The Long Road: Comparison of Approximation Heuristics for Traveling Salesperson Problem

Artem Yushko, Arthur Viegas Eguia, Daniel Chen, Miles Hoene-Langdon, and Layla Oesper

BACKGROUND

Traveling Salesperson Problem

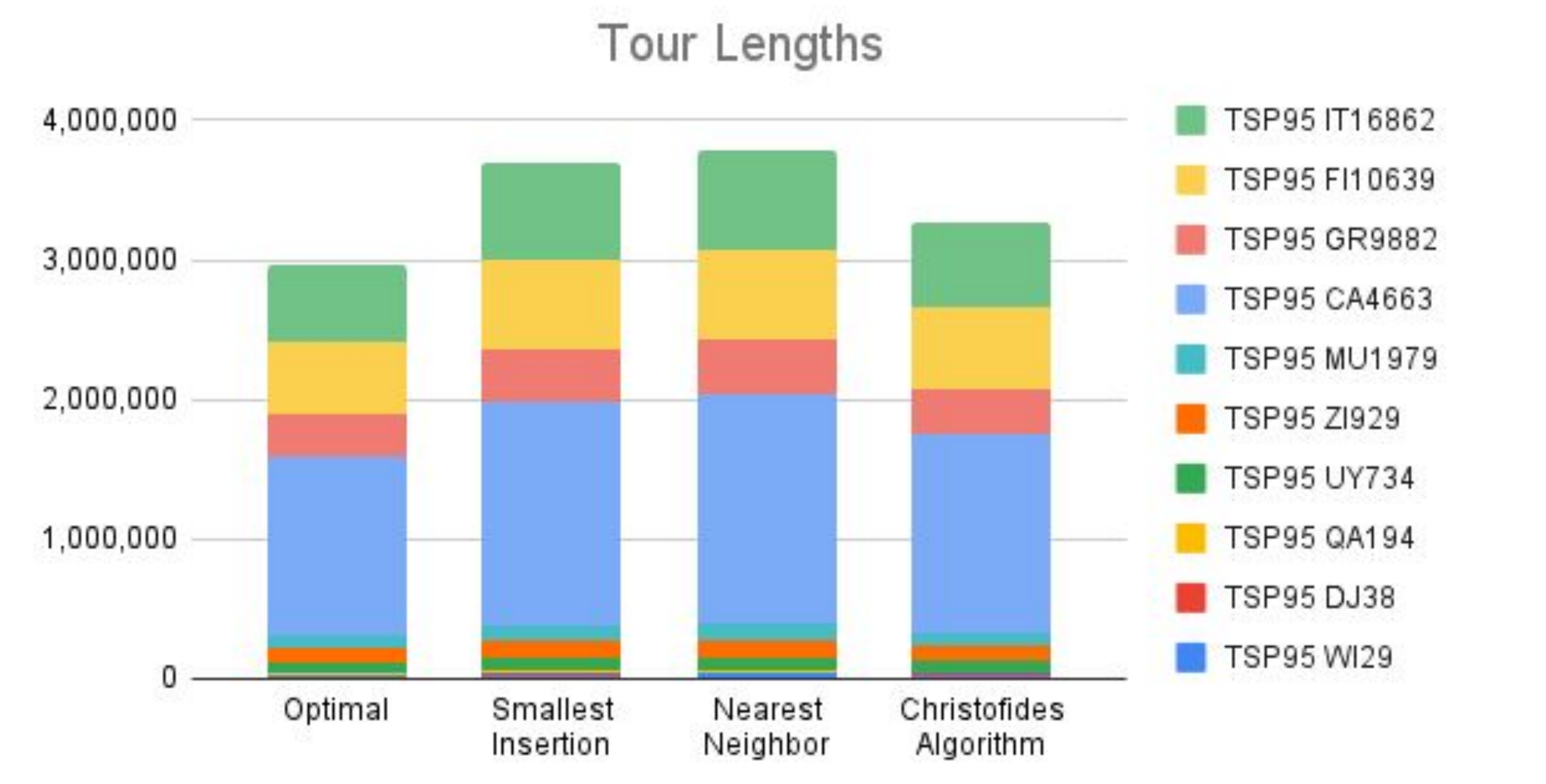
- The traveling salesperson problem (TSP) poses the question "Given a **list** of cities and the **distance** between each pair of cities, what is the **shortest route** that visits each city exactly once and returns to the origin city?" (REF).
- TSP is an **NP-hard** problem, therefore there is no efficient known way of solving it. At the same time, it is also an important problem in the field of **optimization**, meaning that it is used as a benchmark for many optimization methods.
- For the purposes of our project, we define the constraints as:
  - Input: a set of points **S** = (x, y), where each point in the set represents the coordinates of a city.
  - Output: an ordered set **G** = (p1, p2, p3, ..., pn) where each p is a city, and the total length of G is **minimized**.

Approximation Algorithms

- **Nearest Neighbor**: begin at an arbitrary node and add the closest unvisited city from the city where we are right now. Do the same thing for this one and for each subsequent city after it.
- **Smallest Insertion**: begin at two arbitrary nodes. Then, iterate through the nodes to find the one that increases the tour the least. Do the same thing until we visit all the nodes.
- **Christofides Algorithm**: close to the state-of-the-art approximation of this problem. Simplifies the problem to finding a Eulerian Path and then a Hamiltonian Path, creating a tour.

METHODOLOGY

- **Datasets:** 10 datasets taken from the TSP95 collection with the sizes ranging between 29 and 16,862 cities, each one representing a real country. **KEY: TSP95** (Traveling Salesperson Problem 95) **XX** (Country Code) **123** (Number of Nodes)
- **Testing Procedure:** The algorithms were ran on one of the Computational Research Users Group Cluster computers owned by Carleton. The machine we used, Ada, had 28 cores and 377 GB of RAM, making it the most powerful machine owned by Carleton.



RESEARCH QUESTION

What are the advantages and disadvantages of different heuristics used for approximating the TSP?

What kinds of variation can be seen in the results?

SMALLEST INSERTION ALGORITHM

Procedure & Implementation

- The algorithm could be described as follows:
  - Pick two arbitrary nodes
  - While there are unvisited nodes left:
    - Check if a node leads to the smallest increase of the tour length.
    - If so, add it to the tour.
    - If not, keep looking.

Results

- Smallest Insertion was consistently the **fastest** approximation algorithm out of the ones we have tried.
- Its results were worse than that of Christofides, but on average just **1.25** times **bigger** than those of the optimal tour.
- The following images present the map of Zimbabwe (Figure 3), the optimal tour generated by a brute force algorithm (Figure 2), and the tour outputted by the smallest insertion algorithm (Figure 1).

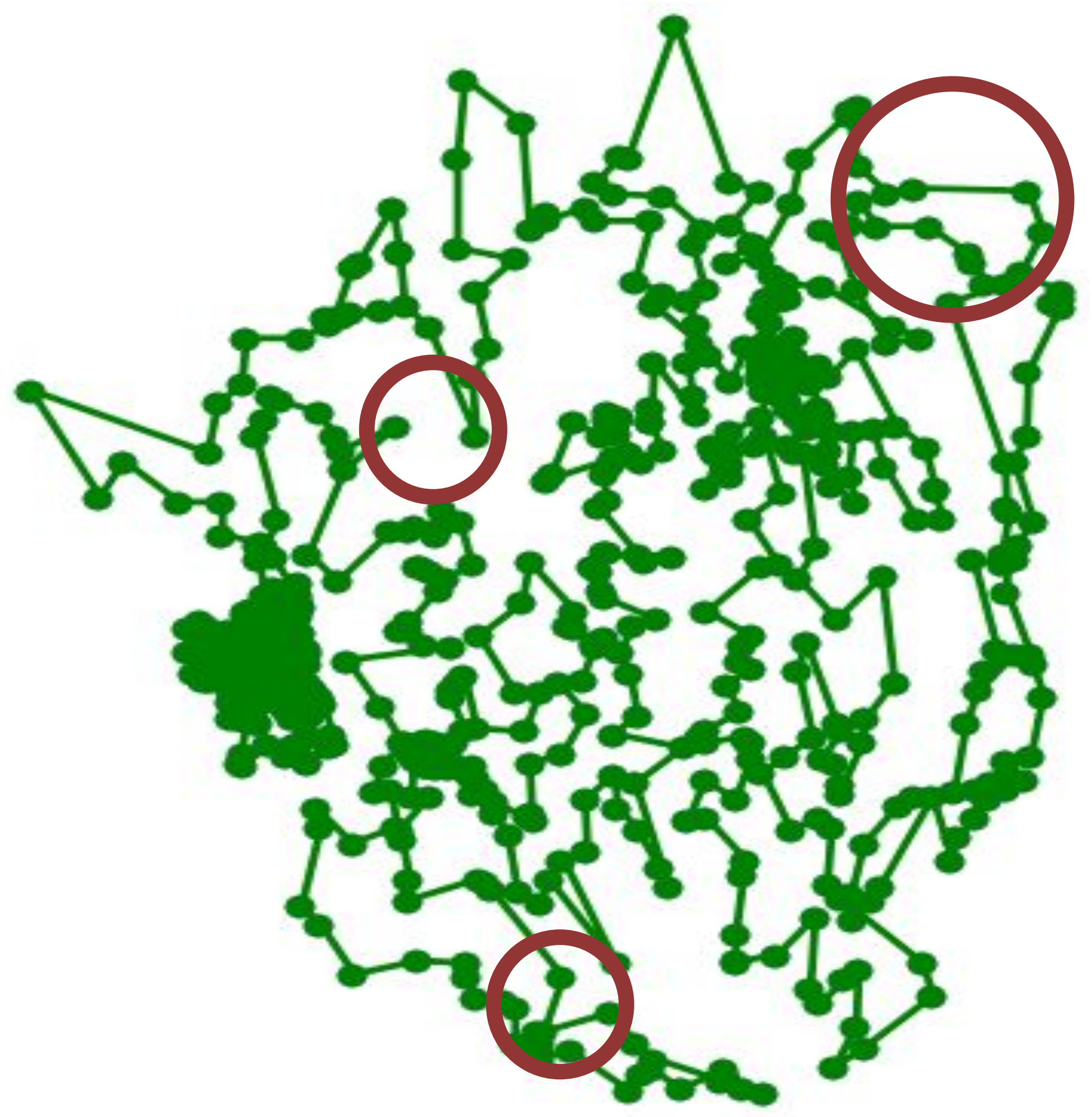
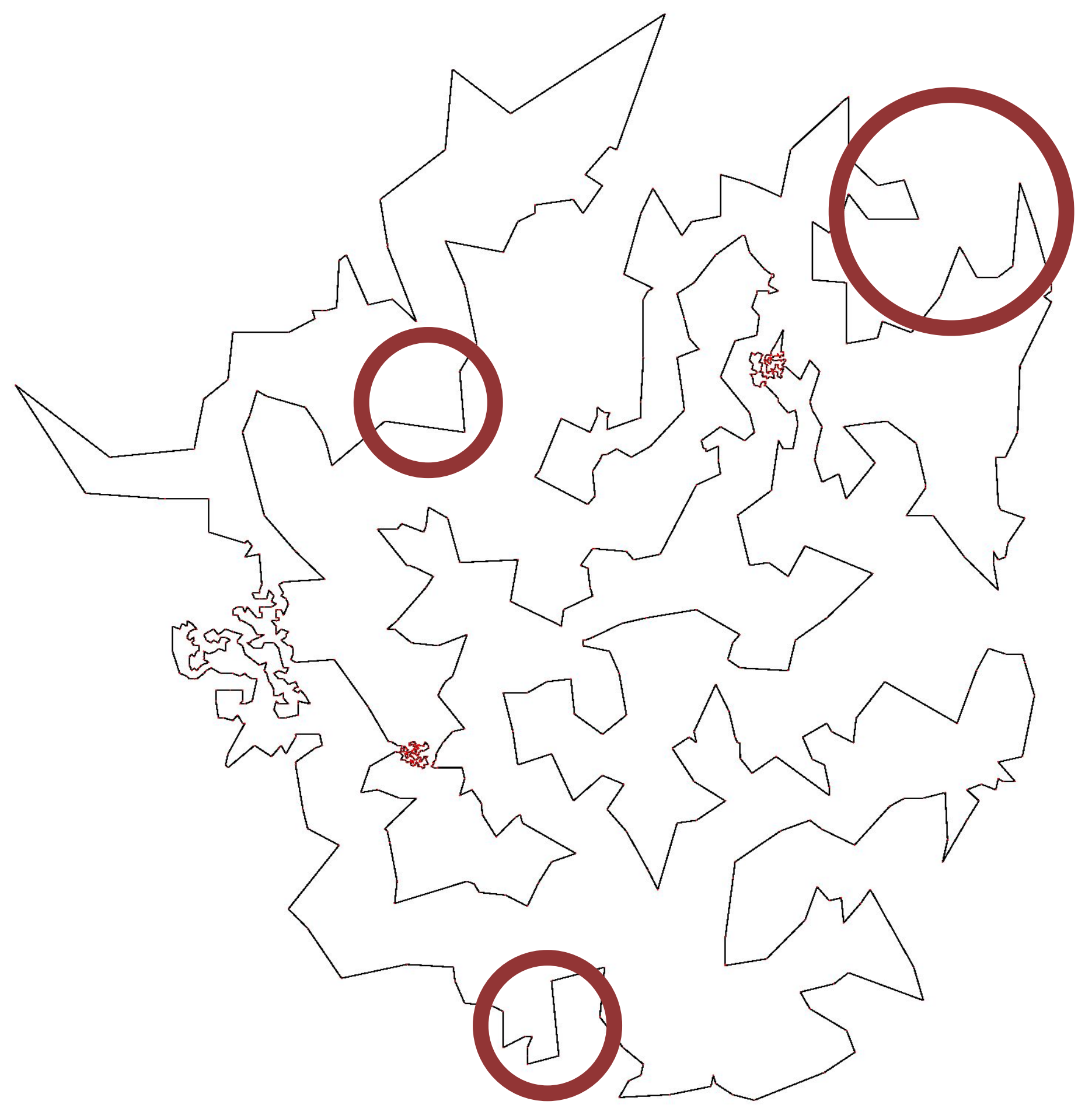


Figure 1: Smallest Insertion Tour

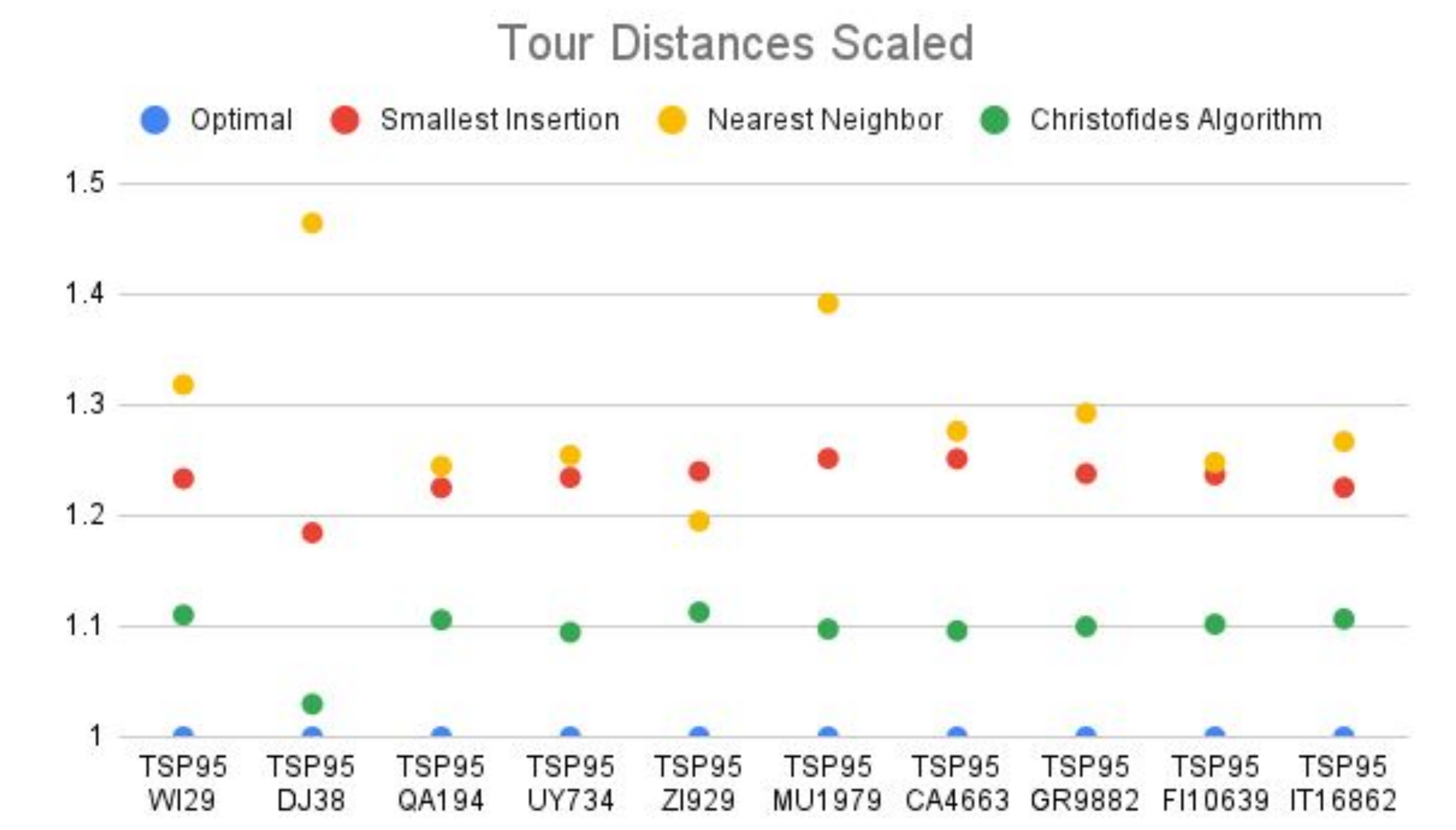
Areas with different decisions made by these two algorithms are circled in red

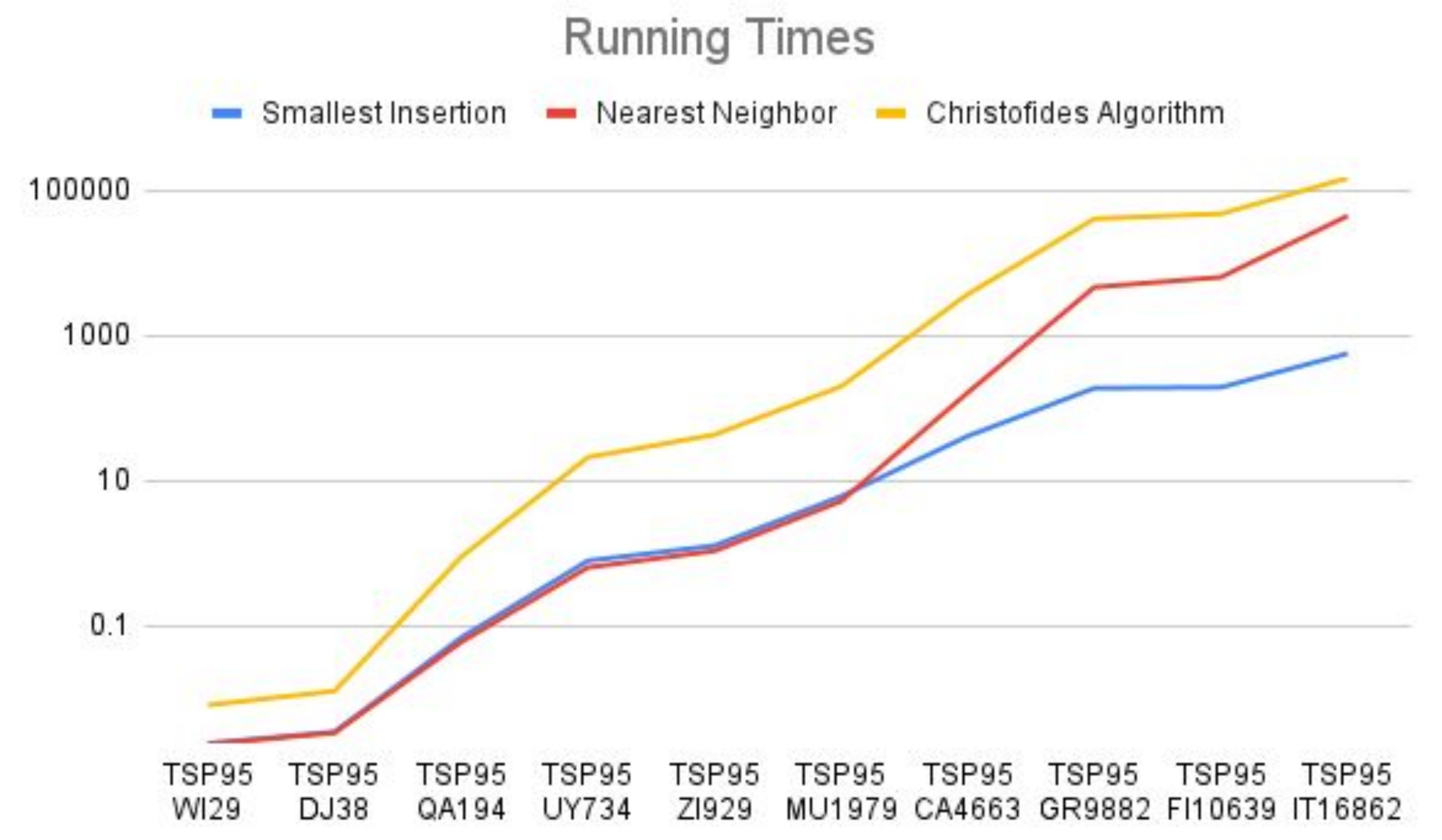
Figure 3: Map of Zimbabwe

Figure 2: Optimal Brute-Force Tour



FINDINGS





DISCUSSION

- The theoretical runtime of Nearest Neighbor is bound by $O(n^2\log(n))$, and the theoretical runtime of Christofides is $O(n^3)$, but the actual runtimes are not following the proportions directly.
- While the theoretical bound of Christofides Algorithm is 1.5 times that of the original, it barely went beyond 1.1 during our research.
- At the same time, the Smallest Insertion and Nearest Neighbor algorithms, while having 2n and n*log(n) worst-case distances, stay pretty close to each other within the 1.2 and 1.5 bounds.
- Our research was constrained by the NetworkX library, which led to our algorithms using 50 GB of RAM throughout the execution. Therefore, any further research would require powerful computation units.

ACKNOWLEDGEMENTS

Thanks are due to Layla Oesper for her guidance throughout the research process. We would also like to thank Mike Tie for getting us access to the Ada Computational Cluster for our research.