Solving Travelling Salesman Problem using Self-Organising Maps

Mukta Patil[1], Udayan Gaikwad[2], Dr.Deepak Mane[3]

[1,2,] *UG Student, Department of Computer Engineering*

[3]*Faculty, Department of Computer Engineering*

BRACT's Vishwakarma Institute of Technology, Pune

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

***Abstract*— This paper introduces a revolutionary approach to the Traveling Salesman Problem (TSP) using Self-Organizing Maps (SOMs), a neural network paradigm known for topology preservation [5]. By constructing a two-dimensional lattice, the algorithm maps cities, promoting the emergence of near-optimal solution segments based on spatial proximity. Dynamic city clustering through iterative updates facilitates the creation of highly efficient tours. To boost performance, the model incorporates distance information and competitive learning, yielding solutions with exceptional quality and computational efficiency. Unlike traditional methods, this novel approach leverages SOMs' unique properties. Extensive testing on real-world datasets provided by University of Waterloo validates its effectiveness beyond a sandboxed environment, showcasing practical applicability in solving TSP with superior efficiency and solution quality.**

***Keywords*— TSP, SOMs, Neural Networks, Optimization, Clustering, Machine learning**

I. INTRODUCTION

The Traveling Salesman Problem (TSP) [9] remains a cornerstone of optimization theory, presenting a challenge to efficiently traverse a set of discrete locations while minimizing total distance traveled [6]. Although established dynamic programming approaches offer powerful solutions, their computational complexity can become prohibitive for large-scale problems. This motivates the exploration of alternative methods that can provide near-optimal solutions with greater efficiency.

Presenting an innovative solution to the Traveling Salesman Problem (TSP)[9], this paper employs Self-Organizing Maps (SOMs[8]), a distinctive type of artificial neural network renowned for preserving topology. Our approach harnesses the unique capability of SOMs to map high-dimensional data onto low-dimensional structures while preserving spatial relationships. The proposed algorithm dynamically clusters cities based on proximity, a feature crucial for identifying promising sub-tours within the overall solution. This departure from traditional exhaustive search methods underscores the novelty of our approach, offering a more efficient and effective strategy for tackling the intricacies of the TSP.

The proposed approach is marked by its ambitious goals, underpinned by clearly defined objectives and performance benchmarks. These objectives include:

- **Innovative TSP Solution**: Present a novel approach using Self-Organizing Maps (SOMs) for efficient Traveling Salesman Problem (TSP) resolution, emphasizing unique mapping capabilities.
- **Dynamic City Clustering**: Develop a proximity-based algorithm within SOMs, departing from exhaustive methods to identify efficient sub-tours in TSP.
- **Real-world Validation**: Rigorously validate the solution's practicality, efficiency, and quality across diverse TSP scenarios using real-world datasets.

In the next section, we review previous works in the domain, delve into the specifics of our proposed solution, detail the benchmarks for performance evaluation, and compare its efficiency with existing methods. Additionally, we briefly touch upon the potential future scope of the project.

II. RELATED WORKS

The TSP has been approached in many ways.From brute-force enumeration to AI-inspired metaheuristics, researchers have thrown a diverse toolbox at the Traveling Salesman Problem. Exact algorithms guarantee the best route, but for giant journeys, fast heuristics take the wheel. Hybrid approaches blend strategies, and even machine learning joins the quest for optimal paths. Each method brings its strengths and limits, shaping the ongoing race to conquer the intricate labyrinth of the travelling salesman.

The paper "Dynamic programming strategies for the Traveling Salesman Problem with Time Window and Precedence Constraints"[10] presents an approach to a complex TSP variant involving time windows and visit order restrictions. Utilizing dynamic programming, it incrementally constructs an optimal tour, minimizing travel cost while adhering to constraints. Despite its power, its high complexity limits its scalability to smaller instances. The paper introduces an efficient computation technique, marking a valuable contribution and enhancing our comprehension of this intricate TSP challenge[10].

The paper title “A Heuristic Approach to Solving Travelling Salesman Problems”[1] by Robert L. Karg and Gerald L. Thompson, introduces two novel heuristic approaches to address the Traveling Salesman Problem (TSP), departing from traditional methods characterized by computational complexity. "Random insertion" efficiently constructs a tour, emphasizing simplicity. "Divide-and-conquer" strategically breaks down the TSP, addressing sub-problems with "random insertion" and merging sub-routes. Evaluation highlights the efficiency of "random insertion" and the quality of "divide-and-conquer," with increased complexity. This contribution provides valuable alternatives for optimizing TSP solutions.

TSP has also been examined using Metaheuristic approaches, drawing inspiration from natural processes such as evolution or ant colony behavior to probabilistically explore the solution space, proving effective for finding near-optimal solutions to complex problems. The paper titled “Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem” (Dorigo & Gambardella, 1997) [11] introduces a TSP algorithm inspired by ant behavior. Artificial ants leave pheromone trails on city edges, probabilistically choosing shorter paths. The system adapts, performs well for diverse TSP variations, and scales efficiently. Challenges include parameter tuning and potential premature convergence. Despite limitations, the approach provides a novel perspective on TSP optimization, contributing to swarm intelligence and metaheuristics research.

In pursuit of clearer evaluation for machine learning-based Traveling Salesman Problem (TSP) solvers, François, Cappart, and Rousseau introduce the "Ratio of Optimal Decisions" (ROD) metric to address the evaluation challenges. ROD provides clarity by comparing model choices to an optimal "oracle." The authors apply ROD to assess four leading TSP models, revealing individual strengths and weaknesses. This approach informs future research, enabling tailored model selection and advancing ML-driven combinatorial optimization boundaries[2].

Ishaya et al. (2019) [3] compare the traditional branch-and-cut algorithm with a machine learning-based approach for the Travelling Salesman Problem (TSP). The former guarantees optimal paths but faces computational challenges with more cities, while the machine learning-powered recurrent neural network swiftly predicts near-optimal tours, excelling on 2D maps with up to 200 cities. Despite promising results on familiar terrain, challenges remain in higher dimensions and complex scenarios. This study hints at a potential future partnership between traditional rigor and machine learning speed in TSP solvers[3].

The analysis of these diverse approaches to resolution of TSP provides valuable insight and inspiration for developing and validating our own SOM-based algorithm. [2] exposes limitations in evaluating machine learning models for combinatorial optimization like TSP, whereas [3] describes the specific scenarios where machine learning shines (e.g., speed for smaller problems) which enables us to position our SOM-based approach within the broader TSP solver landscape.

III. PROPOSED ARCHITECTURE

This research endeavors to bridge the gap between the theoretical realm of the Traveling Salesman Problem (TSP) and the practical challenges of real-world travel planning. We depart from the controlled environments of classical TSPs, characterized by idealized distances and static configurations, and instead delve into the dynamic complexities of actual travel networks.

A. Dataset

To accomplish this, we leverage the National TSP Dataset, a rich resource of real-world city locations curated by the University of Waterloo[4]. This dataset, sourced from the National Imagery and Mapping

Agency, provides a foundation grounded in the intricacies of actual geography, incorporating factors like distance, travel time, and geographical constraints. By utilizing this data-driven approach, we move beyond the limitations of simplified TSP models and aim to develop a solution framework applicable to the multifaceted realities of global travel.

```
NAME : ar9152
COMMENT : 9152 locations in Argentina
COMMENT : Derived from National Imagery and Mapping Agency data
TYPE : TSP
DIMENSION : 9152
EDGE_WEIGHT_TYPE : EUC_2D
NODE_COORD_SECTION
1 36266.6667 62550.0000
2 34600.0000 58633.3333
3 51650.0000 72300.0000
4 37800.0000 67683.3333
```

Fig 1: Glimpse of dataset

Prior to implementing the TSP algorithm, the city data from the National TSP Dataset was processed and normalized [7]. The data, initially structured as a Pandas DataFrame with columns for city names, latitude, and longitude, was converted into a NumPy array for efficient manipulation. To ensure equal dimensionality and facilitate subsequent computations, the data underwent a two-step normalization process. First, the aspect ratio of the geographic space was computed by dividing the range of longitude values by the range of latitude values.

$$\textbf{Aspect Ratio} = \frac{\textbf{Max Longitude - Min Longitude}}{\textbf{Max Latitude - Min Latitude}}$$

This ratio then served as a scaling factor for subsequent normalization. Second, each data point, represented as a two-dimensional vector, was individually normalized using a min-max scaling technique. This technique subtracts the minimum value from each dimension and then divides by the range (maximum minus minimum) in that dimension.

$$\textbf{f(c)} = \frac{\textbf{c - min(c)}}{\textbf{max(c) - min(c)}}$$

$$\textbf{g(p)} = \textbf{Aspect ratio x p}$$

This operation ensures that all dimensions (longitude and latitude) have the same scale and are comparable within the normalized data space. Ultimately, this pre-processing step prepares the city data for efficient and accurate application of the chosen TSP algorithm.

	x	y
0	0.000000	0.0
1	0.500000	0.5
2	1.000000	1.0

Fig 2: Resulting DataFrame

B. Creation of Neural Network

The network initialization process within the Self-Organizing Map (SOM) algorithm relies on the generate_network function, tasked with creating a random distribution of neurons. This function accepts a single parameter, size, defining the desired number of neurons in the network. It subsequently returns a NumPy array of shape (size, 2), representing the coordinates of each neuron within the two-dimensional space. Importantly, these coordinates are randomly distributed within the unit interval [0, 1], ensuring an initial unbiased placement of neurons within the self-organizing landscape. Notably, this randomization introduces an element of stochasticity in the SOM training process, potentially leading to diverse solution outcomes and encouraging further exploration of the search space. In essence, the generate_network function lays the foundation for the emergent behavior of the SOM, initializing the map with a blank canvas onto which the intricate tapestry of self-organization will be woven.

Then the get_neighborhood function is implemented, which calculates a Gaussian distribution around a specified neuron index in a circular network, considering a given radius and the total number of neurons in the network. The center index determines distances to other neurons, influencing weights via a Gaussian function. The radius parameter controls the extent of influence, with a larger value indicating a broader impact. To avoid computational issues, an upper bound on the radius is enforced. Circular distances are computed by finding the minimum path around the network. The Gaussian distribution is then determined by the computed distances, utilizing a decay function proportional to the square of the radius. This concise process ensures effective neighborhood computation within the circular network.

C. Resolution

The resolution of the Travelling Salesman Problem is achieved via a triad of functions, namely, `get_routes`, `euclidean_distance`, and `select_closest`. Particularly germane to this solution is the utilization of a Self-Organizing Map (SOM), a neural network paradigm renowned for its inherent capacity for

autonomous adaptation and spatial organization. The competitive learning and neighborhood functions within SOMs enable them to organize and represent the input data in a way that facilitates the identification of optimal routes in the TSP. The network autonomously organizes itself to capture the underlying structure of the cities' spatial distribution, making it a self-organizing approach to solving the TSP.

In the initial phase led by the `*get_routes*` function, operating on a structured dataset of cities and a corresponding neuron network array, a crucial 'winner' column is added to the cities dataframe using the `*select_closest*` function. This column plays a pivotal role in determining the index of the nearest SOM neuron for each city. Following this determination, the cities dataframe undergoes a meticulous sorting based on these 'winner' indices, revealing an optimal route configuration.

The significance of the `*euclidean_distance*` function lies in its role as a linchpin for proximity assessment. It meticulously computes Euclidean distances, encapsulating nuanced spatial relationships between pairs of cities. These distances, serving as proxies for geographical closeness, validate the SOM's unsupervised learning mechanism. Notably, the select_closest function strategically uses these distances to identify the index of the closest neuron relative to a given origin point. This orchestrated interplay of functions manifests an inherent self-organizing mechanism within the SOM, adeptly capturing and internalizing the intrinsic spatial structure of cities.

Through autonomous adaptability to the spatial configuration of the cityscape, the SOM contributes to a sophisticated, efficient, and inherently self-organizing algorithmic solution for the intricate Traveling Salesman Problem domain.

D. Representation and Plotting

The project culminates into a visually representing optimal paths across geographical coordinates. For visualization, 2 functions are implemented. . The plot_network function accepts datasets representing cities and neurons, generating a scatter plot where cities are denoted in red and neurons in blue. The resulting visualization is stored as a PNG image file named 'diagram.png'. Additionally, the function accommodates an optional parameter, ax, enabling the incorporation of a pre-existing Axes object for the scatter plot, with the option to return the Axes object if provided.

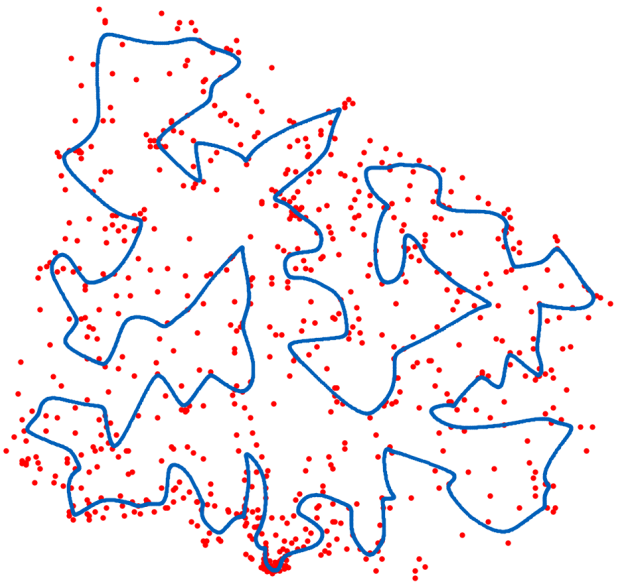

Figure 3: Cities and Neurons of Uruguay

Conversely, the plot_route function operates on a pandas DataFrame 'cities' with 'x' and 'y' columns depicting city coordinates, along with a 'route' parameter denoting the order of city visitation. Employing matplotlib, this function produces a graphical representation of the tour, portraying cities as red dots and the tour as a connecting purple line in the specified route order. Analogous to the plot_network function, the ax parameter allows for the integration of a pre-existing Axes object, and the resultant plot can be stored as a PNG image file if a filename is provided. Together, these functions contribute a visual dimension to the project's exploration of optimal paths within a geographic context.

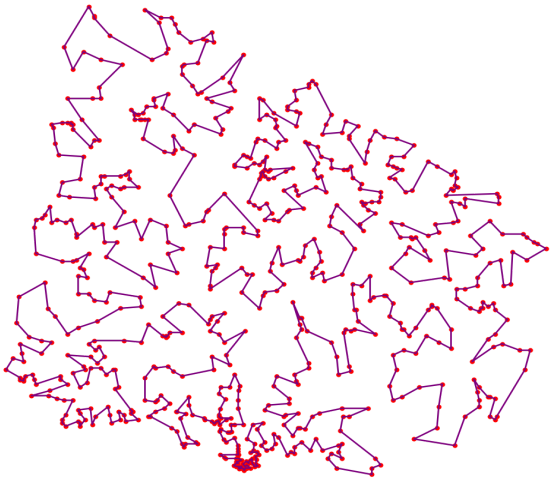

Figure 4: Final Optimized Route of Uruguay

Sr. No	Country	City Count	Iterations Count		Iteration Per City	Overall Length (in units)		Time (seconds)	Difference	Quality
			Max	Actual		Path	Actual			
1	Qatar	194	100000	24487	126.222	9352	9710.5	14.3	358.5	3.82%
2	Uruguay	734	100000	28922	39.403	79114	82953.73	23.4	3839.73	4.85%
3	Finland	10639	100000	37833	3.532	520527	548552	284.0	25355	4.87%
4	Argentina	9152	100000	37331	4.079	837479	908950.87	314.04	71471.87	8.53%
5	Japan	9887	100000	37575	3.781	491924	534297.46	435.99	42373.46	8.61%

Table 1: Performance metrics of the model

IV. MODEL EVALUATION

The assessment of the implemented methodology involves the utilization of instances provided by the National Traveling Salesman Problem library, inspired by real countries and featuring optimal routes for reference. The evaluative strategy encompasses the execution of multiple problem instances, focusing on key metrics articulated as follows:

- Execution Time: time expended by the technique in reaching a solution.
- Solution Quality: The quality of the solution is measured relative to the optimal route. A route deemed "10% longer than the optimal route" corresponds precisely to 1.1 times the length of the optimal trajectory.

The evaluation employs predefined parameters established through a systematic parametrization process, with initial values guided by a specified reference. These parameters include:

- Population Size: Set at 8 times the number of cities in the problem.
- Initial Learning Rate: Fixed at 0.8, complemented by a discount rate of 0.99997.
- Initial Neighborhood Size: Initially matched to the number of cities, with subsequent decay by a factor of 0.9997.

This methodical evaluation, rooted in meticulous parameterization, aims to furnish a comprehensive analysis of the technique's performance across diverse problem instances.
Additionally, the Table1 describes the performance of the model over various available dataset. It tabulates the actual performance of SOM over 5 countries, including the number of cities within the country, number of iterations required to produce the solution, the difference of the derived path from the actual optimal path, and the execution time. The quality column represents the difference in more comprehensible terms.

Other observations recorded from the project were the direct correlation between the number of cities and the time consumed by the project. This can be attributed to the complexity of determining the closest neuron for each city depending on the population to traverse. While the potential for superior results exists in larger city sets, practical considerations often necessitate constraints on computational time. Nevertheless, the obtained results consistently remain within a threshold of 10% as compared to the optimal route.
However, the reason for presenting the performance of the model over Uruguay was due to the noteworthy observations that arise in this case, where superior solution quality is achieved despite the inherent difficulty posed by the country's topography.

V. COMPARISON WITH EXISTING SYSTEMS

TSP has been addressed through diverse algorithms, each employing distinct techniques with inherent advantages, limitations, and associated performance metrics. This section delves into the nuances of these techniques, offering a comparative analysis against proposed mechanisms.
The paper referenced as [10] introduces an algorithm characterized by high computational complexity, limiting its practicality for large-scale problems. This stands in contrast to the proposed methods, which are applied and validated on realistic datasets. In [1], the accuracy can be inferred to fall within the range of 80-90%, as the paper lacks explicit quantification but acknowledges certain limitations. [11] strikes a balance between achieving precise optimality and computational speed.
In the case of [12], the algorithm provides solutions deemed "near-optimal" for smaller datasets, yet its performance diminishes for larger datasets. On the other hand, [13] boasts a highly time-intensive process but yields the most optimal route, a record that remains unbroken to date. The proposed system however,

produces some of the best results out of the analyzed algorithms with lower computation time, and high scalability, performing well for smaller and larger datasets and keeping the results within 5% of the optimum path. The accuracy of the model is around 88-92% while the average time for finding optimum route is 214 secs.

VI. CONCLUSION

In conclusion, the conducted experiments have unveiled a compelling application of organizational techniques, specifically Self-Organizing Maps (SOMs), in addressing the NP-Complete Traveling Salesman Problem (TSP). The modification of the similarity determination technique has yielded a network that effectively organizes cities into one of the most efficient routes achievable. While acknowledging the sensitivity of the technique to parametrization and the potential necessity for a substantial number of iterations in larger instances, the obtained results remain satisfactory and offer inspiration for exploring new applications within established techniques. This project contributes to the evolving landscape of problem-solving methodologies, showcasing the adaptability and potential of self-organizing approaches in tackling complex combinatorial optimization problems like the TSP.

VII. REFERENCES

[1] Karg, Robert L., and Gerald L. Thompson. "A heuristic approach to solving travelling salesman problems." *Management science* 10.2 (1964): 225-248.

[2] François, Antoine, Quentin Cappart, and Louis-Martin Rousseau. "How to evaluate machine learning approaches for combinatorial optimization: Application to the travelling salesman problem." *arXiv preprint arXiv:1909.13121* (2019).

[3] Ishaya, Jeremiah, Abdullahi Ibrahim, and Nassirou Lo. "A comparative analysis of the travelling salesman problem: Exact and machine learning techniques." *Open Journal of Discrete Applied mathematics* 2.3 (2019): 23-37.

[4] National Traveling Salesman Problems. www.math.uwaterloo.ca/tsp/world/countries.html.

[5] Kohonen, Teuvo. "The self-organizing map." Proceedings of the IEEE 78.9 (1990): 1464-1480.

[6] Zambito, Leonardo. "The traveling salesman problem: a comprehensive survey." Project for CSE 4080 (2006).

[7] Heins, Jonathan, et al. "On the potential of normalized TSP features for automated algorithm selection." Proceedings of the 16th ACM/SIGEVO Conference on Foundations of Genetic Algorithms. 2021.

[8] Harlan Crowder, Manfred W. Padberg, (1980) Solving Large-Scale Symmetric Travelling Salesman Problems to Optimality. Management Science 26(5):495-509. https://doi.org/10.1287/mnsc.26.5.495

[9] Hoffman, Karla L., Manfred Padberg, and Giovanni Rinaldi. "Traveling salesman problem." Encyclopedia of operations research and management science 1 (2013): 1573-1578.

[10] Mingozzi, Aristide, Lucio Bianco, and Salvatore Ricciardelli. "Dynamic programming strategies for the traveling salesman problem with time window and precedence constraints." Operations research 45.3 (1997): 365-377.

[11] Dorigo, Marco, and Luca Maria Gambardella. "Ant colony system: a cooperative learning approach to the traveling salesman problem." IEEE Transactions on evolutionary computation 1.1 (1997): 53-66

[12]Anmin Zhu and S. X. Yang, "An improved self-organizing map approach to traveling salesman problem," *IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003*, Changsha, China, 2003, pp. 674-679 vol.1, doi: 10.1109/RISSP.2003.1285655.

[13] Yuichi Nagata, Shigenobu Kobayashi, (2012) A Powerful Genetic Algorithm Using Edge Assembly Crossover for the Traveling Salesman Problem. INFORMS Journal on Computing 25(2):346-363. https://doi.org/10.1287/ijoc.1120.0506

[14] Han SH, Kim KW, Kim S, Youn YC. Artificial Neural Network: Understanding the Basic Concepts without Mathematics. Dement Neurocogn Disord. 2018 Sep;17(3):83-89. doi: 10.12779/dnd.2018.17.3.83. Epub 2018 Dec 13. PMID: 30906397; PMCID: PMC6428006.

[15] Wilson RI. Neural Networks for Navigation: From Connections to Computations. Annu Rev Neurosci.

2023 Jul 10;46:403-423. doi:
10.1146/annurev-neuro-110920-032645. PMID:
37428603.