TSP

The travelling saleman problem. The gist of this is : Given a complete graph find a minimal/low cost tour that starts and ends in a given city and visits every other city exactly once.

The edge weights of the graph could represent miles, gallons of gas, or some other cost that we desire to minimize.

We might ask is there a path of cost k or less? to tuurn this into a desicion problem.

Notes:

The idea is to let you begin your research and design.

Adjustments

==============

Prepare a 3rd algorithm, original.

Prepare a document (paper) describing the results

Requirements:

You describe it using 3 algorithms:

1. Brute force Algorithm.
2. Nearest Neighbor Algorithm

Alright! Imagine you're planning a road trip to visit different cities. You want to find the shortest route that lets you visit each city exactly once and then return back home.

The Nearest Neighbor Algorithm works like this: You start from one city, let's call it your "home city". From there, you look for the closest city that you haven't visited yet. Once you find it, you travel to that city. Then, from this new city, you repeat the process: you find the closest city that you haven't visited, and you travel there.

You keep doing this until you've visited all the cities once. Once you've visited all the cities, you return back home. This forms a loop, where you start and end at your home city.

The idea is simple: at each step, you choose the nearest unvisited city. By doing this repeatedly, you hope to find a pretty good route that's not too long.

Now, keep in mind that this method might not always give you the absolute shortest route. Sometimes, it might miss a better route because it's only looking at the nearest city at each step. But for many situations, it's a quick and easy way to find a decent route for your road trip!

1. Your own Algorithm/idea that you built from scratch that may seem better that the previous two.

As we know nearest neighbor gives a guess but not the best/optimal solution.

You come up with a document of 10-20 pages, describing the idea of how yours is better and provide the codes and screenshots and testing.

You are expected to come up with a better idea even if not getting a real solution.

You can write it in ACM document format.

Find attached the Graphs for this task.

Title: Optimizing the Traveling Salesman Problem: A Comparative Analysis of Brute Force, Nearest Neighbor, and a Novel Algorithm

Abstract:

The Traveling Salesman Problem (TSP) is a classic optimization conundrum aiming to find the shortest path visiting all cities exactly once and returning to the origin city. This paper presents a comparative study of three algorithms for solving the TSP: Brute Force, Nearest Neighbor, and a novel algorithm developed for this research.

1. Introduction:

- Overview of the TSP and its significance.

- Explanation of the three algorithms under consideration.

2. Brute Force Algorithm:

- Description of the brute force approach.

- Complexity analysis and limitations.

- Implementation details and pseudocode.

- Performance evaluation and empirical results.

3. Nearest Neighbor Algorithm:

- Explanation of the nearest neighbor heuristic.

- Advantages and drawbacks compared to brute force.

- Implementation specifics and pseudocode.

- Experimental findings and comparative analysis.

4. Novel Algorithm:

- Proposal and rationale behind the novel algorithm.

- Unique features and theoretical underpinnings.

- Algorithmic design and efficiency considerations.

- Code implementation and integration.

5. Comparative Analysis:

- Evaluation metrics for comparing algorithms.

- Performance benchmarks on various datasets.

- Comparative study of solution quality and computational efficiency.

- Discussion on the strengths and weaknesses of each approach.

6. Results and Discussion:

- Presentation of experimental results.

- Comparison of solution quality and runtime performance.

- Analysis of algorithmic behavior under different scenarios.

- Insights into the practical applicability of each algorithm.

7. Conclusion:

- Summary of findings and contributions.

- Implications for practical TSP problem-solving.

- Future research directions and potential improvements.

8. References:

- Citations of relevant literature and resources.

9. Appendices:

- Source code snippets for each algorithm.

- Screenshots of experimental setup and results.

- Additional data and analysis supporting the main findings.

10. Acknowledgments:

- Recognition of individuals or organizations that contributed to the research.

This document provides a comprehensive exploration of the TSP and offers insights into the performance of three different algorithms for addressing this challenging problem. Through empirical analysis and theoretical discussion, it sheds light on the strengths and weaknesses of each approach, ultimately contributing to the ongoing discourse on optimization algorithms and their practical applications.

Let's consider two cities from the data: City A and City B.

For example, let's say City A is the starting city, and City B is the second city visited.

Here's how to interpret the data in layman's terms:

- The first row represents the distances from City A to each subsequent city, including City B. So, if the data for City A's row is "3032 0", it means that the distance from City A to City B is 3032 units, and the distance from City A back to itself (completing the tour) is 0 units.

- The second row represents the distances from City B to each subsequent city after it. The value "25282" indicates the distance from City B to the next city, and "10805 0" means the distance from City B back to itself (completing the tour) is 10805 units.

In simpler terms, each row in the data provides the distances from one city to the cities that come after it, and the last number in each row indicates the distance back to the starting city, completing the tour.

This interpretation helps understand the structure of the data and how it relates to the Traveling Salesman Problem, where finding the shortest route that visits each city exactly once and returns to the starting city is the objective.