VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

A Comparison in the Implementation of The Travelling Salesman Problem using Brute – Force and Ant Colony Optimisation

J Component Report

Submitted By:

NAME	REG. NO.	SLOT
AMAN ANAND	19BCE0521	B1
AKSHAT TRIPATHI	19BCE0527	B1
AYUSH KHARE	19BCE0498	B1

In partial fulfilment for the award of the degree of

Bachelor of Technology

in

COMPUTER SCIENCE ENGINEERING

Under the guidance of

Faculty: Prof. Raghavan R

School of Computer Science and Engineering

Academic Year: 2019 - 2020

Review One

Abstract:

Problem statement (Travelling Salesman Problem - TSP) –

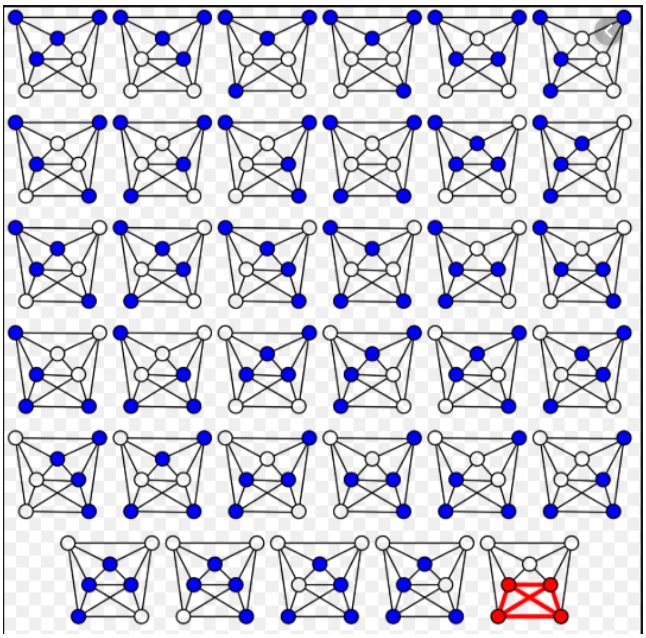
Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point. Measure the time required by Brute Force Strategy and Ant Colony Optimisation to determine the shortest path possible.

The Travelling Salesman Problem is one of the best known NP-hard problems, which means that there is no exact algorithm to solve it in polynomial time. The minimal expected time to obtain optimal solution is exponential. So, for that reason, we usually use heuristics to help us to obtain a “good” solution.

Brute Force Technique:

Introduction-

Brute-force search or exhaustive search, also known as generate and test, is a very general problem-solving technique and algorithmic paradigm that consists of systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem's statement.



This type of algorithm usually involves the direct search of the solution by the compiler constrained to the problem criteria. This algorithm exploits every possible solution available and proceeds in a way so as to select the most optimal solution.
Although this may prove to be a fruititous approach regarding small data sets, it is a very time consuming and compiler over-bearing algorithm with respect to large sets of data. Brute force algorithm has various disadvantages which are derived from the above listed problems. These include:

- Combinatorial Explosion (or) the curse of dimensionality
- Time Complexity
- Memory Storage Efficiency

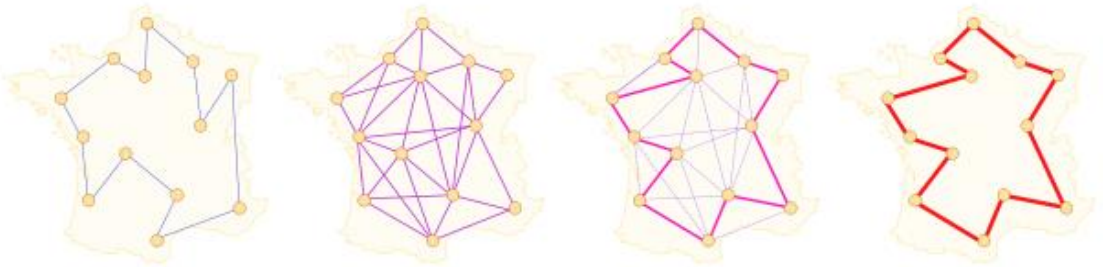
To overcome the above mentioned complexities and disadvantages we have adapted the ACO technique, which has its own set of problems, but is comparatively efficient with respect to the Brute Force Technique.

Ant Colony Optimization (ACO) Technique:

Introduction-

In the natural world, ants of some species (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep travelling at random, but instead to follow the trail, returning and reinforcing it if they eventually find food.
Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path, by comparison, gets marched over more frequently, and thus the pheromone density becomes higher on shorter paths than longer ones. Pheromone evaporation also has the advantage of avoiding the convergence to a locally optimal solution. If there were no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be

constrained. The influence of pheromone evaporation in real ant systems is unclear, but it is very important in artificial systems.
The overall result is that when one ant finds a good (i.e., short) path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually leads to many ants following a single path. The idea of the ant colony algorithm is to mimic this behaviour with "simulated ants" walking around the graph representing the problem to solve.



1 2 3 4

Given an n-city TSP with distances dij, the artificial ants are distributed to these n cities randomly. Each ant will choose the next to visit according to the pheromone trail remained on the paths just as mentioned in the above example. However, there are two main differences between artificial ants and real ants:
(1) The artificial ants have “memory”; they can remember the cities they have visited and therefore they would not select those cities again.
(2) The artificial ants are not completely “blind”; they know the distances between two cities and prefer to choose the nearby cities from their positions. Therefore, the probability that city j is selected by ant k to be visited after city i could be written as follows:

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{s \in allowed_k} [\tau_{is}]^{\alpha} \cdot [\eta_{is}]^{\beta}} & j \in allowed_k \\ 0 & \text{otherwise} \end{cases}$$

where τij is the intensity of pheromone trail between cities i and j, α the parameter to regulate the influence of τij, ηij the visibility of city j from city i, which is always set as 1/dij (dij is the distance between city i and j), β the parameter to regulate the influence of ηij and allowed k the set of cities that have not been visited yet, respectively.

At the beginning, l ants are placed to the n cities randomly. Then each ant decides the next city to be visited according to the probability p_{ij}^{k} given by Eq. (1). After n iterations of this process, every ant completes a tour. Obviously, the ants with shorter tours should leave more pheromone than those with longer tours. Therefore, the trail levels are updated as on a tour each ant leaves pheromone quantity given by Q/Lk, where Q is a constant and Lk the length of its tour, respectively. On the other hand, the pheromone will evaporate as the time goes by. Then the updating rule of τij could be written as follows:

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}$$

$$\Delta\tau_{ij} = \sum_{k=1}^{l} \Delta\tau_{ij}^{k}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{if ant } k \text{ travels on edge } (i,j) \\ 0 & \text{otherwise} \end{cases}$$

Where t is the iteration counter, ρ ϵ [0, 1] the parameter to regulate the reduction of τij, Δτij the total increase of trail level on edge (i, j) and Δτijk the increase of trail level on edge (i, j) caused by ant k, respectively. After the pheromone trail updating process, the next iteration t + 1 will start.

The ACO technique comes under the category of solving the TSP problem statement using a meta-heuristic approach to the solution. Meta-heuristic simply implies the compilers ability to learn from its' previously traversed routes and apply it to make a partially educated guess regarding to remaining part of the algorithm.

There are many disadvantages to approaching the problem using ACO, but this algorithm gives us a much more efficient time complexity **O(nlogn)** with respect to the brute force technique **O(n^2)**. Some of the drawbacks of this algorithm are:

- Probability distribution can change for each iteration.
- Have a difficult theoretical analysis.
- Have uncertain time to convergence.
- Have dependant sequences of random decisions.
- Have more experimental than theoretical research.