A Comparative Study of Unsupervised GNN-based Solvers for the Travelling Salesman Problem

Joffin Koshy *Date: June 5, 2025*

Abstract

The Travelling Salesman Problem (TSP) is a canonical NP-hard problem in combinatorial optimization with wide-ranging applications. Traditional methods struggle with large-scale instances, motivating the exploration of machine learning-based heuristics. This report details the replication and extension of an unsupervised learning framework for the TSP, which uses a Graph Neural Network (GNN) to generate solutions without requiring labeled training data. We implemented a GAT-based model and designed a novel alternative surrogate loss function incorporating the graph Laplacian's Fiedler value to explicitly enforce tour connectivity. We conducted a rigorous experimental comparison between this modified approach and a baseline model across TSP instances of N=20, 50, 100, 200, and 500 cities. Our findings indicate that the alternative loss function is highly competitive, performing marginally better than the baseline for medium-sized problems (N=50, 100). Conversely, the simpler baseline loss function demonstrated superior performance on larger instances (N=200, 500), highlighting a critical trade-off between loss function complexity and generalization at scale. This work validates the effectiveness of GNNs in this domain and provides valuable insights into the design of surrogate loss functions for unsupervised combinatorial optimization.

1. Introduction

The Travelling Salesman Problem (TSP) asks for the shortest possible route that visits a given set of cities and returns to the origin city. Despite its simple definition, the TSP is NP-hard, meaning the computational cost of finding an exact optimal solution grows factorially with the number of cities, making it intractable for large instances. This has led to the development of a wide array of heuristic methods that aim to find near-optimal solutions in a reasonable amount of time.

In recent years, machine learning, particularly deep learning on graphs, has emerged as a promising new paradigm for tackling such problems. This project focuses on replicating and extending the work of Joshi et al. in "An Unsupervised Learning Framework for Combinatorial Optimization with Applications to the Traveling Salesman Problem" (UTSP). The core idea of this framework is to train a Graph Neural Network (GNN) to predict an edge probability "heatmap" using a cleverly designed surrogate loss function, thereby learning to solve the TSP without access to any pre-computed

optimal tours for training.

This project had three primary objectives:

1. To replicate the UTSP framework, using a Graph Attention Network (GAT) as the core GNN architecture.
2. To design and implement a novel alternative surrogate loss function that introduces a spectral graph theory component to more explicitly enforce tour connectivity.
3. To perform a rigorous experimental comparison of the modified approach against the baseline across multiple problem sizes (N=20, 50, 100, 200, 500) to evaluate its performance and scalability.

2. Methodology and Implementation

This section details the technical components of our unsupervised learning framework. The approach involves three main stages: (1) using a Graph Attention Network (GAT) to generate an edge probability heatmap, (2) training the GNN using a surrogate loss function, and (3) constructing a final tour from the heatmap.

2.1. GNN Architecture: Graph Attention Network (GAT)

To learn the complex spatial relationships between cities, we selected a Graph Attention Network (GAT) as our core architecture. Unlike simpler graph convolutions, GAT assigns different importance (attention weights) to different nodes in a neighborhood, allowing it to capture more nuanced relationships.

Our model processes a complete graph where each city is a node. The architecture consists of three main parts:

1. **Input Embedding:** An initial linear layer embeds the 2D coordinates of each city into a higher-dimensional feature vector (64 dimensions in our experiments).
2. **GAT Layers:** Two GAT layers iteratively update each node's representation by attending over its neighbors. This allows information to propagate across the graph, enabling each node's final embedding to contain information about the overall structure of the TSP instance.
3. **Heatmap Prediction:** The final node embeddings are used to predict the probability of an edge existing between every pair of nodes. For each pair of nodes (i, j), their final embeddings are concatenated and passed through a final multi-layer perceptron (MLP) with a sigmoid activation function to produce a single value in [0, 1]. This value, P(i, j), represents the model's predicted probability that the edge from city i to city j is part of the optimal tour. The

collection of these probabilities forms an N x N adjacency matrix, or "heatmap".

2.2. Unsupervised Surrogate Loss Functions

The key to our unsupervised approach is the design of a surrogate loss function that guides the GNN to produce valid and low-cost tours without being shown any optimal solutions. We implemented and compared two such functions.

2.2.1. Baseline UTSP Loss

The first loss function is based on the original UTSP framework. It comprises two components:

1. **Expected Tour Length (L_length):** This term encourages the model to assign high probabilities to short edges. It is calculated as the sum of all edge lengths in the graph, each weighted by its predicted probability from the heatmap.
2. **Cycle Constraint (L_cycle):** This term enforces the structural properties of a valid tour. In a tour, every city must have exactly one incoming and one outgoing edge. This constraint is implemented by penalizing the deviation of the row-sums and column-sums of the heatmap from 1 using Mean Squared Error (MSE). This encourages the heatmap to become a doubly stochastic matrix, a relaxed form of a permutation matrix.

The total loss is a weighted sum: L_total = L_length + C1_penalty * L_cycle.

2.2.2. Proposed Alternative Loss

As a primary modification to the UTSP framework, we designed and implemented an alternative surrogate loss function with a more explicit penalty for subtour elimination. It consists of three components:

1. **Approximate Tour Length (L_length):** Similar to the baseline, this term minimizes the expected tour length.
2. **Degree-2 Penalty (L_degree):** This term encourages the sum of probabilities of all outgoing edges from each node to be 2. This is based on the property that in an undirected tour, each node has a degree of exactly 2.
3. **Subtour Elimination Penalty (L_subtour):** This novel component directly addresses the issue of subtour formation. We use a spectral graph theory approach by penalizing a small **Fiedler value** (the second smallest eigenvalue of the graph's Laplacian matrix). A Fiedler value close to zero indicates that the graph is disconnected or poorly connected. By maximizing this value (i.e., minimizing a penalty if it is too small), we explicitly encourage the GNN to output a heatmap corresponding to a single, connected component, thus forming a

single valid tour.

The total loss is a weighted sum: L_total = lambda_length * L_length + lambda_degree * L_degree + lambda_subtour * L_subtour.

2.3. Tour Construction from Heatmap

The trained GNN outputs a heatmap, not a tour. To decode this heatmap into a final solution, we use a two-step process:

1. **Greedy Tour Construction:** Starting from a random city, we build an initial tour by iteratively selecting the unvisited city with the highest edge probability from the current city, as indicated by the heatmap.
2. **2-Opt Refinement:** The initial greedy tour is often suboptimal. We apply the classic 2-Opt local search algorithm to refine it. 2-Opt iteratively improves the tour by removing two non-adjacent edges and reconnecting their endpoints in the only other possible way, keeping the change if it results in a shorter tour. This process is repeated until no further improvements can be found. To ensure efficiency, our 2-Opt implementation is optimized using Numba JIT compilation.

3. Experimental Setup

- **Dataset:** All experiments were performed on random Euclidean TSP instances, with city coordinates generated uniformly in the [0, 1] unit square for N = 20, 50, 100, 200, and 500.
- **Training Procedure:** Models were trained using the Adam optimizer. For stable training on larger instances (N ≥ 50), an initial learning rate of 0.0005 (or lower for N=500) was used with gradient clipping (max norm of 1.0). A StepLR learning rate scheduler was employed. The specific hyperparameters for each reported run are detailed in the appendix.
- **Evaluation Metrics:** Performance was measured by the average tour length over 100 test instances (20 for N=500). We also calculated the Optimality Gap (%) against known near-optimal solutions and measured the average inference time.
- **Environment:** Training was performed on Google Colab using NVIDIA GPUs. Evaluation was performed on a MacBook Pro with an Apple M1 Pro CPU. Key libraries include PyTorch, NumPy, and Numba.

4. Results and Analysis

The experimental campaign yielded a comprehensive dataset comparing the two loss functions across five problem sizes.

4.1. Overall Performance Comparison

Figure 1: Best average tour length achieved by the baseline and alternative loss functions for each problem size.

As shown in Figure 1, the performance of the two models is highly competitive. The proposed alternative loss function achieves marginally better results for medium-sized problems (N=50 and N=100). Conversely, the baseline loss function shows superior performance on small (N=20) and large (N=200, 500) instances. The most dramatic difference is seen at N=500, where the baseline model produced a significantly shorter tour length.

4.2. Scalability and Optimality Gap

Figure 2: Best average tour length vs. problem size (log scale).

Figure 3: Optimality gap vs. problem size (log scale).

Figure 2 illustrates the scalability of both models. Both tour lengths grow as expected with problem size, with the notable exception of the N=500 baseline result. This point is annotated as an anomaly; it is not plausible for a 500-city tour to be only marginally longer than a 200-city tour. This suggests a potential failure mode in the N=500 baseline training where the model may have converged to a state producing invalid or incomplete tours that were not fully corrected by the local search.

The Optimality Gap chart (Figure 3) provides the most insightful comparison. Both models remain within a 10-21% gap from optimal up to N=200, demonstrating their viability. The alternative model's gap is stable and predictable even at N=500 (~17.0%), whereas the baseline model's anomalous result leads to an impossible negative gap, reinforcing the need for further investigation of this specific outcome.

4.3. Scalability of Inference Time

Figure 4: Average inference time vs. problem size (log-log scale).

As shown in Figure 4, the inference time grows exponentially with problem size, which is expected. This cost is dominated by the Numba-optimized 2-Opt local search, whose complexity increases significantly with N. While fast for small problems (<1 second for N=100), the time required for N=500 (estimated >30 seconds/instance) highlights a practical limitation of this decoding strategy for very large-scale applications.

5. Conclusion

In this project, we successfully replicated and extended an unsupervised GNN-based framework for the TSP. We introduced a GAT model and a novel alternative surrogate loss function using the Fiedler value to enforce tour connectivity. Our findings show that this modified approach is highly effective, demonstrating competitive and even superior performance compared to the baseline on small and medium-sized problems.

However, the baseline loss function, despite its simpler formulation, exhibited better generalization to larger problem scales (N=200 and N=500), albeit with an anomalous result at N=500 that warrants further investigation. This suggests a critical trade-off between the complexity of the surrogate loss and its robustness on very large instances.

Future work could explore more advanced local search heuristics (e.g., 3-Opt or LKH) to improve decoding, investigate other GNN architectures like GIN or GraphSAGE, and further refine the balance of components in the alternative loss function to improve its scalability.