CSE 318 - Assignment 3

Anik Saha - 2005001

November 17, 2024

1 Introduction

Travelling Salesman Problem (TSP) is a famous problem of computer science. It asks for a circular tour of minimum cost through n cities so that the tour visits every city exactly once.

This report summarizes the results of running various constructive and perturbative heuristics on the Traveling Salesman Problem (TSP) for the **a280.tsp, berlin52.tsp, bier127.tsp, ch130.tsp, ch150.tsp, eil101.tsp, eil51.tsp, eil76.tsp, kroA100.tsp, kroB100.tsp, kroC100.tsp, kroD100.tsp, kroE100.tsp, lin105.tsp, lin318.tsp, pr124.tsp, pr144.tsp, pr76.tsp, rat195.tsp, rat99.tsp, st70.tsp** problem instances.

I conduct experiments on 6 constructive and 3 perturbative heuristic algorithms. I run the algorithm combinations on the 21 files for 10 iterations each and report the aggreagate results and analysis over the output.

2 Heuristics Used

2.1 Constructive Heuristics

- **NearestNeighbour**
- **SemiGreedyNearestNeighbour**
- **CheapestInsertion**
- **SemiGreedyCheapestInsertion**
- **SemiGreedyRandomInsertion**
- **MSTSimple**

2.2 Perturbative Heuristics

- **TwoOpt**
- **NodeSwap**
- **NodeShift**

3 Results

In the following tables and figures, the relative performances of the constructive and perturbative algorithms are analyzed according to mean values over 10 test runs for each combinations.

Heuristic Combination	Win Count
NearestNeighbour+TwoOpt	12
MSTSimple+TwoOpt	5
CheapestInsertion+NodeShift	2
SemiGreedyNearestNeighbour+TwoOpt	2

Table 1: Win Counts by Heuristic Combination

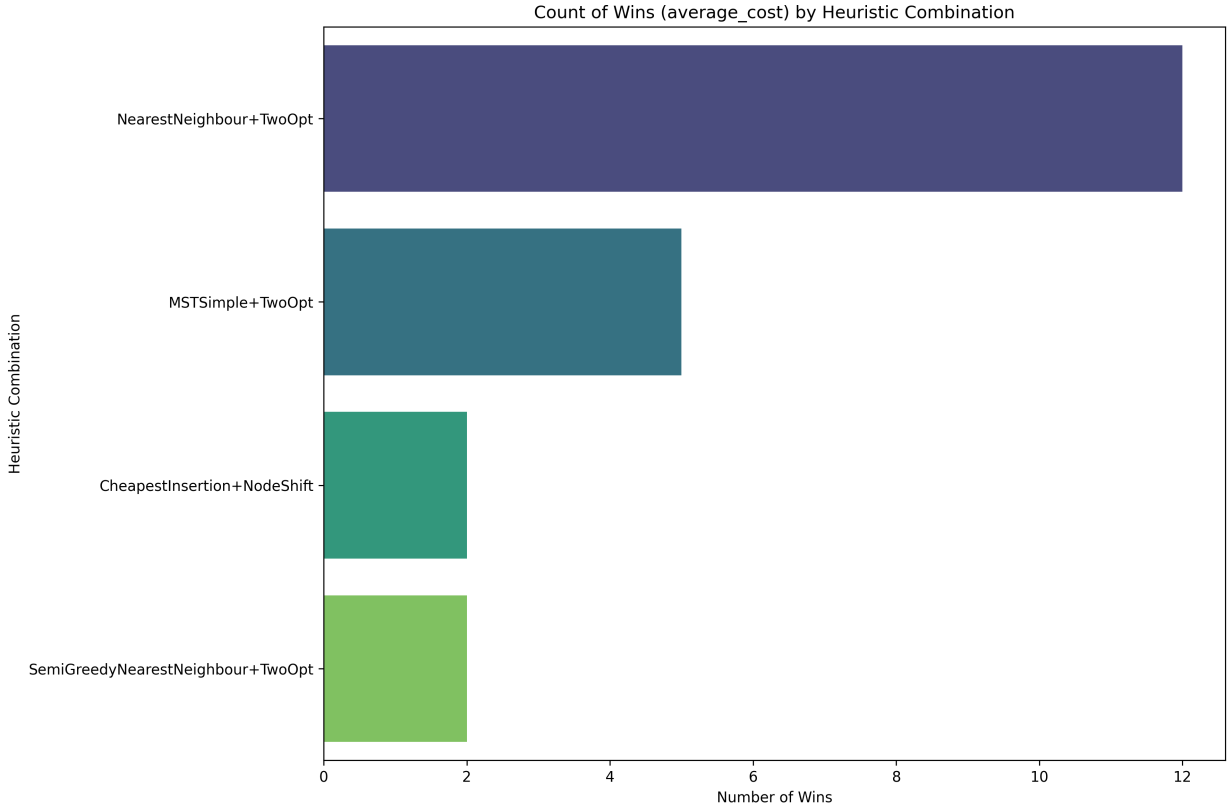

Figure 1: Count of Wins (average wins) by Heuristic Combination

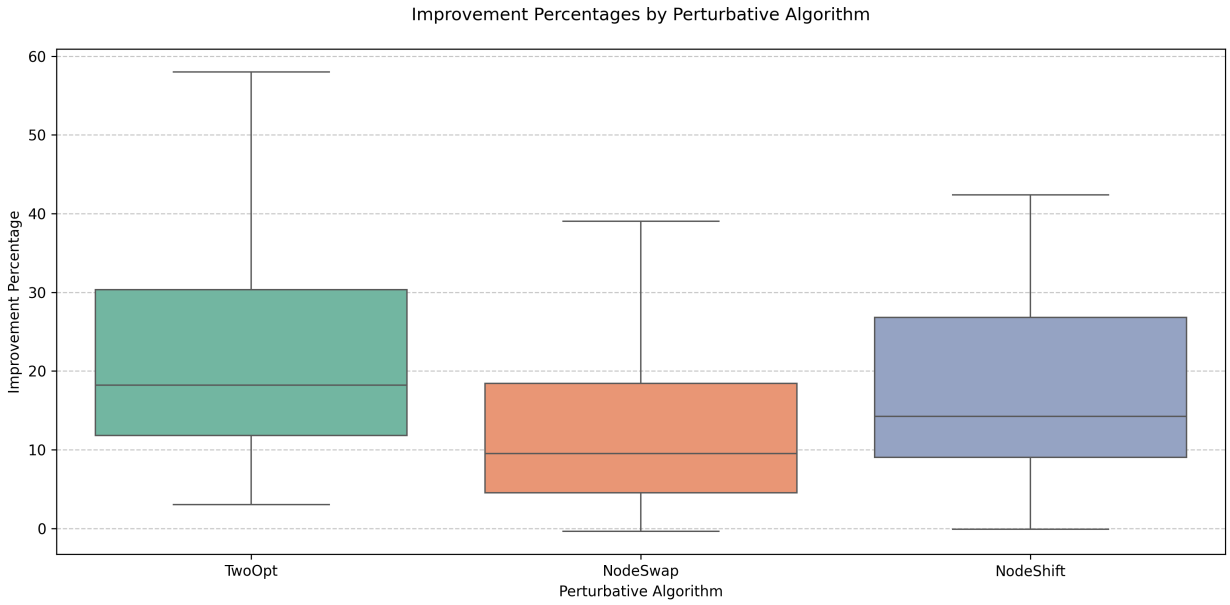

Figure 2: Improvement Percentages by Perturbative Algorithm

Constructive	Perturbative	Normalized Range
CheapestInsertion	—	0.0465
CheapestInsertion	NodeShift	0.0432
CheapestInsertion	NodeSwap	0.0423
CheapestInsertion	TwoOpt	0.0526
MSTSimple	—	0.0447
MSTSimple	NodeShift	0.0545
MSTSimple	NodeSwap	0.0583
MSTSimple	TwoOpt	0.0525
NearestNeighbour	—	0.1252
NearestNeighbour	NodeShift	0.0973
NearestNeighbour	NodeSwap	0.1246
NearestNeighbour	TwoOpt	0.0636
SemiGreedyCheapestInsertion	—	0.0863
SemiGreedyCheapestInsertion	NodeShift	0.0756
SemiGreedyCheapestInsertion	NodeSwap	0.0781
SemiGreedyCheapestInsertion	TwoOpt	0.0708
SemiGreedyNearestNeighbour	—	0.2118
SemiGreedyNearestNeighbour	NodeShift	0.1495
SemiGreedyNearestNeighbour	NodeSwap	0.1680
SemiGreedyNearestNeighbour	TwoOpt	0.0769
SemiGreedyRandomInsertion	—	0.0903
SemiGreedyRandomInsertion	NodeShift	0.1076
SemiGreedyRandomInsertion	NodeSwap	0.1159
SemiGreedyRandomInsertion	TwoOpt	0.0749

Table 2: Algorithm Combination Consistency Analysis

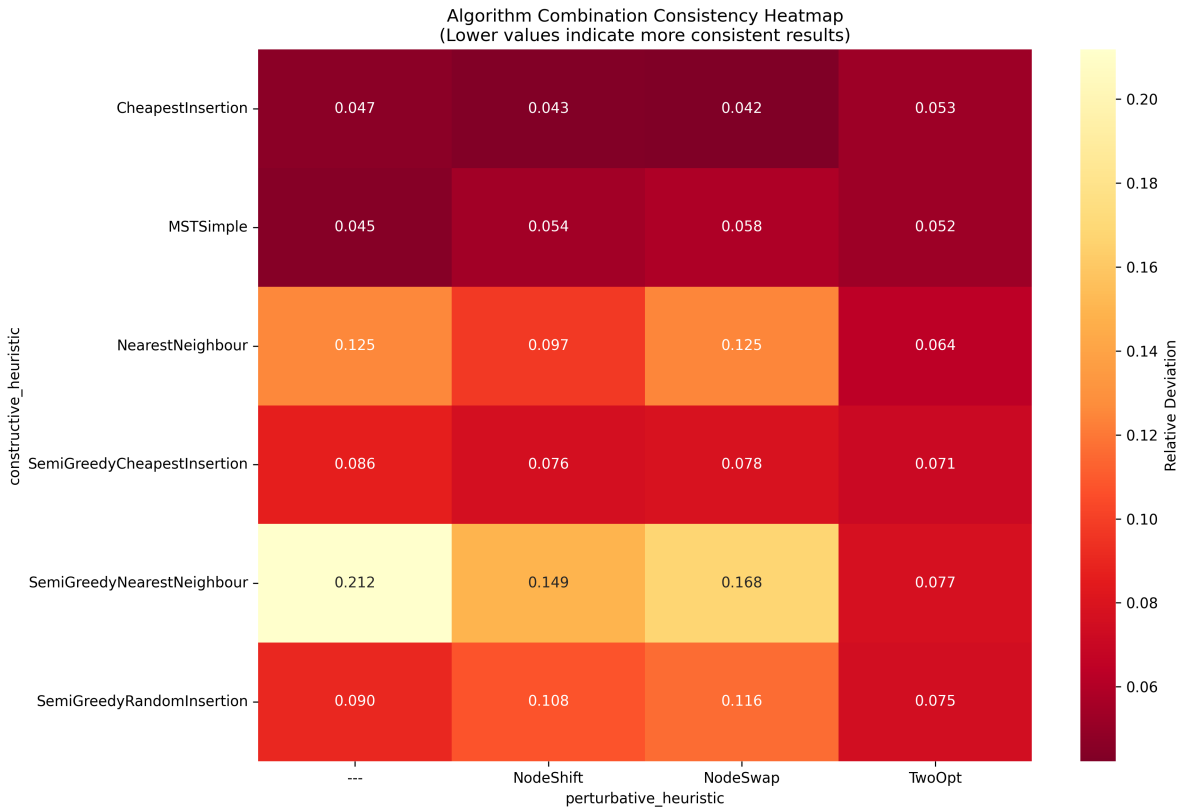

Figure 3: Algorithm Combination Consistency Heatmap (Lower values indicate more consistent results)

4 Discussions

I present the discussions over the results based on three major criteria, namely, number of wins on specific problem instances, improvement percentages, and consistency of algorithm results over multiple runs.

4.1 Number of Wins

For each specific TSP instance, I determine the algorithm combination that results in the minimum average cost for that instance. It is evident that **NearestNeighbour+TwoOpt** performs best with 12 wins while the second place is held by **MSTSimple+TwoOpt** with 5 wins. **CheapestInsertion+NodeShift** and **SemiGreedyNearestNeighbour+TwoOpt** each wins 2 times jointly being the third best combination.

4.2 Improvement Percentages

The relative improvements of cost after applying a perturbative algorithm are compared for each of the 3 perturbative method implemented. **TwoOpt** provides an improvement in the range of 7% to 60% with an average of around 18% which is higher than the other two heuristics.

4.3 Performance Consistency

Among the 10 iterations for each combination, the deviation of results are compared. From the heatmap, **CheapestInsertion** proves to be the most consistent among the constructive heuristics. However, the semi-greedy variants of the algorithms seem to be less consistent, which is explainable because they further randomize the process by selecting an arbitrary one among the heuristically promising choices. Among the perturbative algorithms, **TwoOpt** seems to the most consistent performer.