Enhanced Self-Organizing Map Solution for the Traveling Salesman Problem

Joao P. A. Dantas[1], Andre N. Costa[1], Marcos R. O. A. Maximo[2], Takashi Yoneyama[3]

[1]Decision Support Systems Subdivision, Institute for Advanced Studies
12.288-001 – Sao Jose dos Campos – SP – Brazil

[2]Autonomous Computational System Lab, Aeronautics Institute of Technology
12.228-900 – Sao Jose dos Campos – SP – Brazil

[3]Electronic Engineering Division, Aeronautics Institute of Technology
12.228-900 – Sao Jose dos Campos – SP – Brazil

{dantasjpad,negraoanc}@fab.mil.br, {mmaximo,takashi}@ita.br

Abstract. *Using an enhanced Self-Organizing Map method, we provided sub-optimal solutions to the Traveling Salesman Problem. Besides, we employed hyperparameter tuning to identify the most critical features in the algorithm. All improvements in the benchmark work brought consistent results and may inspire future efforts to improve this algorithm and apply it to different problems.*

1. Introduction

The Traveling Salesman Problem (TSP) is one of the most studied routing problems within the combinatorial optimization field [Laporte 1992], which is defined as a salesman that must visit a series of cities once and only once, returning to the starting city [Brocki and Koržinek 2007]. Although coming from a relatively simple idea, finding the solution for the problem, i.e., the shortest path, is NP-Hard [Huang et al. 2017], thence the great interest to find efficient ways to solve it.

Due to the problem's complexity and to the fact that many of its applications require fast ways to solve it, it is common to employ heuristics to generate approximate (suboptimal) solutions to the TSP. Examples of these methods are Genetic Algorithm (GA), Simulated Annealing (SA), tabu search, and ant colony optimization [Xu et al. 2008, Calado and Ladeira 2011]. Another form to find approximate solutions to complex optimization problems is to use neural networks [Peterson 1990], which can be efficient due to their adaptability.

According to the literature, the first neural network solution for the TSP was the one presented in [Hopfield and Tank 1985], based on the minimization of an energy function. From that, many methods have been proposed to improve the Hopfield neural network [Xu et al. 2008]. Another use of neural networks for solving the TSP is the one proposed in [Kohonen 1998] through competitive unsupervised learning based on winner-take-all and winner-take-most algorithms, also called Self-Organizing Map (SOM), due to how the adaptation of the neurons works [Brocki and Koržinek 2007]. The SOM adjusts its neurons to fit the input cooperatively by inspecting the list of cities. This localized response to the input list generates a neighborhood preserving map, resulting in a near-optimal path [Xu et al. 2008].

In summary, most of the TSP heuristics solutions in the literature have used one of the following methods: Hopfield network, Kohonen's SOM, GA, and SA [Markovic et al. 2012]. In comparison with other heuristic methods, the SOM has presented low computation complexity and promising performance. Therefore, these neural networks have attracted researchers to explore and enhance their performance when applied to the TSP [Favata and Walker 1991, Budinich 1996].

In our work, the main contribution is to modify the method of SOM as proposed and implemented in [Martin 2018], adding some features and tuning hyperparameters. We offer these modifications to solve a particular problem involving TSPs varying in size from 50 to 200. The remainder of this paper is structured as follows: Section 2 provides details with respect to the experiment setting; Section 3 states the improvements on the method for this context; Section 4 presents the results obtained; Section 5 brings the conclusions and the possible future work.

2. Experiment

The proposed experiment aimed at predicting edges belonging to optimal solutions of $2{,}000$ two-dimensional Euclidean TSP ranging from 50 to 200 city nodes. We evaluate the solution considering the F1 score of the predicted adjacency matrix compared to the optimal solution. The F1 score is the harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0 [Zhang et al. 2015]. The relative contribution of precision and recall to the F1 score is equal, which is preferred over accuracy as this problem is often very imbalanced [Mele et al. 2021].

3. Improvements

In this section, we discuss the main improvements performed in the method adopted. We employ hyperparameter tuning to adapt the algorithm. Besides, we created two additional features to find solutions with better results.

3.1. Hyperparameter Tuning

When exploring the algorithm used in this work, we first identified its primary hyperparameters (population size, number of iterations, learning rate, and discount rates for the latter two) and their effects on the final scores. To understand the influence of each of these factors and find their best configuration, we employed a single-factor design to tune the proposed technique, i.e., varying approximately one feature at a time to examine its isolated influence. The search for the hyperparameters that better suit the dataset led us to the baseline algorithm, which received all modifications discussed next.

3.2. Additional Modifications

The first improvement to the baseline algorithm was to change the way to choose the first node considered. The standard technique randomly selects the starting point, which may lead the solution into a local minimum, depending on the location of this first node. In addition to the randomly chosen initial city, we forced the algorithm to start from the city in the centermost position and the furthest position from the centroid of all cities. After running the algorithm in these three different initial positions, we chose the one that presents the shortest path.

As the second modification, after employing the hyperparameters tuning and identifying the most significant feature as the population size, we improved the algorithm based on the variation of this hyperparameter in each SOM iteration. Figure 1 exemplifies the SOM iterations until the result with the shortest path is obtained. We used 20 different population sizes in each TSP, from 1 to 20 times the number of cities, and we calculated the route length of all of these TSP solutions, choosing the one with the shortest path. Indeed, that additional feature turned the process computationally more costly. However, it brought better results since the population size was constantly changing, and we chose the one that produced the shortest distance in all solutions.

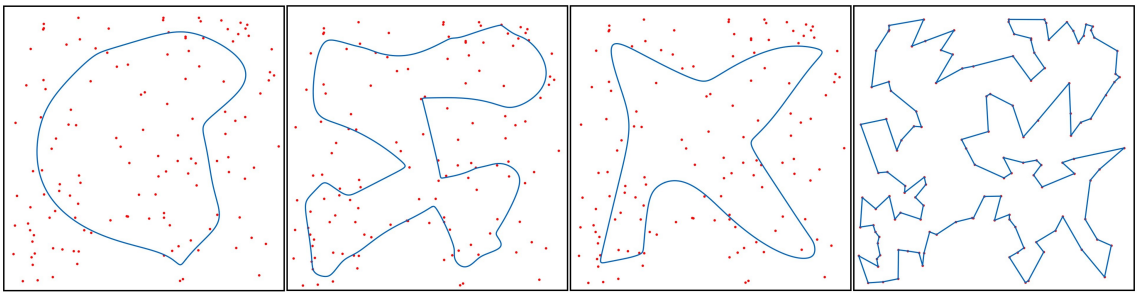

Figure 1. SOM algorithm iteration samples.

4. Results

Table 1 shows the F1 Scores results of the hyperparameters tuning and clarifies the importance of the population size feature. We chose as our baseline the following configuration: $100{,}000$ iterations, 0.9997 for the discount rate of initial neighborhood, 0.8 for the learning rate, 0.99997 for the discount rate of the learning rate, and 6 for the population size multiplier factor.

Table 1. Hyperparameters tuning with F1 scores

Number of Iterations	Discount Rate of Initial Neighborhood	Learning Rate	Discount Rate of Learning Rate	Population Size Multiplier Factor	F1 Score
100	0.9997	0.8	0.99997	8	0.06878
100,000	0.9997	0.8	0.99997	8	0.07800
100,000	0.9997	0.8	0.99997	16	0.07784
100,000	0.9997	0.8	0.99997	4	0.07855
100,000	0.9997	0.8	0.99997	2	0.07853
100,000	0.9997	0.8	0.99997	6	0.07885
100,000	0.9997	0.8	0.99997	5	0.07869
100,000	0.99997	0.8	0.99997	6	0.07854
100,000	0.997	0.8	0.99997	6	0.07834
100,000	0.997	0.01	1.00000	6	0.07769
100,000	0.9997	0.8	0.99997	7	0.07878

After choosing the baseline hyperparameter configuration, we employed the additional modifications described in subsection 3.2, achieving slightly better results with 0.07891 of the evaluation metric proposed. Therefore, these results show that, through subtle and well-planned changes in algorithms already consolidated by the scientific community, it is possible to obtain better results in classical optimization problems.

5. Conclusions

This work brings an enhanced way to use SOM to find suboptimal solutions for the TSP. Firstly, we employ hyperparameter tuning to identify the baseline algorithm configuration. Secondly, we combined two different modifications to the benchmark technique, leading to consistent route length results. For future work, we suggest additional improvements, mainly considering the stochastic aspects of the method, and advanced techniques to choose the best hyperparameters, which may bring better results to the SOM algorithm.

References

Brocki, Ł. and Koržinek, D. (2007). Kohonen self-organizing map for the traveling salesperson problem. In *Recent Advances in Mechatronics*, pages 116–119. Springer.

Budinich, M. (1996). A self-organizing neural network for the traveling salesman problem that is competitive with simulated annealing. *Neural Computation*, 8(2):416–424.

Calado, F. M. and Ladeira, A. P. (2011). Problema do caixeiro viajante: Um estudo comparativo de técnicas de inteligência artificial. *e-xacta*, 4(1).

Favata, F. and Walker, R. (1991). A study of the application of kohonen-type neural networks to the travelling salesman problem. *Biological Cybernetics*, 64(6):463–468.

Hopfield, J. and Tank, D. (1985). Neural computation solutions for tsp applications. *Biological Cybernetics*, 52:141–152.

Huang, L., Wang, G.-c., Bai, T., and Wang, Z. (2017). An improved fruit fly optimization algorithm for solving traveling salesman problem. *Frontiers of Information Technology & Electronic Engineering*, 18(10):1525–1533.

Kohonen, T. (1998). The self-organizing map. *Neurocomputing*, 21(1-3):1–6.

Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247.

Markovic, D., Madic, M., Tomic, V., and Stojkovic, S. (2012). Solving travelling salesman problem by use of kohonen self-organizing maps. *Acta Technica Corviniensis-Bulletin of Engineering*, 5(4):21.

Martin, D. (2018). Solving the traveling salesman problem using self-organizing maps. `https://github.com/DiegoVicen/som-tsp`.

Mele, U. J., Gambardella, L. M., and Montemanni, R. (2021). A new constructive heuristic driven by machine learning for the traveling salesman problem. *Algorithms*, 14(9):267.

Peterson, C. (1990). Parallel distributed approaches to combinatorial optimization: benchmark studies on traveling salesman problem. *Neural computation*, 2(3):261–269.

Xu, X., Jia, Z., Ma, J., and Wang, J. (2008). A self-organizing map algorithm for the traveling salesman problem. In *2008 Fourth International Conference on Natural Computation*, volume 3, pages 431–435. IEEE.

Zhang, D., Wang, J., and Zhao, X. (2015). Estimating the uncertainty of average f1 scores. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, pages 317–320.