Assignment-3: Travelling Salesman Problem (TSP) by Local Search

1. Introduction

The Travelling Salesman Problem (TSP) models the challenge of a salesperson who must visit multiple cities. The objective is to find the shortest possible route that visits each city exactly once and returns to the starting point. In its general form, TSP is represented by a directed, edge-weighted graph. Here, the goal is to identify a minimal cyclic path covering all nodes. Typically, TSP is defined for complete graphs. In complete graphs, each pair of vertices has a connecting edge. For this assignment, we assume all TSP instances are complete graphs. Each vertex represents a point in Euclidean space, and each edge weight is the Euclidean distance between two points.

Local search is a common approach to solving TSP. In local search, we start at a point in the search space and move iteratively to neighboring positions. Local search methods can be categorized as constructive or perturbative. Constructive search operates in a search space of partial candidate solutions, adding components step-by-step. Perturbative search starts with a complete candidate solution and modifies it iteratively. Constructive search builds solutions from scratch. It begins with an empty solution and adds components until the solution is complete. Perturbative search, in contrast, improves upon existing solutions by altering one or more components. Although “local search” generally refers to perturbative methods, it also applies to constructive methods. Typically, perturbative search depends on initial solutions generated by constructive search heuristics. In this assignment, we explore TSP using both constructive and perturbative search techniques. The aim is to find efficient heuristics for obtaining near-optimal solutions in TSP instances.

2. Methodology

In this assignment, we implement three heuristics from both constructive and perturbative search methods to tackle the Travelling Salesman Problem (TSP). We then test these methods on 5 benchmark datasets and examine the performance of various heuristic combinations. The constructive search methods focus on building initial solutions, while the perturbative search methods aim to improve upon these initial solutions through iterative refinement for each constructive methods.

Constructive Search Methods

Constructive search methods build a tour from scratch by gradually adding vertices in a way that aims to minimize the total tour length. The three constructive heuristics we use are:

2.1 Nearest Neighbour Heuristic (NNH)

- We start by choosing a random initial vertex

- We then iteratively add the nearest unvisited neighbor to the current partial tour by selecting the edge with the smallest distance to the current vertex.

- This process continues until all vertices are included in the tour, at which point we return to the starting vertex to complete the tour.

- We also experiment with a semi-greedy version of NNH, where instead of choosing the absolute nearest neighbor, we randomly select a vertex from the top 3 nearest vertices.

2.2 Insertion Heuristics

We explore three variations of insertion heuristics, where the tour is extended by inserting a vertex in a position that minimizes the increase in tour length:

- **Nearest Insertion:** Insert the unvisited vertex that has the minimum distance to any vertex in the current partial tour.

- **Cheapest Insertion:** Insert a vertex that leads to the minimal increase in tour length over all unvisited vertices.

- **Farthest Insertion:** Insert the vertex with the maximum minimum distance to any vertex in the current partial tour.

2.3 Minimum Spanning Tree (MST) Based Heuristic

- We first compute a Minimum Spanning Tree (MST) of the graph using an algorithm like Prim's

- Once the MST is constructed, we perform a Depth-First Search (DFS) traversal to obtain a Hamiltonian path.

- To complete the tour, we return to the starting vertex, producing an approximate solution for the TSP.

2.4 Christofides Heuristic

- This heuristic begins by constructing an MST of the graph.

- Next, it finds a minimum-weight matching on the set of vertices with odd degrees in the MST to create an Eulerian circuit.

- Finally, the Eulerian circuit is converted into a Hamiltonian tour by skipping repeated vertices, yielding a feasible TSP solution with a performance guarantee of 1.5 times the optimal solution

Perturbative Search Methods

Perturbative search methods begin with a complete candidate solution and iteratively improve it by exploring neighboring solutions. We focus on the following perturbative heuristics:

2.1 2-opt

- The 2-opt algorithm iteratively removes two edges from the tour and reconnects the segments in the optimal way, eliminating any crossover.

- This process reduces the total tour length and is repeated until no further improvement is possible.

2.2 3-opt

- The 3-opt algorithm extends 2-opt by considering removing three edges and reconnecting the segments in an optimal way.

- This increases the scope of improvements and can yield better results than 2-opt but with a higher computational cost.

2.3 Node Swap

- Swaps the positions of two cities in the tour, generating a neighboring solution that may improve the tour length.

3. Experimental Setup

3.1 Constructive Methods as Initial Solutions:

- Use constructive algorithms, such as Nearest Neighbor (NNH), Insertion Hueristics, Minimum Spanning Tree (MST), Christofides' algorithm, to generate initial tours for each dataset.

3.2 Perturbative Optimization:

- Use the initial tour as a starting point for perturbative methods, Local Search (2-opt, 3-opt) and node swap.

3.3 Evaluation Across Benchmark Datasets:

- Choose 5 benchmark TSP datasets with varying sizes and complexities.

- Apply each constructive and perturbative method to each dataset, using the initial tour generated by the constructive method.

3.4 Calculate TSP Costs:

- For each solution found by the perturbative method on each dataset, calculate the TSP cost (total distance or cost of the tour).

- Store these results for further analysis.

3.5 Mean Cost Calculation:

- For each benchmark dataset, calculate the mean TSP cost across multiple runs .

3.6 Performance Ratio:

- To evaluate each method’s efficiency, calculate the ratio of the mean TSP cost obtained to the known optimal or best-known solution for each dataset.

4. Results

Table 1 presents the performance comparison of various heuristics on five benchmark TSP datasets, assessing mean TSP cost and relative effectiveness. The Christofides heuristic emerges as the most effective, with the lowest mean cost (12,474.6) and a baseline performance ratio of 1, serving as the standard for comparison. Farthest insertion and Nearest Neighbor Heuristic (NNH) yield higher mean costs, with ratios of 1.4 and 1.5, indicating less efficiency relative to Christofides. Methods like Cheapest Insertion, MST, and Nearest Insertion perform moderately well, with performance ratios between 1.2 and 1.3, suggesting these are reasonably effective alternatives but still less optimal than Christofides.

Methods	**st70**	**kroE100**	**rat195**	**a280**	**lin318**	**Mean**	**Ratio**
Farthest insertion	805.5	27587.1	2761.9	3148.1	54033.5	17667.2	1.4
Christofides heuristic	567.2	19246.5	2171.1	2438.5	37949.8	12474.6	1
Cheapest insertion	779.7	25361.1	2783.0	3029.2	49454.8	16281.6	1.3
MST	714.3	23244.1	2623.5	2914.6	45411.8	14981.7	1.2
NNH	873.3	30507.4	3317.7	3616.2	58145.4	19292	1.5
Nearest insertion	804.8	27236.8	2740.8	2988.9	48993.1	16552.9	1.3

Table 1: Constructive search methods

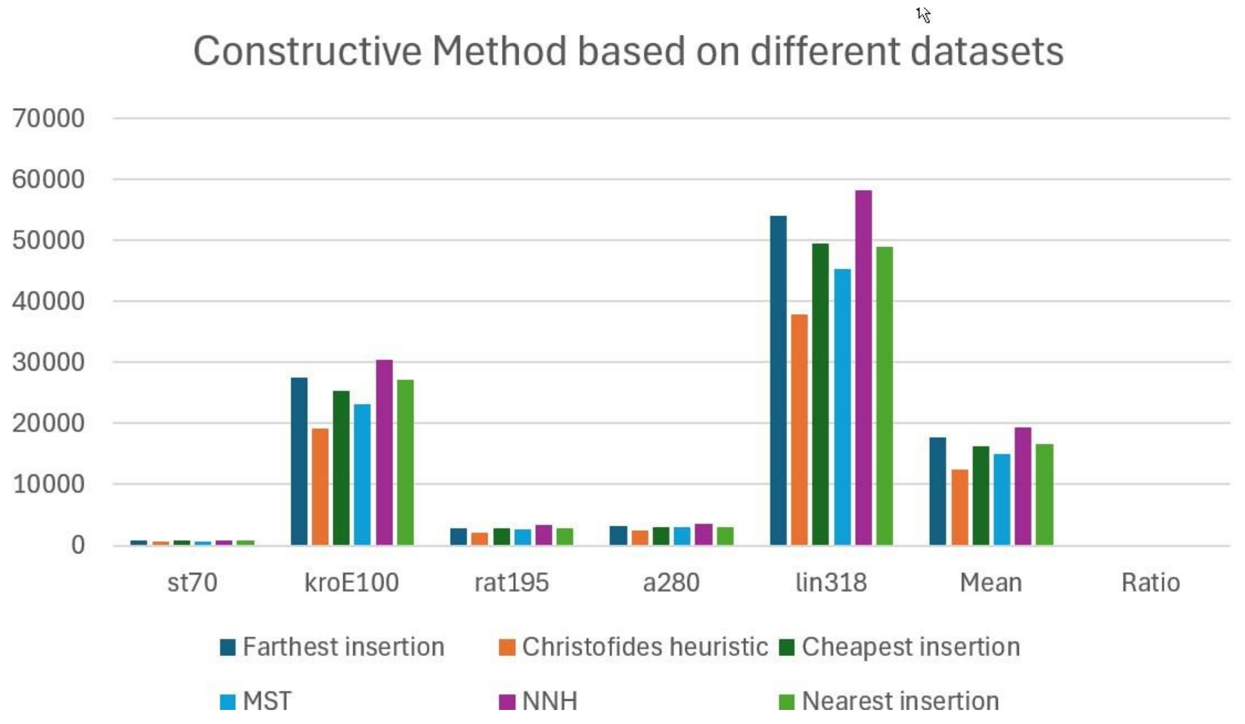

Figure 1: Constructive Method based on different datasets

Table 2 highlights how each initial tour method combined with 2-OPT, 3-OPT, Node Swap performed on the datasets.

Initial Tour Method	**Method**	**Dataset**	**Ratio**
NNH	2-OPT	708.3	1.03
NNH	3-OPT	702.1	1.02
NNH	Node Swap	1268.9	1.84
Nearest insertion	2-OPT	725.3	1.06
Nearest insertion	3-OPT	721.6	1.05
Nearest insertion	Node Swap	857.1	1.24
Cheapest insertion	2-OPT	754.2	1.10
Cheapest insertion	3-OPT	721.9	1.05
Cheapest insertion	Node Swap	771.2	1.12
Farthest insertion	2-OPT	710.0	1.03
Farthest insertion	3-OPT	710.0	1.03
Farthest insertion	Node Swap	710.6	1.04
MST	2-OPT	698.2	1.02
MST	3-OPT	695.6	1.01
MST	Node Swap	851.3	1.24
Christofides heuristic	2-OPT	717.3	1.04

Christofides heuristic	3-OPT	687.4	1.00
Christofides heuristic	Node Swap	794.6	1.16

Table 2: Perturbative search methods

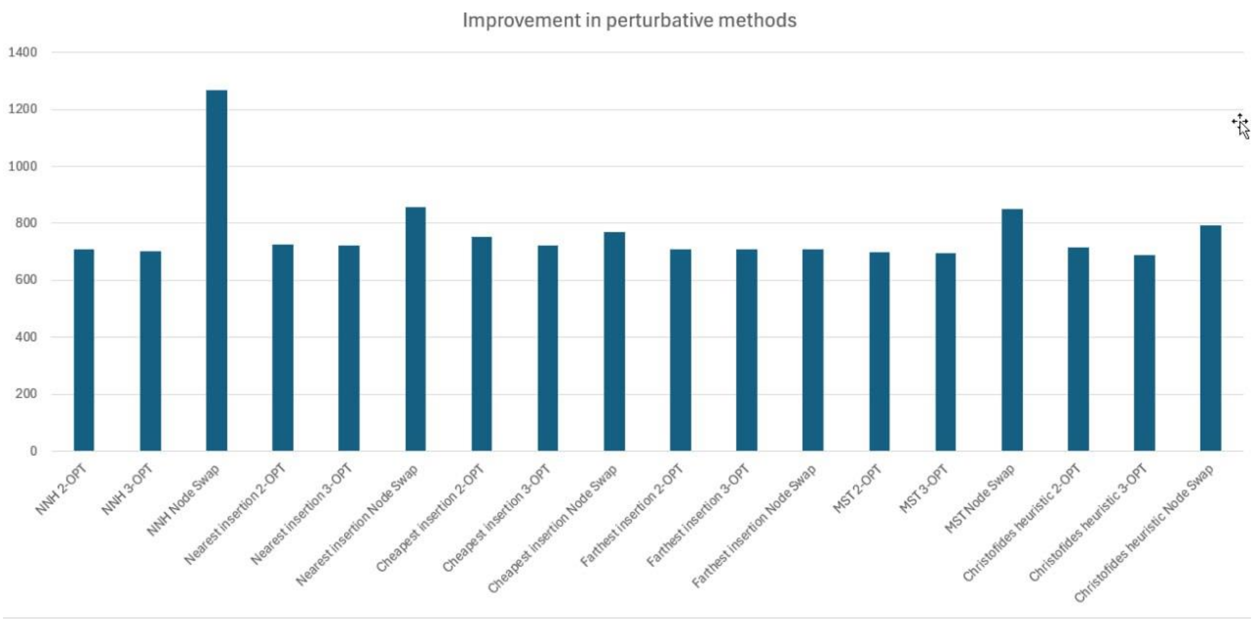

Figure: Perturbative search methods

5. Discussion

Performance comparison in Constructive search methods :

Nearest Neighbor Heuristic (NNH) < Farthest Insertion < Nearest Insertion < Cheapest Insertion < Minimum Spanning Tree (MST) Based Heuristic < Christofides Heuristic

Best Performance:

Christofides heuristic

Performance comparison in Constructive search methods :

Node Swap < 2-OPT < 3-OPT

Best Performance:

3-OPT

In comparing the constructive search methods, the **Christofides heuristic** consistently provides the best performance, achieving the lowest TSP cost across datasets. This is due to its ability to find near-optimal solutions by leveraging a minimum spanning tree and perfect matching, making it more efficient than other methods. The MST-based heuristic also performs well but lacks the **added matching step** that refines the solution. In the MST, some nodes will end up with an odd degree (an odd number of connections), which means we can't directly turn the MST into a tour. The Christofides heuristic identifies all nodes with an odd degree and pairs them up by adding extra edges to create a *Eulerian graph* (a graph where each node has an even degree).

Among insertion methods, **Cheapest Insertion** outperforms Nearest and Farthest Insertions, as it strategically minimizes incremental costs at each step. Nearest Neighbor Heuristic (NNH) ranks lowest as it often gets trapped in locally optimal tours without considering the broader structure.

In comparing local search methods for improving TSP solutions, we see a clear progression in effectiveness from Node Swap to 2-OPT to 3-OPT. The Node Swap technique is the simplest, where pairs of nodes are swapped to try to reduce the total tour cost. However, this approach only allows for very localized changes, so it’s limited in how much it can improve a solution since it can easily get trapped in suboptimal configurations.

The 2-OPT method is more powerful. Rather than just swapping individual nodes, it involves breaking two edges in the tour and re-connecting the nodes in a different order**, which can eliminate crossovers in the path.** This allows 2-OPT to explore a broader set of potential solutions and leads to more substantial improvements in tour cost, though it can still miss more complex optimizations.

The 3-OPT method goes a step further, removing three edges at a time and reconnecting them in a way that can yield even greater reductions in the tour length. By enabling more complex rearrangements, 3-OPT can escape local minima that 2-OPT might settle into, resulting in a more globally optimized solution. When paired with an effective initial solution, like the one from the Christofides heuristic, 3-OPT achieves the best performance as it has the highest potential to refine the solution to near-optimal quality by making larger, yet well-guided adjustments.

5. Conclusion

In conclusion, the comparison of various constructive and perturbative search methods for solving the Travelling Salesman Problem (TSP) reveals that the Christofides heuristic consistently yields the most effective solutions, with the lowest TSP cost across all datasets. Among perturbative methods, 3-OPT proves to be the most efficient, significantly improving initial solutions. The combination of Christofides’ heuristic with 3-OPT delivers near-optimal results, showcasing the strength of hybrid approaches in solving complex optimization problems like TSP. Overall, these findings highlight the importance of selecting both efficient initial solutions and powerful local search methods for achieving optimal TSP solutions.