform a well detailed analysis on them to develop a conclusion as to how we can optimise these algorithms.

By changing the temperature function, we generate enough analysis to justify and uphold one modified function that successfully optimised the existing algorithm.

We can thus use this modified algorithm to solve real world problems more efficiently.

Implementation

Data set generation

absorandom2d

Generates a single array which has absolutely random values within the bound specified.

absorandom3d

Generates an array of arrays each of which has absolutely random values within the bound specified.

gen_row

Generates an array with each element varying within a given bound, with the starting element specified.

lessrandom2d

Generates an array with each element varying within a given bound, with the starting element specified.

lessrandom3d

Generates the first row using the gen_row function and subsequent rows by using the number right above the current element and checks that any two adjacent elements in the subsequent rows are within the bound specified.

Algorithms

Hill climbing 2D

Returns the local maxima from the given array.

Hill climbing 3D

Returns the local maxima from the given dataset and there is no guarantee of returning a global maxima.

Simulated Annealing 1D

Takes a 1-D array, the time limit and the temperature function T as input parameters. It will map out the entire dataset and look for a local maxima which will be the most optimised solution. Also we can change the temperature function to observe changes to the algorithm. The algorithm will only run for the specified time.

Simulated Annealing 2D

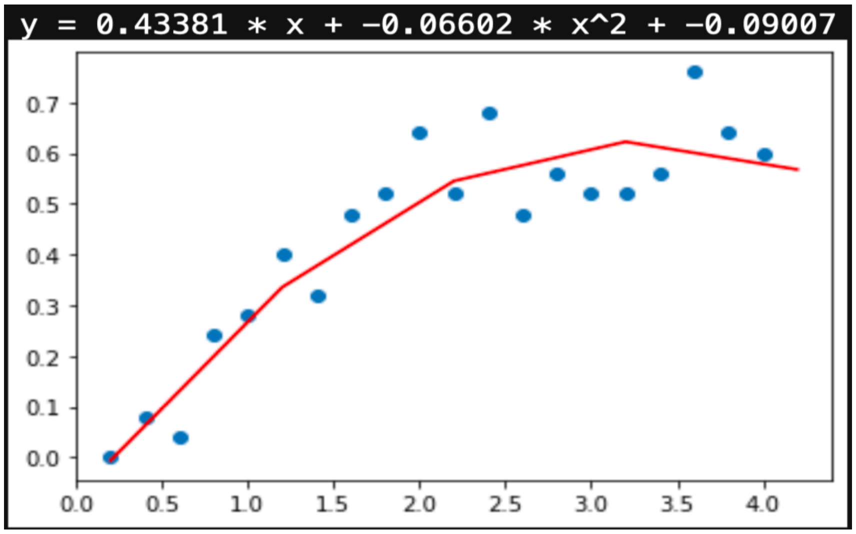
Takes a 2-D array, the time limit and the temperature function T as input parameters. It will map out the entire dataset and look for a local maxima which will be the most optimised solution. Also we can change the temperature function to observe changes to the algorithm. The algorithm will only run for the specified time.
The results obtained by varying the temperature function are collected and the most optimised and efficient function can be used for solving real problems.

Results and Analysis

1)Temperature = time_limit-t

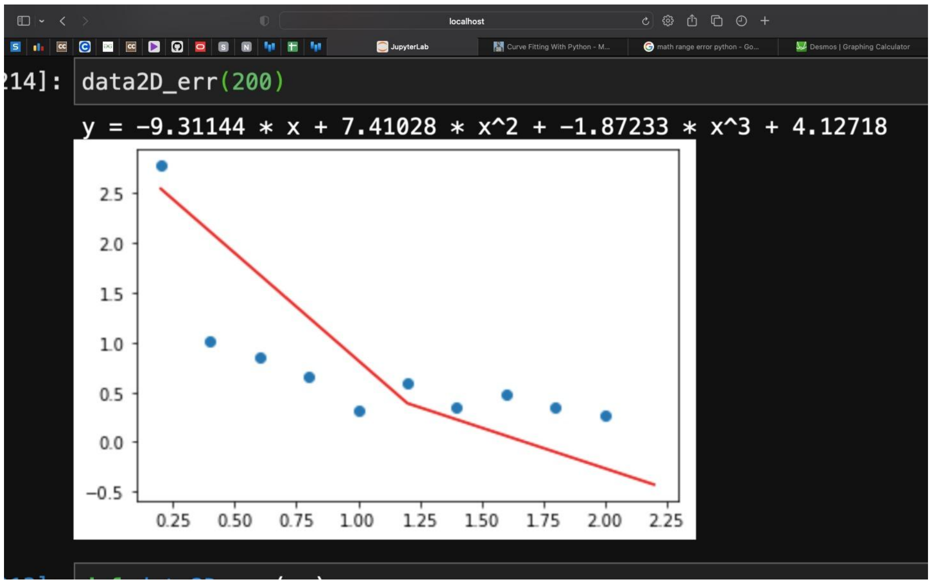
Simulated Annealing 3-d with a dataset of size 200, 20 different times each incremented by a value of 0.2 and for each different time 25 test cases are generated.

Probability of correct answer



```
y = 0.43381 * x + -0.06602 * x^2 + -0.09007
```

Relative error in the result according to the time taken

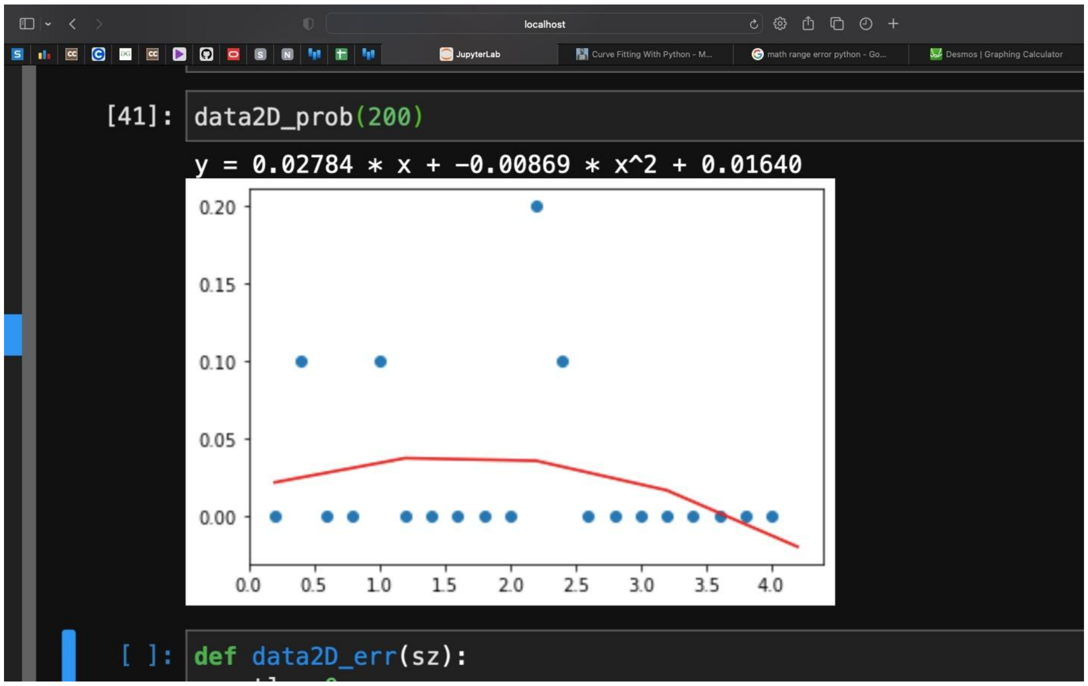


```
[214]: data2D_err(200)
y = -9.31144 * x + 7.41028 * x^2 + -1.87233 * x^3 + 4.12718
```

2)Temperature = 4^t(time_limit-t)

Simulated Annealing 3-d with a dataset of size 200, 10 different times each incremented by a value of 0.2 and for each different time 20 test cases are generated.

Probability of correct answer



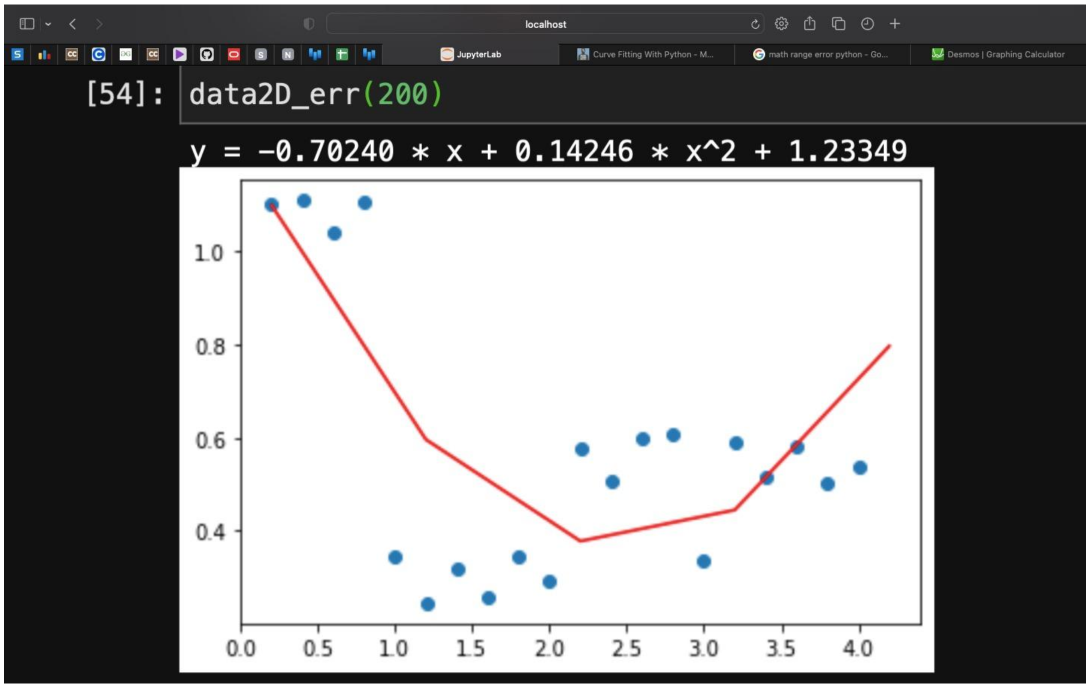
```
[41]: data2D_prob(200)

y = 0.02784 * x + -0.00869 * x^2 + 0.01640
```

localhost
JupyterLab
Curve Fitting With Python - M...
math range error python - Go...
Desmos | Graphing Calculator
0.20
0.15
0.10
0.05
0.00
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

```
[ ]: def data2D_err(sz):
```

Relative error in the result according to the time taken



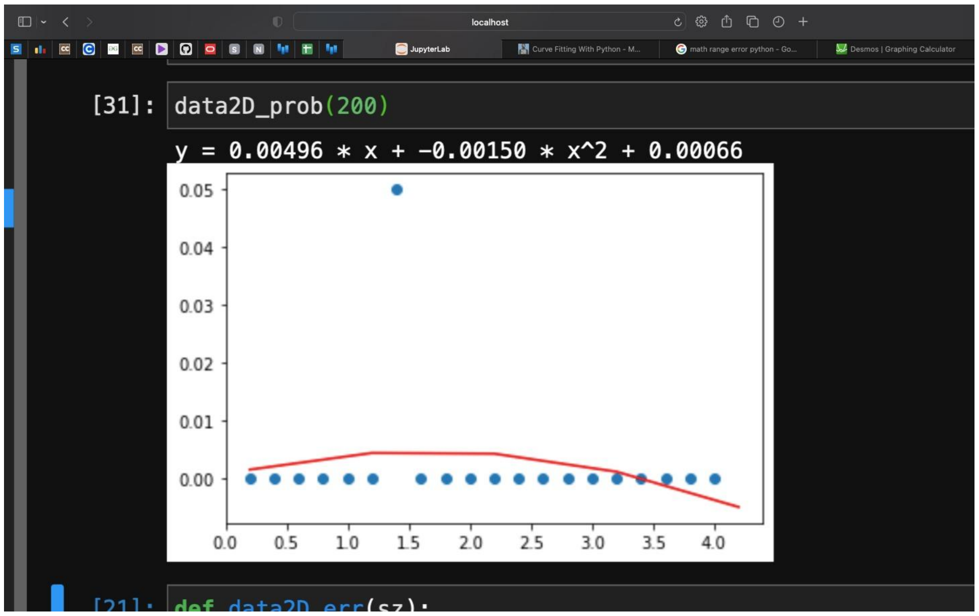
```
[54]: data2D_err(200)

y = -0.70240 * x + 0.14246 * x^2 + 1.23349
```

localhost
JupyterLab
Curve Fitting With Python - M...
math range error python - Go...
Desmos | Graphing Calculator
1.0
0.8
0.6
0.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

3)Temperature =(time_limit-t)/log(t+2)
Simulated Annealing 3-d with a dataset of size 200, 10 different times each incremented by a value of 0.2 and for each different time 20 test cases are generated.

Probability of correct answer

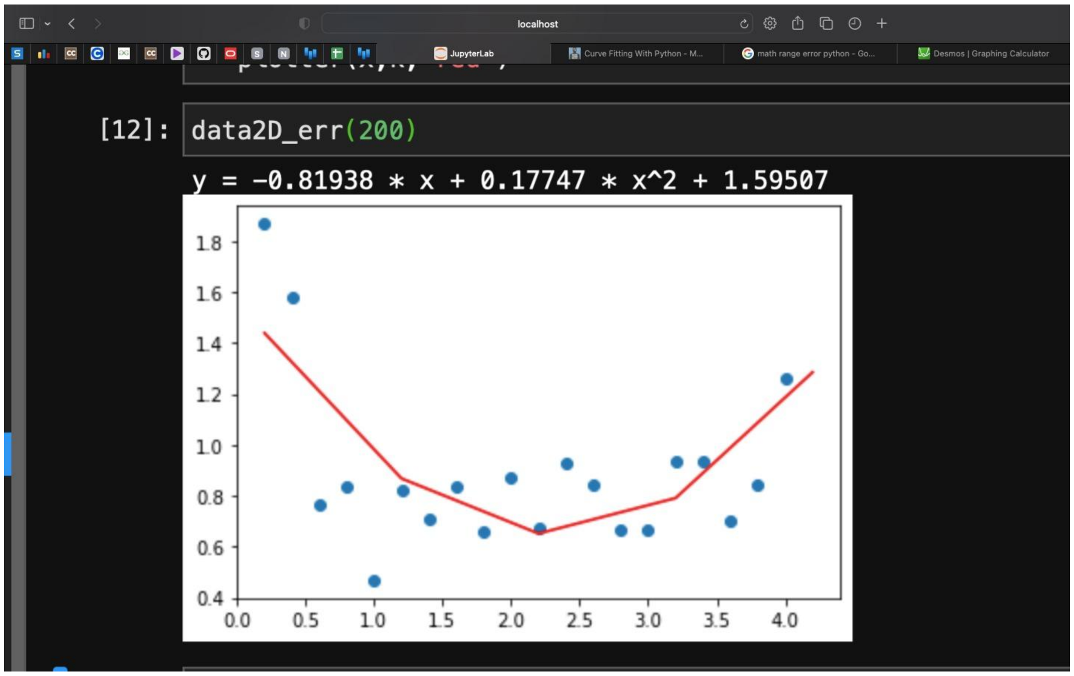

localhost
JupyterLab
Curve Fitting With Python - M...
math range error python - Go...
Desmos | Graphing Calculator
0.05
0.04
0.03
0.02
0.01
0.00
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

```
[31]: data2D_prob(200)

y = 0.00496 * x + -0.00150 * x^2 + 0.00066
```

```
[21]: def data2D_err(sz):
```

Relative error in the result according to the time taken


localhost
JupyterLab
Curve Fitting With Python - M...
math range error python - Go...
Desmos | Graphing Calculator
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

```
[12]: data2D_err(200)

y = -0.81938 * x + 0.17747 * x^2 + 1.59507
```

Conclusion

We can see that for each of these functions, the error is decreasing as we increase the time limit for each algorithm. The second algorithm does not lead to very good results, and hardly ever gives the maximum value. The best results were obtained for the first function which is linear in nature. The final function gives intermediate values.

In conclusion, we find that simulated annealing is a promising optimization algorithm that can be applied to a wide range of optimization problems. Its ability to escape local optima and its robustness to noisy and stochastic environments make it a valuable tool for solving complex real-world problems.
However, its performance can be highly sensitive to the choice of parameters, and careful tuning is necessary to achieve optimal results. Overall, simulated annealing provides a powerful optimization technique that can be used to solve a wide range of challenging problems.

References

https://machinelearningmastery.com/simulated-annealing-from-scratch-in-python/#:~:text=Simulated%20Annealing%20is%20a%20stochastic,algorithms%20do%20not%20operate%20well

https://www.sciencedirect.com/topics/mathematics/metropolis-hasting-algorithm

https://in.mathworks.com/help/gads/what-is-the-genetic-algorithm.html

https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization/

http://www.scholarpedia.org/article/Ant_colony_optimization

https://towardsdatascience.com/optimization-techniques-simulated-annealing-d6a4785a1de7