Optimization using Simulated Annealing

Kunal Gehlot - 190020065
Ramkrishna Kamble - 190020094
Jos Katiyar - 190020054

Idea

- SA algorithm is a heuristic methods for solving the optimization problems.
- Simulated annealing is based on **metallurgical practices.** When you heat a particular metal, there's a lot of energy there, and you can move things around quite systematically. But over time, as the system cools down, it eventually settles into a final position.
- It mimics the Physical Annealing process but it is used for **optimizing parameters in a model.**

Idea

- The Simulated Annealing algorithm is commonly used when we're stuck trying to optimize solutions that generate local minimum or local maximum solutions

- SA algorithm proposes an effective solution to this problem as incorporating two iterative loops which are the cooling procedure for the **annealing process** and **Metropolis criterion**.

- Basic idea behind the Metropolis criterion is to be executed randomly to extra search the neighborhood of the candidate solution to avoid being trapped to local extreme points.

Idea

Physical Annealing	Simulated Annealing
Metal	Optimization Problem
Energy State	Cost Function
Temperature	Control Parameter
Crystalline Structure	The optimal Solution
Global optima solution can be achieved as long as the cooling process is slow enough	

Algorithm (Minimization)

- Important part of this algorithm is the following analogy with Thermal Dynamics:

$$P(\Delta E) = e^{-\frac{\Delta E}{k*t}}$$

- In this algorithm, we define an initial temperature as hyperparameter, often set as 1, and a minimum temperature, on the order of 10^-4. The current temperature is multiplied by some fraction alpha and thus decreased until it reaches the minimum temperature.

Algorithm (Minimization)

- For each distinct temperature value, we run the core optimization routine a fixed number of times. The optimization routine consists of finding a neighboring solution and accepting it with probability defined below

$$P = \begin{cases} 1 & if \Delta c \leq 0 \\ e^{-\Delta c / t} & if \Delta c > 0 \end{cases}$$

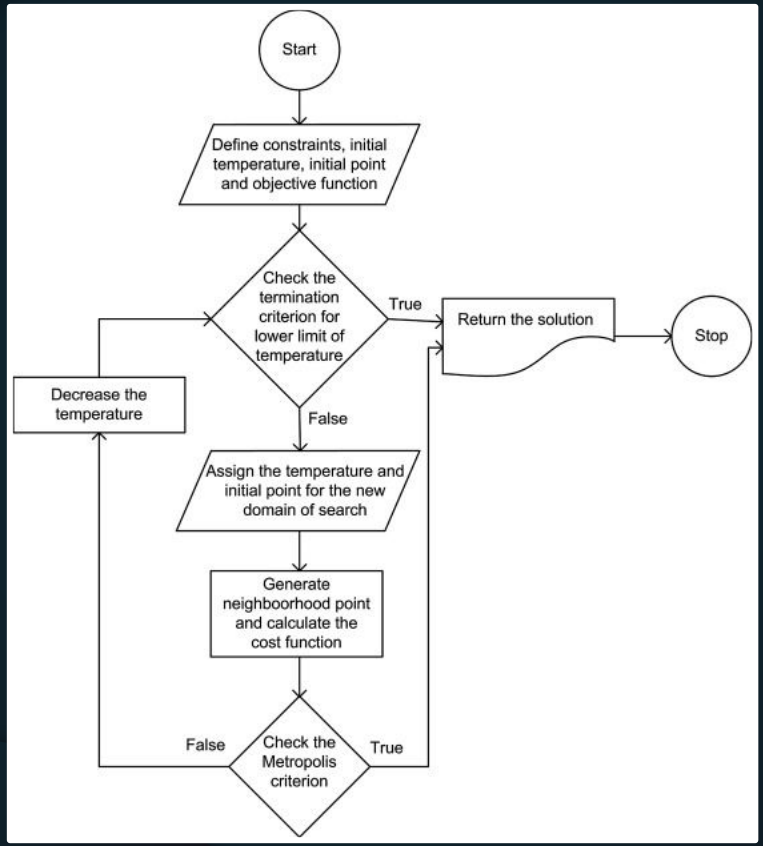
- **Where exp() is e (the mathematical constant) raised to a power of the provided argument, and objective(new), and objective(current) are the objective function evaluation of the new (worse) and current candidate solutions.**

Algorithm (Minimization)

- The intent is that the high temperature at the beginning of the search will help the search locate the basin for the global optima and the low temperature later in the search will help the algorithm hone in on the global optima.

- Before starting with the algorithm an initial solution is needed so that algorithm knows where to start. This can be done in two ways:
  - Using prior knowledge about the problem to input a good starting point
  - Generating a random solution.

Algorithm (Minimization)



Literature Survey

Simulated Annealing and Genetic Algorithms

- Genetic Algorithms are very different. For one thing--and this is a big thing--it generates not a single candidate solution unlike Simulated annealing but an entire 'population of them'.

- A Genetic Algorithm maintains a population of possible solutions, and at each step, selects pairs of possible solution, combines them (crossover), and applies some random changes (mutation). The algorithm is based the idea of "survival of the fittest" where the selection process is done according to a fitness criteria

Literature Survey

Simulated Annealing and Genetic Algorithms

- While Simulated Annealing only tracks one solution in the space of possible solutions, and at each iteration considers whether to move to a neighboring solution or stay in the current one according to some probabilities
- GA nearly always beats SA (returns a lower 'best' return value from the cost function--ie, a value close to the solution space's global minimum), but at a higher computation cost.
- But there are situation when SA outperform GA. The general scenario would be those optimization problems having a small solution space so that the result from SA and GA are practically the same, yet the execution context favors the faster algorithm (which should always be SA).

Problem Implemented

Problem:

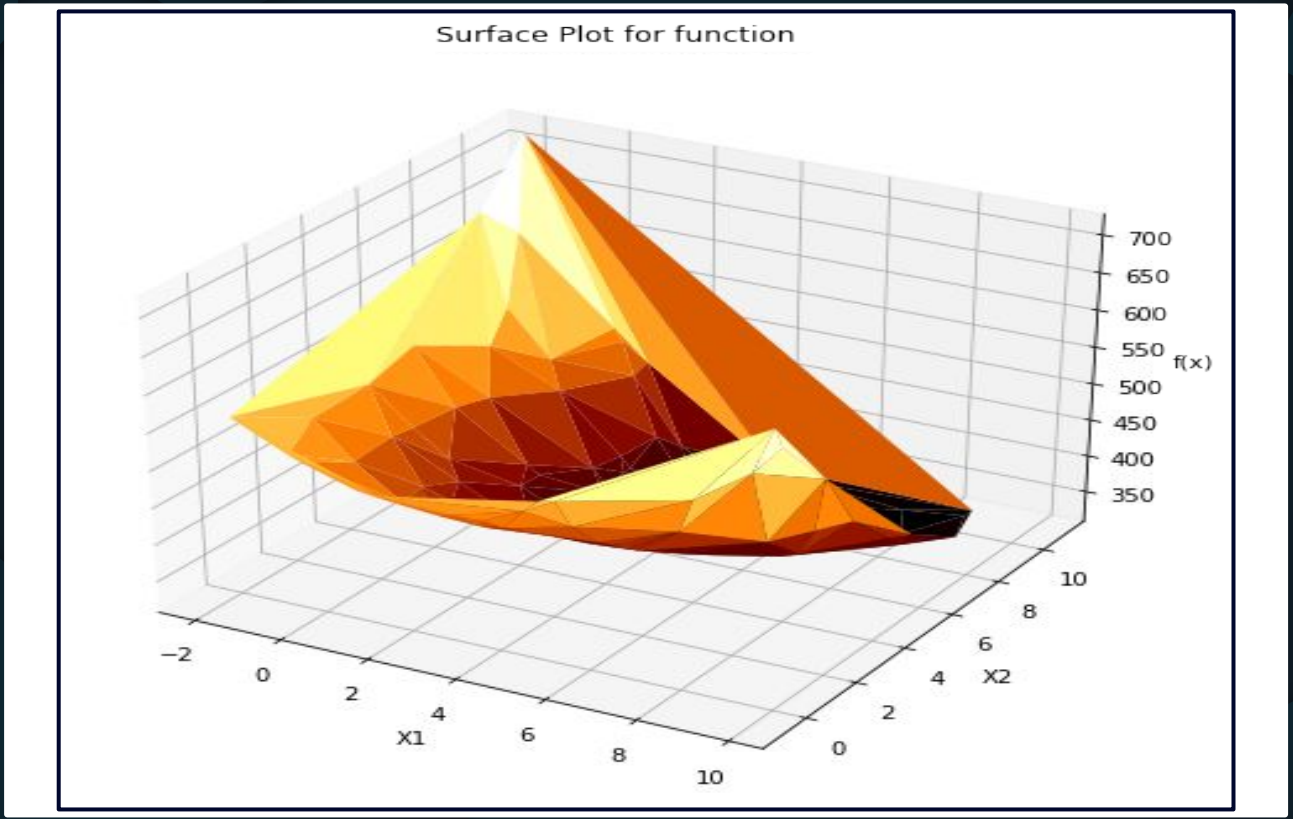
$$\min f(x)$$

$$\text{s.t } -2<x1<10$$
$$-1<x2<11$$

where,

$$f(x) = 500-20x1-26x2-4x1x2+4x1^2+3x2^2$$

Visualizing the function

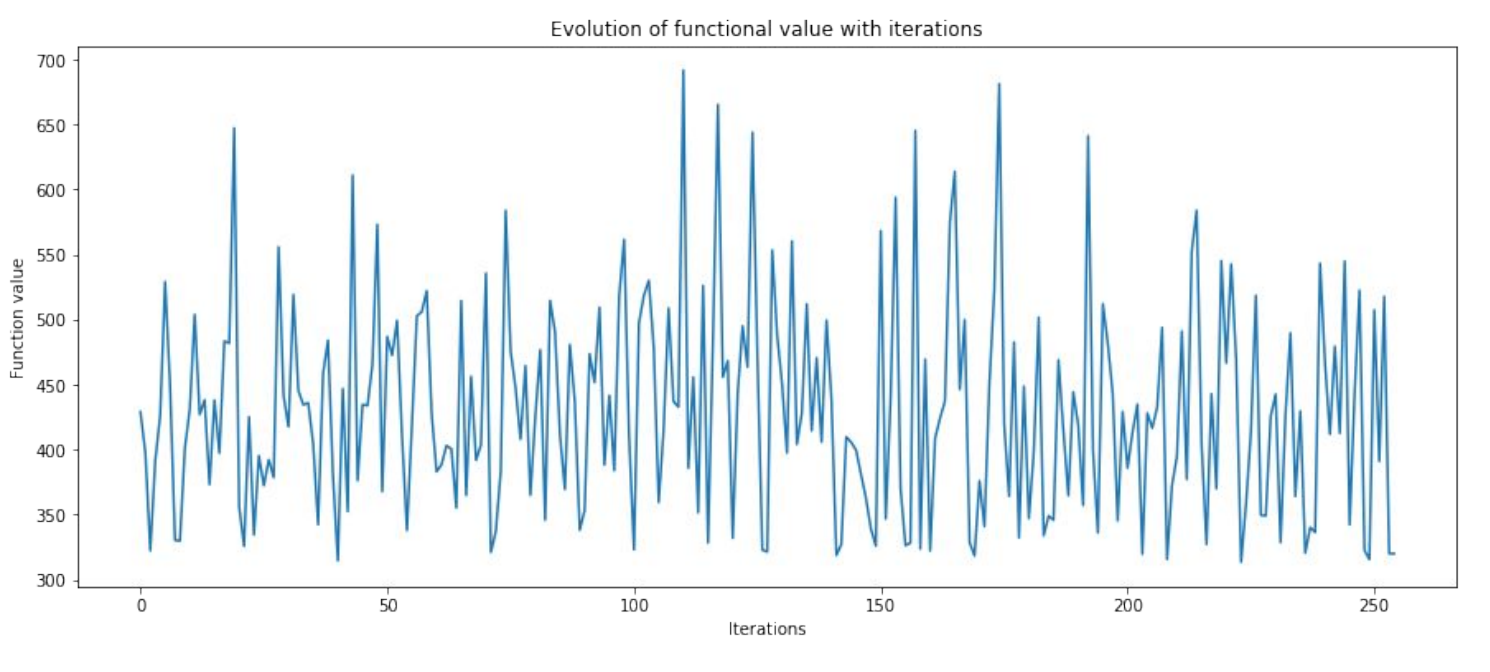


Results

Final minimum Value: 319.920090059250 36
Converging Value of x: [8.23937813 ,10.79646426]
No. of iterations required : 254

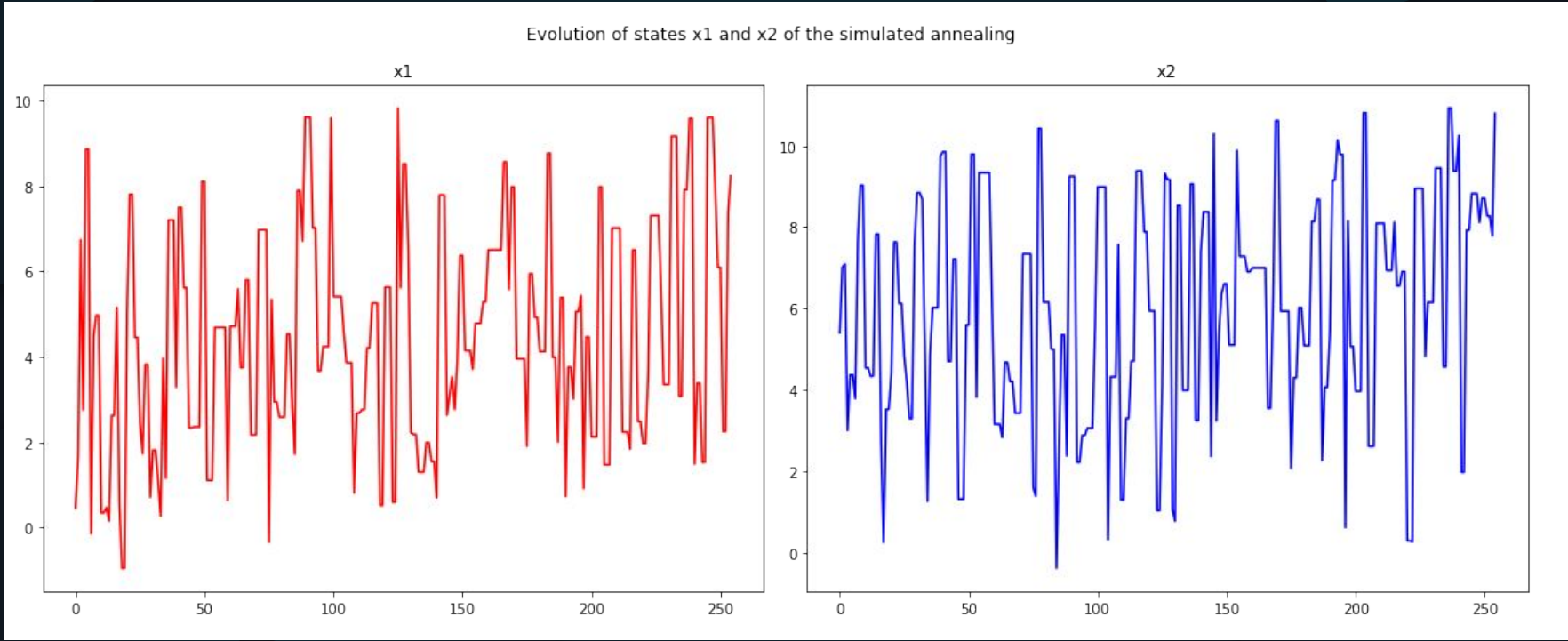
Results . . .

How value of Function changes wrt. iterations



Results...

How State i.e x1 and x2 values changes wrt. iterations



Conclusions and Remarks

Simulated Annealing Algorithm works more nicely on cost functions which are oscillating and have lot of local minimums. Algorithm which we studied in class such as steepest descent, Newton CG ,BGFS may get stuck in one of minimums. This algorithm will mostly not face this problem.

But for simple function this algorithm seems like an overkill. Simpler algorithms mentioned above seems much better and faster

Conclusions and Remarks

But functions with lot of minimum values are complex. And due to complex cost function algorithm becomes too slow and computationally expensive

Another disadvantage will be that this method cannot tell if the solution is optimal. Some other method is required to use in combination of this

THANKS!