CS362 Report (Lab 3)

Submitted to: Dr. Pratik Shah

Akshay	Ankur Kumar Shukla	Kalash Kala	Keshav Garg
202051018	*202051029*	*202051095*	*202051102*

***Abstract*—Travelling Salesman Problem (TSP) is a hard problem, and is simple to state. Given a graph in which the nodes are locations of cities, and edges are labelled with the cost of travelling between cities, find a cycle containing each city exactly once, such that the total cost of the tour is as low as possible.**

I. Introduction

This document is elaborating our work of week 4 (Lab 3) of CS362 (Artificial Intelligence Laboratory).

Learning Objective: Non-deterministic Search — Simulated Annealing For problems with large search spaces, randomized search becomes a meaningful option given partial/ full-information about the domain.

For the state of Rajasthan, find out at least twenty important tourist locations. Suppose your relatives are about to visit you next week. Use Simulated Annealing to plan a cost effective tour of Rajasthan. It is reasonable to assume that the cost of travelling between two locations is proportional to the distance between them.

II. Theory :

A. Non-deterministic search

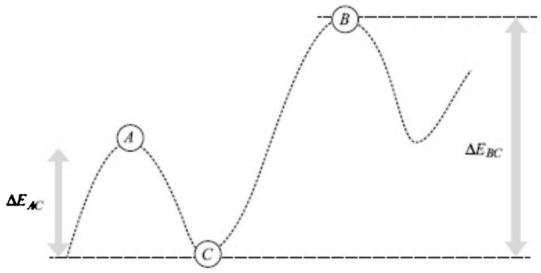
Non-deterministic search is a type of search algorithm used in artificial intelligence to solve problems where there are multiple possible solutions, but it is not clear which solution is the best or most efficient. In contrast to deterministic search algorithms, which systematically explore all possible solutions, non-deterministic search algorithms use randomness to explore the solution space in a more efficient way.

In non-deterministic search, the algorithm explores the solution space by making random choices at each step. These random choices can include selecting a random node to expand or selecting a random path to follow. The goal is to find a solution that satisfies certain criteria, such as minimizing the cost of a solution or maximizing the likelihood of a certain outcome.

Non-deterministic search algorithms have the advantage of being able to explore a large solution space in a relatively short amount of time. However, they also have the drawback of not guaranteeing that the best solution will be found, since the randomness of the search may cause the algorithm to miss optimal solutions. Nonetheless, non-deterministic search algorithms are an important tool in AI for solving complex problems where deterministic methods may not be feasible.

B. Simulated Annealing

- The basic idea is that the algorithm makes a probabilistic move in a random direction.
- The probability of making the move is proportional to the gain of value made by the move.
- Traditionally, this gain is associated with $Energy^2$ and we use the term ΔE to represent the change in the evaluation value.
- The larger the gain, the larger is the probability of making the move.



Source: A first course in Artificial Intelligence, Deepak Khemani

Both A and B are maxima with B having a higher evaluation value.

It is likely that the algorithm moves from A to C more often than from B to C because ΔE from A to C is less negative than from B to C

Also, the algorithm is more likely to move from C to B than to A (positive gain)

The probability of making a move from the current node C to the new node N :

$$P(C, N) = \frac{1}{1 + e^{-\frac{\Delta E}{T}}}$$

T is an externally controlled parameter called temperature. Note that the above formula is for maximization of the evaluation function value. For minimization, the negative sign in the denominator will be removed.

III. Pseudo Code :

Following are the pseudo codes of main and most important functions -

Algorithm 1 Simulated Annealing Function

$path \leftarrow initialPath$
$cost \leftarrow getCost(path)$
$T_m \leftarrow$ initial temperature
$coolingFactor \leftarrow$ initial value
$itr \leftarrow 0$
while $itr \neq itrMax$ **do**
    $nextPath \leftarrow getNeighbour(path)$
    $nextCost \leftarrow calculateCost(nextPath)$
    $\Delta E \leftarrow cost - nextCost$
    $T \leftarrow T_m * coolingFactor$
    if T is too low ($\leq 10^{-6}$) **then**
        break loop
    end if
    if $\Delta E > 0$ **then**
        $path \leftarrow nextpath$
    else if $random(0,1) < 1/(1+e^{\frac{-\Delta E}{T}})$ **then**
        $path \leftarrow nextPath$
    end if
    $itr \leftarrow itr + 1$
end while
return $path$

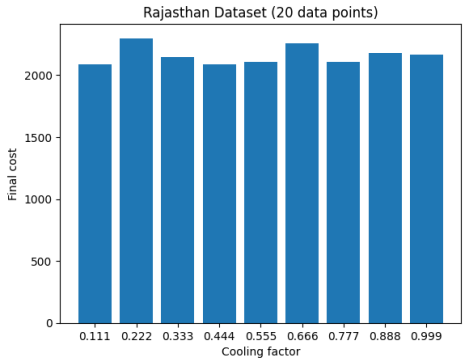
Algorithm 2 Tour Cost Function

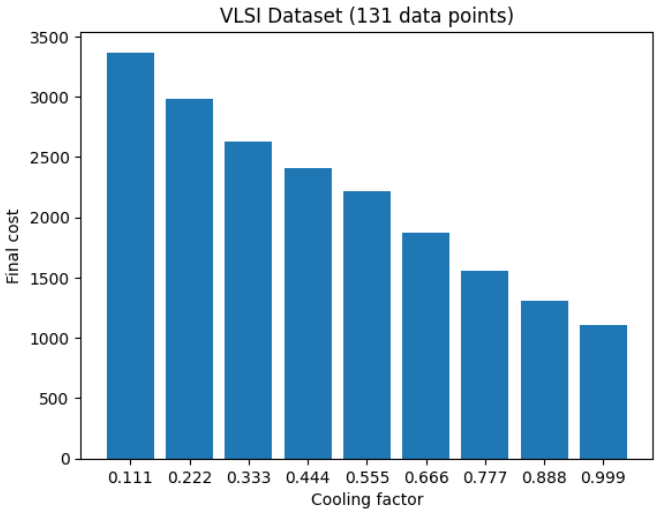
$distance \leftarrow 0$
$itr \leftarrow 0$
while $itr \neq path.length() - 1$ **do**
    $distance \leftarrow distance+$ distance b/w node $path[i]$ and $path[j]$
    $distance \leftarrow distance+$ distance b/w last node and first node
end while
return distance

IV. Experiments and their Graphical Analysis

A. Cooling Factor Vs Final Cost

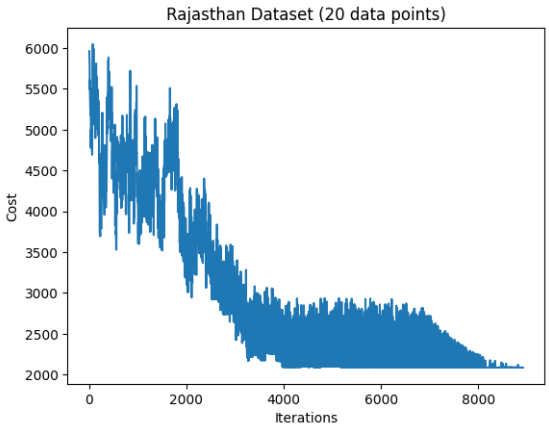
In the above Pseudo code we make use of α which is a cooling factor that tell the code to make a jump in the wrong direction provided that it satisfies the probability. So we thought that how can the value of cooling factor affect the overall final cost of the system, So we ran Simulated Annealing code for α ranging from $0.111 \rightarrow 0.999$

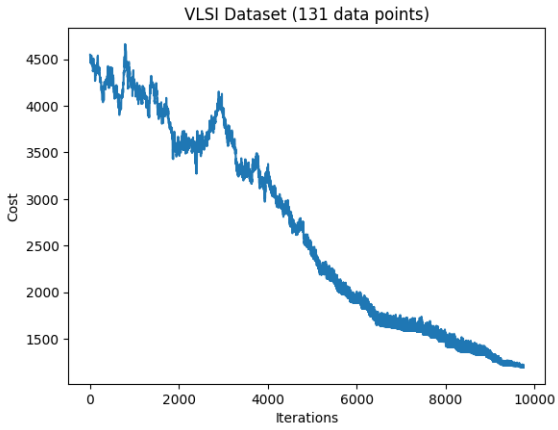




B. No. of Iterations Vs Intermediate Cost

Simulated Annealing code runs for the number of iterations it is fed as a parameter. So one might think that how does the code behave on increasing the number of iterations. So in the code, we added a list that also keeps track of the intermediate costs. Then we plot these costs for every iteration

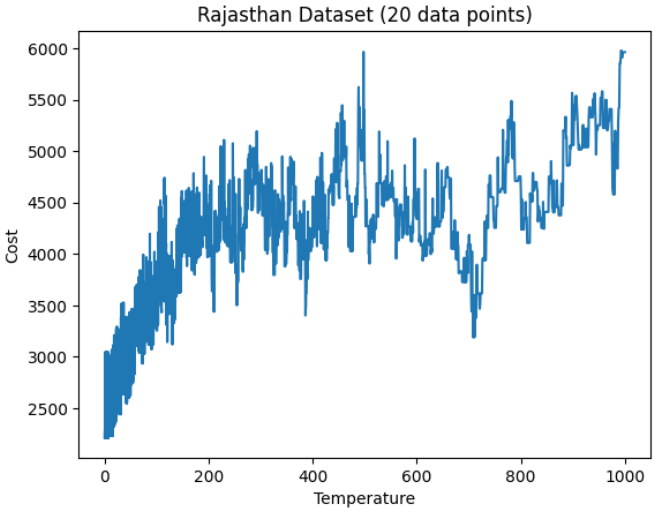


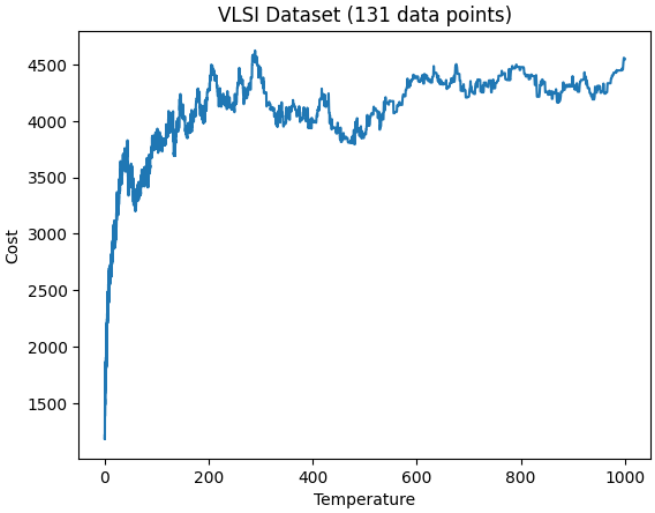


C. Intermediate Temperature Vs Intermediate Cost

Similar to intermediate cost, we have intermediate temperature, which is the temperature recorded at every iteration. These values are stored in a list during the simu-

lated annealing process. This experiment also shows how initial temperature selection can affect the end output





V. Some Final takeaways

- As cooling factor increases, the tendency to get a good path increases
- As the number of iteration increases, the path length we receive decreases till a certain point after that iteration point there is no visible change
- As the intermediate temperature decreases, the intermediate cost not only decreases but also the randomness with which it changes its state also decreases

References

Helped in code implementation:
https://www.geeksforgeeks.org/
program-distance-two-points-earth/
https://github.com/pratikiiitv/cs302/blob/main/tsp_sa.m
https://gis.stackexchange.com/questions/212723/
how-can-i-convert-lon-lat-coordinates-to-x-y
https://youtu.be/rldKl1CNx-A
https://tsplib95.readthedocs.io/en/stable/ (by Robert Grant)
VLSI Data set (by Andre Rohe)
https://www.math.uwaterloo.ca/tsp/vlsi/index.html#XQF131
https://stackoverflow.com/questions/477237/
how-do-i-simulate-flip-of-biased-coin