# Solving Travelling Salesman Problem using Simulated Annealing

Prashil Tumbade

# Contents

# Introduction

- Searching through a very large number of possible solutions to find the best (optimal) one can be an extremely difficult task, (if not practically impossible)
- The number of possible solutions can be too large even for current powerful computers.
- Because of this, we need a solutions that is close or good enough
- We need a technique or algorithm which can find a good enough solution in reasonable amount of time

# Travelling Salesman Problem

Problem Definition

- The travelling salesman problem consists of a salesman and a set of cities. The salesman has to visit each one of the cities starting from a certain one (e.g. hometown) and returning to the same city. The challenge of the problem is that the travelling salesman wants to minimize the total length of the trip.

# Brute Force Approach

- As we know, one method to find optimal tour
  - Make list of all possible routes (tours)
  - Calculate the total distance of each tour by adding up the distances between consecutive cities
  - Choose the tour with smallest distance
- This is called Exhaustive Search because all the combinations are attempted
- Brute force solution: $O(n!)$

# Simulated Annealing

- Inspired from the process of annealing in metal work

  For obtaining low energy states of a solid metal

- Annealing in metal work:
  - Increase the temperature of the heat bath to a maximum value at which the solid melts.
  - Decrease carefully the temperature of the heat bath until the particles arrange themselves in the ground state of the solid. Ground state is a minimum energy state of the solid.
  - The ground state of the solid is obtained only if the maximum temperature is high enough and the cooling is done slowly.

- The process of annealing can be simulated with the Metropolis algorithm, which is based on Monte Carlo techniques.
- This algorithm can be used to generate a solution to combinatorial optimization problems.
- We assume the following analogy (i.e equivalence)
  - Solutions in the problem ~ States in a physical system
  - Cost of a solution ~ Energy of the state

# Simulated Annealing Algorithm

- We will have a temperature variable to simulate the heating process.
- We will assign it a high initial value and then slowly decrease it (cool it) as the algorithm runs
- We will accept the solutions (more frequently) which are worse than our current solution as long as this temperature variable is still high
- By doing this, we allow the algorithm to jump out of any local optimums it goes to early while it is executing
- The chance of accepting worse solutions will reduce as the temperature decreases.
- By doing this, we allow the algorithm to slowly focus on an area of the search space in which a close to optimum solution can be found

# Acceptance Function

- How do we decide which solutions to accept?
    - First we check if the neighbour solution is better than our current solution.
    - If it is, we accept it unconditionally
    - If however, the neighbour solutions is not better we need to consider two factors:
        - Firstly, how much worse the neighbour solution is
        - Secondly, how high the current ‘temperature’ of our system is
- At high temperatures the system is more likely to accept solutions that are worse.
- The math for this is simple:

$$P = \exp(\,(\text{solutionEnergy} - \text{neighbourEnergy}) / \text{temperature}\,)$$

$$P > r\ ?$$

- Basically, smaller the change in energy and higher the temperature, the more likely it is for the algorithm to accept the solution

# Algorithm Overview

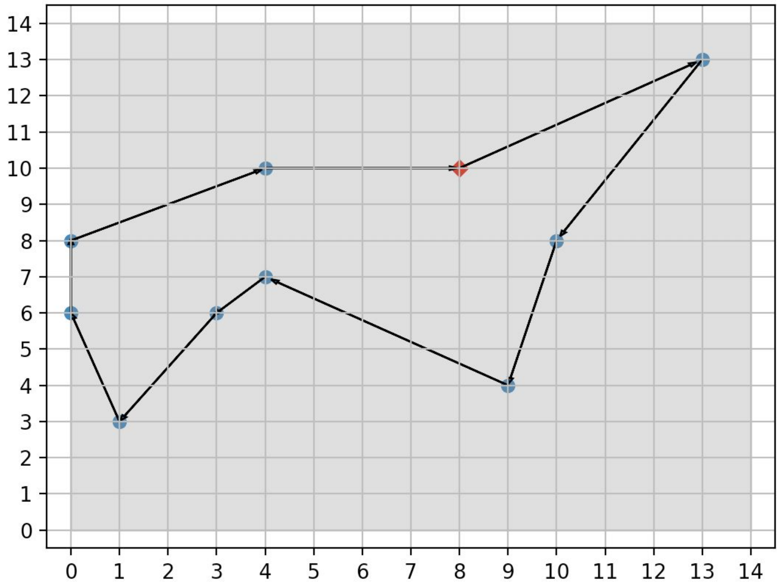
- We will first set the initial temperature and create a random initial solution
- Then we will begin looping until our stop condition is met:
  - Either the system has sufficiently cooled
  - Or a good enough solution has been found
- We will select a neighbour by making a small change to our current solution
- We then decide whether to move to that neighbour solution
- Finally, we decrease the temperature and continue looping

# Advantages of Simulated Annealing

- Really good at avoiding the problems of getting stuck in a local optimum
- Also good at finding a good solutions (i.e an approximate global optimum)

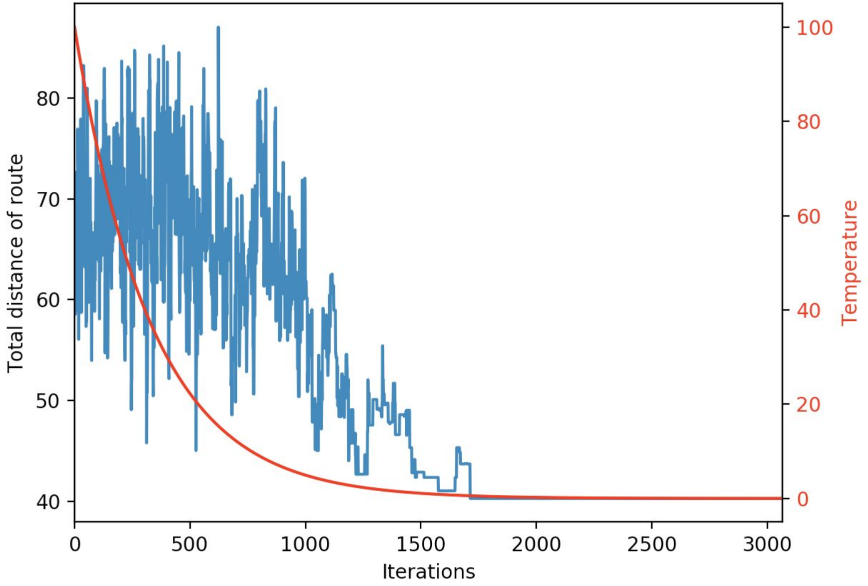
# Results

Optimized route for 10 cities




Time

- Brute force : 23.60 sec
- SA : 0.34 sec




Convergence rate and cooling schedule

# Conclusion

- Simulated Annealing outperforms the Brute force method
- Simulated Annealing guarantees a convergence upon running sufficiently large number of iterations.

Thank you