INFO 6205 Spring 2023 Final Project

Traveling Salesman

Team member 1: Subbu Manickam

Team member 2: Manikanta Reddy Thikkavarapu

Introduction:

Finding the shortest path between a number of points and places that must be visited is the goal of the algorithmic problem known as the "traveling salesman problem" (TSP). The cities a salesperson might visit are the points in the problem statement.

Aim:

The goal of the issue is to determine the shortest path that travels through each listed city exactly once before turning around and returning to the starting location.

The issue is named after the analogy of a salesman who must visit several towns and wants to travel the fewest amount of distance possible while doing so. The issue has significant applications in network routing, logistics, and other fields.

The TSP is an NP-hard problem, which means that it is computationally impractical to find an accurate solution for large instances of the problem. However, there are a variety of approximation algorithms and heuristics that can be applied to quickly find effective answers.

Approach:

The traveling salesman problem (TSP) can be solved in a number of ways, including accurate and heuristic methods.

Exhaustively examining every potential solution, exact approaches seek to identify the best answer to the issue. These techniques include dynamic programming, cutting-plane algorithms, and branch-and-bound algorithms. These techniques, nevertheless, are computationally expensive and might not be practical for widespread cases of the issue.

Even though the solution is not always optimal, heuristic approaches try to find a good one in a fair period of time. These techniques consist of Nearest neighbor, Insertion, Simulated annealing, and Genetic Algorithms

In our project, we are going with the Simulated annealing approach.

Simulated annealing approach: This is a probabilistic algorithm that starts with an initial solution and iteratively makes small random changes to the solution. The changes are accepted or rejected based on a probability function that gradually decreases over time, similar to the process of annealing in metallurgy.

Program:

Data Structures & Classes: Data structures and classes used in our project are mentioned below:

1. Optimization classes: AntColonyOptimization, Simulated annealing, ThreeOptForSA, ThreeOptSwapOptimization, and TwoOptSwapOptimization.
2. Entity class: Point
3. Preprocess class: CsvReader
4. Algorithm classes: Christofides, GreedyAlgorithm, and MinimumSpanningTree

AntColonyOptimization:
The Ant Colony Optimization (ACO) algorithm is a metaheuristic inspired by the behavior of ants searching for food, used to solve the Traveling Salesman Problem (TSP) and other optimization problems. A colony of artificial ants searches for good solutions by repeatedly choosing the next city to visit based on pheromone trail strength and a heuristic function. The algorithm deposits pheromone on the edges used by the ants, guiding future ants towards good solutions. However, parameter settings need to be fine-tuned for each problem instance.

Simulated annealing:
Simulated annealing (SA) is a metaheuristic algorithm for solving the Traveling Salesman Problem (TSP) that is inspired by the physical process of annealing. SA iteratively explores the solution space by randomly perturbing the current solution and accepting the perturbation based on a probability function. The temperature parameter is gradually decreased over time, allowing the algorithm to converge towards a good solution. SA is effective for large problem instances, but requires careful parameter tuning.

ThreeOptForSA:
ThreeOptForSA is an extension of the Simulated Annealing (SA) algorithm for the Traveling Salesman Problem (TSP) that combines SA with the Three-Opt local search heuristic. ThreeOptForSA explores the solution space more efficiently and converges towards a good solution faster than the basic SA algorithm, particularly for large problem instances. However, careful parameter tuning may still be required.

ThreeOptSwapOptimization:
The Three-Opt Swap Optimization algorithm is a heuristic algorithm for the Traveling Salesman Problem (TSP) that combines the Three-Opt and Swap algorithms to explore the solution space more efficiently and converge towards a good solution faster than using either algorithm alone. It has been shown to be effective for the TSP, particularly for large problem instances, but requires careful parameter tuning.

TwoOptSwapOptimization:
The Two-Opt Swap Optimization algorithm is a heuristic algorithm for the TSP that combines the Two-Opt and Swap algorithms to explore the solution space more efficiently and converge towards a good solution faster than using either algorithm alone. It has been shown to be effective, but may not be as effective as other heuristic algorithms and requires careful parameter tuning.

Algorithms:

Christofides:
The Christofides Algorithm is an approximation algorithm for solving the Traveling Salesman Problem (TSP). It guarantees a solution within a factor of 3/2 of the optimal solution. It works by finding the minimum spanning tree (MST), finding a minimum-weight perfect matching among the vertices with odd degree in the MST, and then combining the MST and the matching to form a multigraph. Finally, it finds an Eulerian circuit in the multigraph and converts it to a Hamiltonian circuit, which is a solution to the TSP.

GreedyAlgorithm:
The Greedy Algorithm is a simple and fast algorithm for solving the Traveling Salesman Problem (TSP). It starts at a random city and at each step, visits the nearest unvisited city until all cities have been visited. However, it does not guarantee an optimal solution and can give suboptimal results for large problem sizes.

MinimumSpanningTree:
The Minimum Spanning Tree (MST) algorithm is an approach for solving the Traveling Salesman Problem (TSP). It builds a complete graph on the cities, computes the MST using Prim's or Kruskal's algorithm, and traverses the MST to obtain a sequence of cities. Finally, it converts the sequence to a Hamiltonian circuit by adding the starting city at the end. While the MST algorithm is not guaranteed to produce an optimal solution, it can provide a good approximation in practice.