Fuzzy Systems and Data Mining X
A.J. Tallón-Ballesteros (Ed.)

doi:10.3233/FAIA241426

Efficient Early Sparsification for Accelerating Solutions to the Traveling Salesman Problem

Yonglu JIANG [a], Hongyu XING [b,c] Hanchen SHI [b,c] Zijing WEI [b,c] and Zhongguo YANG [a,d,1]

[a] *School of Information Science and Technology, North China University of Technology, Beijing, China*
[b] *School of London Brunel, North China University of Technology, Beijing, China*
[c] *Department of Mathematics, Brunel University London, London, UK*
[d] *Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data, North China University of Technology, Beijing, China*

Abstract. The Travelling Salesman Problem (TSP) is a well-established NP-hard problem, often tackled using heuristic algorithms. While state-of-the-art learning-driven approaches for TSP perform comparably to classical solvers when trained on trivially small instances, they struggle to generalize the learned policy to larger, practical-scale instances. Consequently, as the number of cities increases, the efficiency of these algorithms diminishes significantly. This paper focuses on the early elimination of non-advantageous edges while retaining advantageous ones, thereby accelerating the TSP solution. Our approach involves capturing the differential frequencies of various edges during the early stages of an intelligent evolutionary algorithm, which we term as "knowledge" derived from the evolutionary process. We then design effective features to characterize the similarities and differences between edges. Leveraging this "knowledge", we label edges as either advantageous or non-advantageous. By integrating the guidance of a predictor, we intervene in the intelligent evolutionary algorithm to sparsify the TSP graph, ultimately enhancing the TSP solution. Our method effectively prunes edges in TSP instances, preserving the optimal path while minimizing redundant edges. Experimental results demonstrate that our approach significantly improves both computational efficiency and solution quality on the TSP50 and TSP100 test sets compared to other models. This methodology offers a novel perspective by extracting and utilizing latent knowledge among edges, thereby enhancing the performance of intelligent evolutionary algorithms in solving the TSP. Our findings not only improve current methodologies but also encourage further exploration and development in related fields and practical applications.

Keywords. Traveling Salesman Problem, Intelligent Evolutionary Algorithm, Machine Learning, Knowledge, Speed up

[1] Corresponding Author: Dr.Zhongguo Yang, E-mail: yangzhongguo@ncut.edu.cn

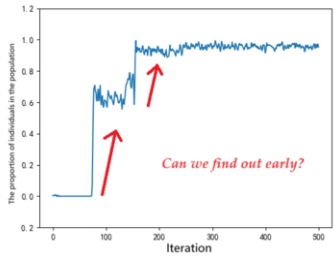
1. Introduction

The Traveling Salesman Problem (TSP) is a classic example of a combinatorial optimization problem with significant real-world implications. Originating in the fields of logistics and routing, it involves finding the shortest route that visits each city exactly once before returning to the starting point. Despite its apparent simplicity, the TSP is classified as an NP-hard problem, making the determination of an exact optimal solution computationally intractable as the number of cities increases, due to the problem's exponential time complexity. This challenge has driven the exploration of alternative approaches to efficiently tackle large-scale instances of the TSP.

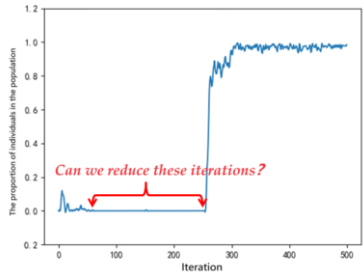
While state-of-the-art learning-driven TSP approaches perform comparably to classical solvers when trained on trivially small instances, they struggle to generalize the learned policy to larger, practical-scale instances. As the number of cities grows, the efficiency of these algorithms significantly diminishes. Traditional exact algorithms are also constrained by their exponential time complexity, rendering them impractical for real-world scenarios with numerous locations. To alleviate this computational burden, heuristic and approximation methods have been developed, though their limitations become apparent as the problem size increases [1]. Graph sparsification has emerged as a critical technique for addressing the TSP by reducing the solution search space. Researchers, including James Fitzpatrick, have investigated TSP instance sparsification using supervised learning models on the MATILDA dataset's CLKhard and LKCChard categories [2]. Their approach seeks to eliminate edges unlikely to be part of the optimal path. However, the effectiveness of this method is limited when faced with constrained data or the need for rapid processing, underscoring the necessity for enhanced generalization capabilities.

Fitzpatrick's work offered valuable insight into the unlikelihood of certain edges appearing in the optimal solution, which inspired our approach. By tracking the behavior of different edges within a TSP instance, we observed that the selection dynamics of edges within the population undergo significant changes as the algorithm iterates. Some edges gradually withdraw from selection, while others are consistently chosen, indicating early patterns. As depicted in Figure 1 (a) and (b), advantageous edges are swiftly identified and continually selected, reducing the number of iterations required for convergence. Conversely, Figure 1 (c) and (d) illustrates the rapid decline of non-advantageous edges, facilitating their early elimination and leading to effective graph sparsification. This method leverages empirical knowledge from the intelligent evolutionary algorithm, obviating the need for a pre-training dataset, to accelerate the sparsification of TSP problems and significantly enhance generalization ability. Our approach enhances the efficiency and effectiveness of TSP solutions by concentrating on promising edges and discarding less relevant ones. In the work of Liu, Bokai and Lu, Weizhuo[3], machine learning methods were employed to enhance computational efficiency, utilizing particle swarm optimization (PSO), which inspired our approach to combinatorial optimization.

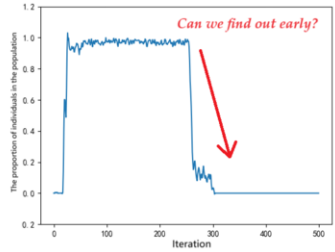
To operationalize this approach, we designed a set of features based on edge fitness and adjacency, extracting a feature set for each edge at the start of the iteration. As the algorithm progresses, we collect historical data on edge appearances and the optimal routes at various stages, using this information to create dynamic labels. Through this iterative



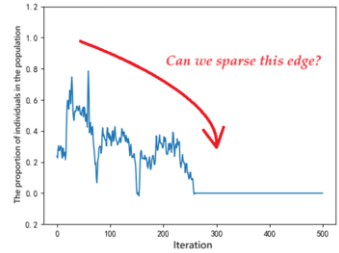
(a) Early recognition of advantageous



(b) Iteration reduction of advantageous



(c) Early detection of non-advantageous



(d) Sparsification potential

Figure 1. Introducing edge frequency-based sparsification

process, we identify dominant edges likely to appear in the optimal solution and non-dominant edges that are less likely to do so.

In subsequent iterations, we apply this knowledge to prune TSP instances. Non-dominant edges are deliberately removed from the population, while edges likely to appear in the optimal solution are retained. This strategy ensures the quality of the population, fosters its evolution, and further accelerates the sparsification of the TSP problem.

Our original contributions in this paper can be summarized as follows:

- **Knowledge-driven Feature Engineering:** We apply principles from intelligent evolutionary algorithms, integrating knowledge from iterative processes with carefully designed features to dynamically improve algorithm performance.
- **Graph Sparsification Enhancement through Intelligent Edge Analysis:** By dynamically identifying and prioritizing edges based on their likelihood of being part of the optimal solution, we achieve more effective graph sparsification. This reduces the search space and accelerates convergence.
- **Improved Generalization Capability of Swarm Evolution Algorithms:** Our algorithm does not rely on pre-existing datasets. Instead, it extracts knowledge directly from the input TSP instance, reducing dependency on external data and enhancing generalization capability.

Through comprehensive exploration and empirical evaluation, we aim to validate the effectiveness of our proposed methodology relative to existing approaches. These contri-

butions collectively highlight the novelty and potential impact of our methodology in solving large-scale TSP instances.

2. Related Work

In our research, we built upon the optimization algorithms and methodologies developed by previous scholars. By integrating their innovative approaches and collective expertise, we aimed to advance existing algorithms and contribute meaningfully to the discourse on TSP. This collaborative foundation enabled us to enhance the performance and effectiveness of TSP solutions, driving forward the field with improved methodologies.

2.1. Exact, Heuristic, Hybrid Solvers

The Concorde TSP solver is a leading tool for solving TSP instances [4]. It employs a range of efficient algorithms, including branch and bound, cutting planes, linear relaxation, and dynamic programming, to achieve high accuracy. The Lin-Kernighan-Helsgaun (LKH) TSP solver, an extension of the classic Lin-Kernighan algorithm improved by Keld Helsgaun, is a highly effective heuristic for large-scale TSP problems, demonstrating exceptional performance with thousands to millions of cities [5]. Google's OR-Tools provides a comprehensive suite of open-source libraries for solving complex optimization problems, including various algorithms for TSP, such as heuristic, exact, and hybrid methods [6]. Additionally, heuristic methods like nearest insertion and farthest insertion offer simplicity and effectiveness for small-scale problems. Despite the strengths of these tools, opportunities for improvement exist in computational resource utilization and the generalization of algorithms for very large-scale TSP instances.

2.2. Graph Neural Network Solution

Deep learning has shown promise in solving the TSP. Oriol Vinyals introduced the pointer network, a neural architecture designed to handle output sizes that vary with input sequence length [7]. By generating one million training samples, Ptr-Net can produce paths close to the optimal solution for small-scale problems, paving the way for deep learning applications in TSP. Similarly, Chaitanya K. Joshi developed a method using a graph convolutional neural network to solve the TSP, employing a highly parallelized beam search to generate non-autoregressive travel routes [8]. Although these methods excel in solution quality, reasoning speed, and sample efficiency, they face challenges when scaled to larger TSP instances. Issues include substantial training data requirements, limited solution accuracy, and restricted generalization. Additionally, as black box models, neural networks often present challenges in interpretability and verification, complicating their use in critical applications.

2.3. Graph Sparsification

The goal of graph sparsity is to reduce the number of edges, enhancing efficiency and simplifying computational tasks. For the TSP, this means removing edges unlikely to be part of the optimal solution. Yuan Sun's work involves extracting various features from

TSP instances, including topological and path features, and using an SVM model to predict and eliminate these less likely edges [9]. Similarly, James Fitzpatrick's approach utilizes a supervised model to achieve graph sparsity in the TSP [2], while Yong Wang generates sparse maps by analyzing frequency plots and applying frequency thresholds, which helps reduce the search space and complexity [10].

Our method builds on these approaches but differs by avoiding pre-trained models. Instead, we collect frequency information during the optimization process of an intelligent evolutionary algorithm to achieve graph sparsity.

3. Methodology

In this section, we outline the theoretical framework that supports our research and provide a detailed explanation of our methodology and its specific components. Our approach begins with an extensive feature extraction process that evaluates all edges within the problem domain. We then track the evolution of these edges throughout the iterations of the intelligent evolutionary algorithm. By examining changes in edge selection patterns, we classify edges as either advantageous or non-advantageous. Furthermore, we integrate a predictive model to estimate the potential advantages or disadvantages of other edges. The entire process is depicted in Figure 2.

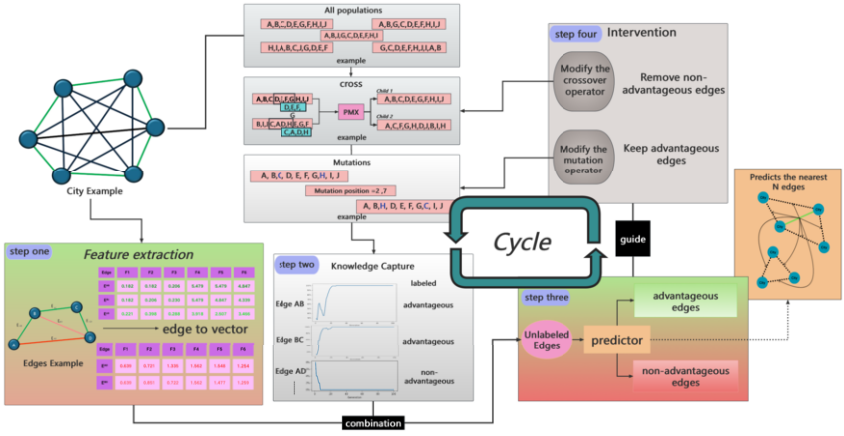


Figure 2. Knowledge-driven sparse graph optimization for TSP

3.1. Features Extraction

The work by Fitzpatrick et al. constructs six local features [2], which play a key role in guiding the decision-making process of classifiers. These features are grounded in the Euclidean distance between cities (i.e., $dist_{ij}$). Specifically, six features, denoted as F1 through F6, were designed to represent different proportional relationships, capturing normalized distance metrics. F1 quantifies the distance between two cities relative to the longest edge in the network, while F2 and F3 measure the distance between each city

and its farthest neighbor. To account for the inverse strength of these relationships, the reciprocals of these distances were also computed, corresponding to F4, F5, and F6.

$$dist_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2},$$

$$F1 = \frac{dist_{ij} + 1}{\max\_edge + 1}, \quad F2 = \frac{dist_{ij} + 1}{\max\_dist\_from\_i + 1}, \quad F3 = \frac{dist_{ij} + 1}{\max\_dist\_from\_j + 1},$$

$$F4 = \frac{\max\_edge + 1}{dist_{ij} + 1}, \quad F5 = \frac{\max\_dist\_from\_i + 1}{dist_{ij} + 1}, \quad F6 = \frac{\max\_dist\_from\_j + 1}{dist_{ij} + 1}.$$

These ratio-based features, incorporating both absolute distances and their relative relationships, form a comprehensive feature set that encapsulates the spatial interrelationships between cities within the TSP network. By considering both the magnitude of distances and the relative positions of cities, the model gains a deeper understanding of the spatial dynamics in play. This dual consideration of absolute and relative metrics enriches the model's capacity to detect patterns and dependencies within the TSP structure.

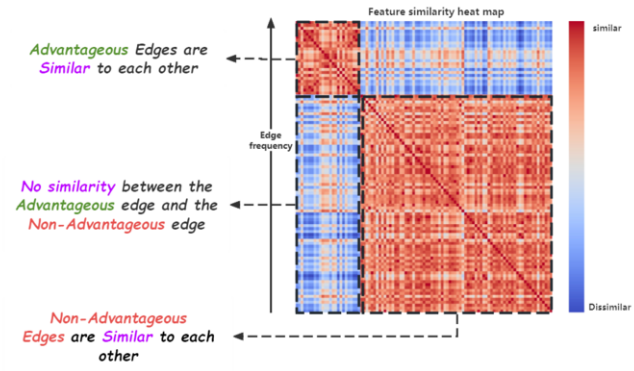


Figure 3. Heatmap of edge differences in feature space

Figure 3 presents a heatmap illustrating the feature differences across various edges. It is evident that significant distinctions exist between edges with high frequency (advantageous edges) and those with low frequency (non-advantageous edges). The similarity among advantageous edges is high, as is the similarity among non-advantageous edges, while the similarity between these two groups remains low.

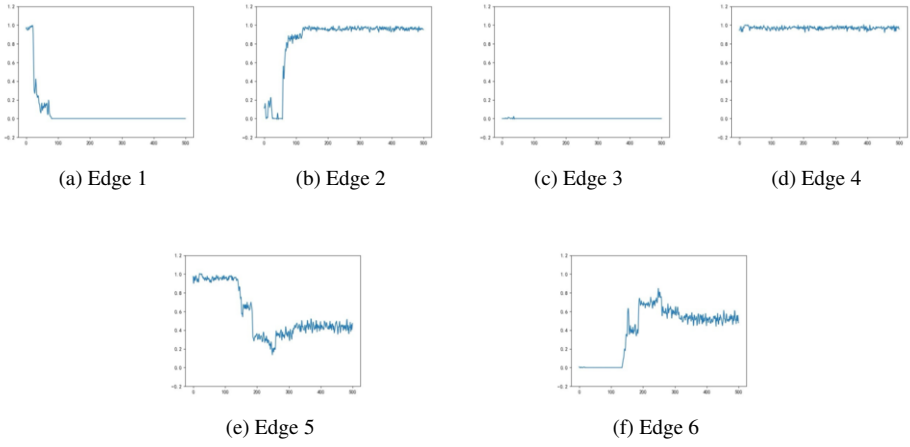
By learning these features and incorporating edge frequency data, the predictive model can be effectively trained. This approach allows the model to grasp the complex spatial interrelationships within the TSP network, providing a robust framework for predicting the behavior of coordinate networks.

3.2. Knowledge Capture

By analyzing the structure of the optimal solution for the TSP, we observe that a significant portion of the edge set does not appear in the optimal solution. Inspired by previous research [7] [8], such edges can be identified in advance. During the initial phase of the intelligent evolutionary algorithm, we monitor each edge in the population, recording the frequency of their appearance across iterations. As illustrated in Figure 4, the frequency of certain edges varies as the algorithm progresses. For example, Edge 3 and Edge 4 display distinct behaviors: Edge 3 is identified as non-advantageous, while Edge 4 is classified as advantageous—representing what we term "accurate knowledge". Conversely, Edges 1 and 2 initially exhibit advantageous or non-advantageous traits, but these characteristics diminish in later iterations, signifying "incorrect knowledge".

To reduce the impact of such incorrect knowledge, we introduce a sampling and labeling strategy, which helps to mitigate its influence. Based on this process, we categorize the edges we track into two distinct groups:

- **Advantageous Edges** Edges that appear with high frequency in high-quality solutions (i.e., those with shorter tour lengths) are classified as advantageous.
- **Non-Advantageous Edges** Edges that appear infrequently or predominantly in lower-quality solutions are classified as non-advantageous.



(a) Edge 1 (b) Edge 2 (c) Edge 3 (d) Edge 4

(e) Edge 5 (f) Edge 6

Figure 4. Trace the typical edges of the evolutionary process

Regarding Edge 5 and Edge 6 in Figure 4, their frequency change graphs exhibit complementary patterns, indicating a competitive relationship. Therefore, we handled these edges separately and included them in the search space. As described in Section 3.4, we subsequently selected the optimal edge from this competitive set.

3.3. Knowledge-driven predictor

Building on insights from Section 3.2, we predict unmarked edges in the search space to identify both promising and non-promising edges, with Edge 1 and Edge 2 in Figure 4

serving as key examples.

We begin by using features from Section 3.1 and the knowledge from Section 3.2 to label a subset of the population. This labeled subset is then used to evaluate the entire population, identifying N edges that are most and least advantageous in the feature space.

Frequency data is gathered every 10 iterations, with high-frequency edges labeled as 1 and low-frequency edges labeled as 0. Significant frequency changes, like those seen with Edge 1 and Edge 2, result in inverse labeling. For edges labeled 1 but similar in features to those labeled 0, their inclusion in the marked set is determined by an internal voting mechanism. Finally, marked edges are flagged as 1, and the K-nearest neighbors algorithm is applied to find the N nearest neighbors, helping to identify dominant and non-dominant edges, thereby enhancing the search process through intelligent evolution.

3.4. Intervention algorithm for sparse

We utilize the insights from Section 3.3 to identify both advantageous and non-advantageous edges within the population. This information is then used to guide and adjust the genetic algorithm operators, effectively reducing the search space and enabling edge pruning. To select the next generation of parent solutions, we apply a tournament selection mechanism. In this approach, a random subset of solutions is selected from the population, and the best-performing solution within this subset is chosen as the parent. This process is repeated to select additional parents, incorporating predicted dominant edges to increase the likelihood of selecting superior solutions.

The selected parent solutions undergo crossover operations to generate offspring, introducing variability and promoting exploration of the solution space. We utilize partial mapping crossover (PMX) and order crossover (OX) [11]. To maintain population diversity and prevent premature convergence, mutation operations are applied to the offspring, including exchange and inversion mutations [12]. During mutations, dominant edges are avoided, focusing on non-dominant edges to encourage high-quality mutations.

Additionally, we introduce a repair operator designed to correct edges unlikely to appear in the optimal solution. This involves checking new individuals in each generation and replacing suboptimal edges through local optimization or replacement operations. The fitness function is adjusted to reward individuals containing high-quality edges by incorporating an edge reward term: the more high-quality edges an individual contains, the higher the reward. This helps retain superior edges for the next iteration.

These four operations collectively eliminate non-advantageous edges and retain advantageous ones, effectively reducing the search space and sparsifying the graph. This iterative approach accelerates the TSP solution process. As depicted in the flowchart in Figure 2, these steps form a continuous loop. Given that incorrect information may arise during execution, it is crucial to continuously update and integrate the latest knowledge to maintain the robustness and efficiency of the graph sparsification process.

For more information we refer the reader to the following surveys [13][14].

4. Experiments and Results

In this section, we discuss benchmark problem instances, outline the procedures for conducting training and experiments, and evaluate the effectiveness of our method on instances from Ruprecht-Karls-Universität Heidelberg and other notable cases with distinct characteristics [15][16]. Specifically, our approach was implemented using Python (3.8), SciPy (1.7.3), NumPy (1.21.5), and Pandas (1.4.2) libraries for numerical operations, the Scikit-learn (1.0.2) library for machine learning tasks, and the Matplotlib (3.5.1) library for visualizing experimental results. The experiments were conducted on a Windows 11 system equipped with an Intel i9 processor, 16GB of RAM, and an Nvidia 4060 GPU. These details provide a comprehensive overview of the software and hardware environment utilized in our experiments.

4.1. Learn Features and Predict Advantageous Edges

Features The experimental results identified six key features (F1-F6), represented as ratios. These features capture the relative significance of edges, city distances, and overall network properties. By combining these features with edge frequency data from the Traveling Salesman Problem (TSP), we utilize a frequency-based ranking system to better understand edge importance. This approach highlights the role of feature extraction and frequency ranking in improving optimization accuracy and emphasizes the importance of understanding the interaction between various features in predicting optimal routes.

Predict of Edge Frequencies The experimental process begins with loading a sorted and merged dataset containing both TSP features and actual edge frequencies. The data is split into features and labels, with the first 'n' entries used as the training set. A K-Nearest Neighbors Regression model predicts edge frequencies, while a Nearest Neighbors model identifies the closest unlabeled neighbors for each sample. The results, including the predicted labels and corresponding edges, are stored in a DataFrame.

To improve accuracy, duplicate entries are filtered, keeping only edges with the highest predicted values. The refined results are saved to a new CSV file. This method demonstrates the potential of machine learning models in route prediction and emphasizes the importance of data preprocessing in optimizing the accuracy of practical solutions.

4.2. Analysis of Frequency Dynamics of Advantageous and Non-Advantageous Edges

Figure 5 illustrates the evolutionary dynamics of edge selection frequencies during TSP optimization using a learning-based algorithm. The central network graph represents a TSP instance, with nodes as cities and edges as potential paths. Edges are classified based on their contribution to optimal or near-optimal solutions.

The left panel displays the frequency dynamics of advantageous edges, while the right panel focuses on non-advantageous edges across successive generations of the genetic algorithm.

- **Advantageous Edges (Left Panel)** The three plots show the proportion of advantageous edges increasing rapidly, stabilizing near 100% by the 30th generation. This indicates the algorithm effectively identifies and reinforces useful edges.

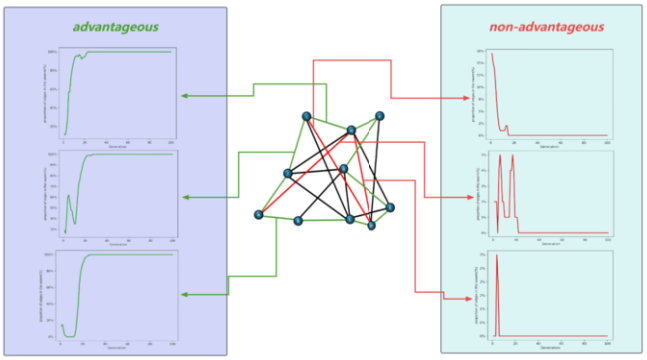


Figure 5. Frequency dynamics of advantageous and non-advantageous edges

- **Non-Advantageous Edges (Right Panel)** The plots illustrate a rapid decline in the frequency of non-advantageous edges, approaching 0% within the first 30 generations, highlighting the algorithm's ability to eliminate less optimal edges.

Overall, Figure 5 demonstrates how edge frequencies evolve during optimization. By tracking these changes, edges can be categorized into advantageous and non-advantageous groups, reducing the search space and sparsifying the TSP graph. Focusing on advantageous edges enhances both the efficiency and generalization of the optimization process, with the frequency dynamics validating the algorithm's effectiveness.

4.3. Sparsification of Complete Graphs based on Prediction Results

To assess the sparsification performance of our feature-based prediction model, we tested three TSPLIB instances: wi29, dj38, and st70 [15]. We first solved these instances using a traditional genetic algorithm and then applied our prediction model. The number of iterations required for each algorithm to converge was compared. Figure 6 shows the results for the predictive model.

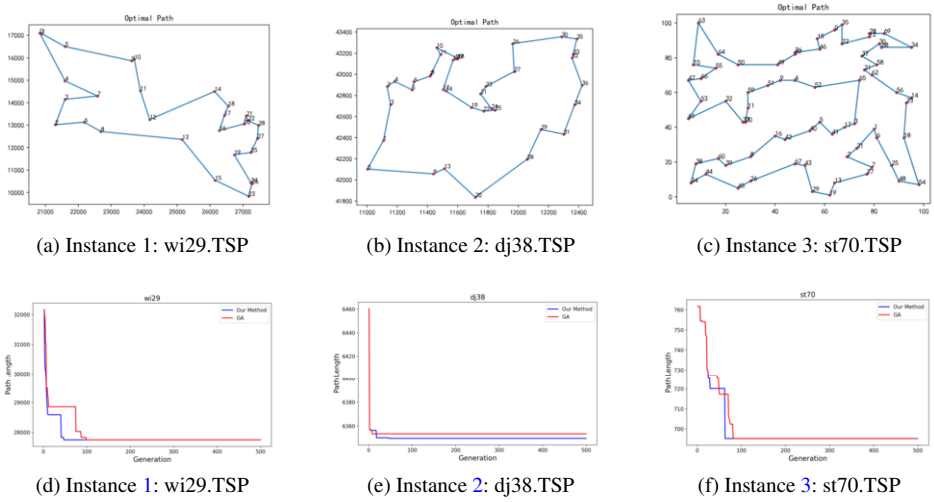
Instance 1 The prediction model converged to the optimal solution at the 44th iteration, and the sparse result contained 100% of the optimal solution at the 30th iteration.

Instance 2 The prediction model converged to the optimal solution at the 49th iteration, and the sparse result contained 100% of the optimal solution at the 30th iteration.

Instance 3 The prediction model converged to the optimal solution at the 63th iteration, and the sparse result contained 100% of the optimal solution at the 40th iteration.

The model demonstrated rapid convergence across all test cases, highlighting its efficiency in feature extraction and knowledge utilization, which resulted in faster convergence to optimal solutions.

Specific Analysis Instance 3 Using Instance 3 (st70.TSP) as an example, we explored the sparsification process (Figure 7). After 200 iterations, the graph's initial 4556 edges were reduced to 441 edges. Notably, these sparsified edges contained all the optimal so-

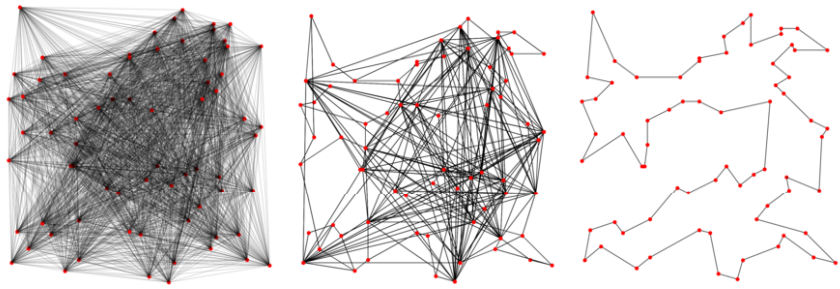


(a) Instance 1: wi29.TSP (b) Instance 2: dj38.TSP (c) Instance 3: st70.TSP

(d) Instance 1: wi29.TSP (e) Instance 2: dj38.TSP (f) Instance 3: st70.TSP

Figure 6. Optimal solution obtained by prediction model

lutions identified by the genetic algorithm at the 985th iteration. This result underscores the effectiveness of the prediction model in identifying and predicting optimal edge selections, significantly reducing problem complexity.



(a) Complete Graph (b) Sparse of the 200th (c) Optimum of 985th

Figure 7. Instance 3: st70.TSP sparsification process

4.4. Comparative Analysis of Algorithm Performance

Instances Experiments To assess the performance of our method, we tested it against established optimization algorithms on 16 TSPLIB benchmark instances [15]. The algorithms compared include Ant Colony Optimization (ANT), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Simulated Annealing (SA), and Tabu Search (TS). We evaluated the optimum solution and the number of iterations required to achieve this optimum. Figure 8 presents a scatter plot of these results, with the horizontal axis showing the optimum found and the vertical axis representing the iteration at which the last change occurred. The TSP instances are indicated as numbers within circles.

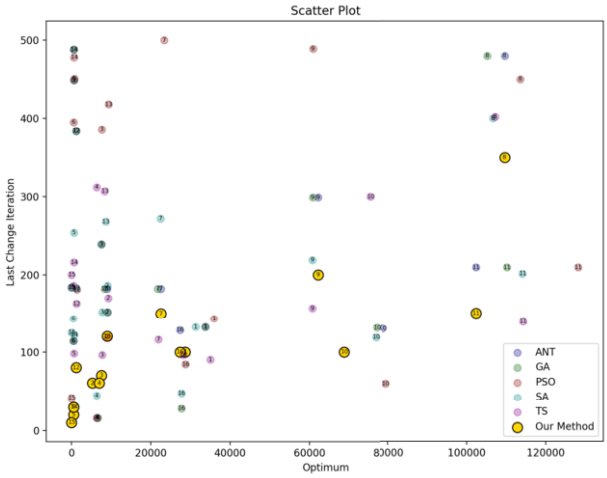


Figure 8. Scatter plot of optimum and iteration counts for TSPLIB instances across various algorithms

Our method exhibits competitive efficiency, achieving optimal solutions in fewer iterations compared to most other algorithms. For instance, our method reached the optimum within the first 100 iterations for several instances (e.g., instances 1, 2, and 3), outperforming others in convergence speed.

While many algorithms achieve similar optimum values, our method consistently delivers near-optimal or optimal solutions. The clustering of brown markers at lower optimum values indicates robust performance. Although algorithms like GA and SA also show clusters of low optimum values, their higher iteration counts suggest a trade-off between solution quality and computational effort.

Certain outliers in the scatter plot reveal instances where some algorithms either significantly underperform or require more iterations to reach the optimum. For example, instance 8 shows higher optimum values with ANT and PSO, indicating challenges in those cases. Our method avoids such extreme outliers, demonstrating greater reliability and consistency across diverse problem instances.

Numerical Experiments During testing, our model was evaluated using beam search and demonstrated superior performance with the smallest error. Table 1 compares our model against methods from existing literature, including GA, PSO, and SAA, as well as heuristic methods like Nearest Insertion and Farthest Insertion for TSP50 problems [16].

The results reveal that while GA, PSO, and SAA provide competitive objective values with minimal optimality gaps (ranging from 0.40% to 0.42%), they require slightly more computational time compared to our model, which achieves the best balance between solution quality and efficiency. Heuristic methods such as Nearest Insertion and Farthest Insertion, though fast, exhibit much higher optimality gaps (22.94% and 5.53%). Our model excels by providing high-quality solutions with reduced computational time, making it ideal for scenarios where both speed and accuracy are critical.

Table 1. Comparison of results for 10,000 instances of TSP50 instances

Method	Objective	Optimality Gap	Total Time	Inference Time
Our Model	5.707	0.31%	13.7s	0.07s
GA [17]	5.3	0.40%	14.5s*	0.09s
PSO [18]	5.1	0.42%*	14.8s*	0.08s
SAA [19]	5.2	0.41%*	14.2s*	0.085s
Nearest Insertion	7.00*	22.94%*	0s*	-
Farthest Insertion	6.01*	5.53%*	2s*	-

* Approximate values.

5. Discussion and Conclusions

The proposed knowledge-driven hybrid method for predicting and sparsifying paths in the TSP represents a notable advancement in optimizing algorithm efficiency. By leveraging insights from an intelligent evolutionary algorithm, the method effectively prunes non-advantageous edges while retaining beneficial ones, leading to a more streamlined solution space. This approach accelerates the TSP solution process and enhances solution quality.

A key strength of this method is its dynamic adaptation and learning from the evolutionary process, which reduces reliance on extensive pre-training datasets and improves generalization. However, challenges remain with high-dimensional and complex data structures, and future research should consider integrating advanced techniques such as graph neural networks (GNNs) and topological structure analysis. In the work of Liu Bokai et al. [20], a hybrid machine learning method was used to predict the thermal conductivity of PNC. They combined ANN with PSO, which provides valuable insights and potential directions for future research. Additionally, while efforts are made to avoid local optima, the risk of knowledge solidification persists. Adaptive parameter adjustments and search strategies could help maintain diversity and prevent premature convergence.

Overall, this study introduces a novel approach by combining knowledge-driven prediction with intelligent evolutionary methods, demonstrating significant improvements in efficiency and accuracy, particularly for TSP50 and TSP100 with Euclidean distances. Future work should focus on refining feature extraction techniques and exploring adaptive strategies to enhance robustness and effectiveness. The promising results underscore the potential of knowledge-driven approaches in solving complex combinatorial optimization problems and suggest avenues for further advancement in this field.

References

[1] CK Joshi, Q Cappart, LM Rousseau, T Laurent, and X Bresson. Learning tsp requires rethinking generalization. arxiv 2020. *arXiv preprint arXiv:2006.07054*.

[2] James Fitzpatrick, Deepak Ajwani, and Paula Carroll. Learning to sparsify travelling salesman problem instances. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 410–426. Springer, 2021.

[3] Bokai Liu and Weizhuo Lu. Surrogate models in machine learning for computational stochastic multi-scale modelling in composite materials design. *International Journal of Hydromechatronics*, 5(4):336–365, 2022.

[4] David L Applegate, Robert E Bixby, Vasek Chvatal, and William J Cook. *The traveling salesman problem: a computational study*. Princeton University Press, 2006.

[5] Keld Helsgaun. *An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems*. Roskilde University, Roskilde, 2017.

[6] Google. Or-tools: Google's operations research tools, 2015.

[7] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks, 2017.

[8] Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*, 2019.

[9] Yuan Sun, Andreas Ernst, Xiaodong Li, and Jake Weiner. Generalization of machine learning for problem reduction: a case study on travelling salesman problems. *Or Spectrum*, 43(3):607–633, 2021.

[10] Yong Wang. An approximate method to compute a sparse graph for traveling salesman problem. *Expert Systems with Applications*, 42(12):5150–5162, 2015.

[11] Colin R Reeves. Genetic algorithms. *Handbook of metaheuristics*, pages 109–139, 2010.

[12] Noraini Mohd Razali, John Geraghty, et al. Genetic algorithm performance with different selection strategies in solving tsp. In *Proceedings of the world congress on engineering*, volume 2, pages 1–6. International Association of Engineers Hong Kong, China, 2011.

[13] Samuel A Oluwadare, Bosede A Ogunsanmi, and John C Nwaiwu. Towards solving travelling salesperson problem using hybrid of genetic algorithm and lin-kernighan algorithm: A comparative evaluation with neural network model. *International Journal of Computer Applications*, 975:8887.

[14] Bladimir Toaza and Domokos Esztergár-Kiss. A review of metaheuristic algorithms for solving tsp-based scheduling optimization problems. *Applied Soft Computing*, page 110908, 2023.

[15] G. Reinelt. Tsplib - a traveling salesman problem library, 1991.

[16] Xavier Bresson and Thomas Laurent. The transformer network for the traveling salesman problem. *arXiv preprint arXiv:2103.03012*, 2021.

[17] Fei Liu and Guangzhou Zeng. Study of genetic algorithm with reinforcement learning to solve the tsp. *Expert Systems with Applications*, 36(3):6995–7001, 2009.

[18] Kang-Ping Wang, Lan Huang, Chun-Guang Zhou, and Wei Pang. Particle swarm optimization for traveling salesman problem. In *Proceedings of the 2003 international conference on machine learning and cybernetics (IEEE cat. no. 03ex693)*, volume 3, pages 1583–1585. IEEE, 2003.

[19] Xiutang Geng, Zhihua Chen, Wei Yang, Deqian Shi, and Kai Zhao. Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search. *Applied Soft Computing*, 11(4):3680–3689, 2011.

[20] Bokai Liu, Nam Vu-Bac, and Timon Rabczuk. A stochastic multiscale method for the prediction of the thermal conductivity of polymer nanocomposites through hybrid machine learning algorithms. *Composite Structures*, 273:114269, 2021.