Using Self-organizing Maps to Solve the Travelling Salesman Problem: A Review

STAVROS SARIKYRIAKIDIS
Department of Informatics Engineering, Alexander Technological Educational Institute of Thessaloniki, Alexander Campus Sindos Thessaloniki,
GREECE

KONSTANTINOS GOULIANAS
Department of Information and Electronic Engineering, International Hellenic University, Associate Professor, Alexander Campus Sindos Thessaloniki,
GREECE

ATHANASIOS I. MARGARIS
Department of Digital Systems, University of Thessaly, Adjunct Lecturer, Larissa-Trikala Ring-Road,
GREECE

Abstract: - This survey paper presents a collection of the most important algorithms for the well-known Traveling Salesman Problem (TSP) using Self-Organizing Maps (SOM). Each one of the presented models is characterized by its own features and advantages. The modes are compared to each other to find their differences and similarities. The models are classified in two basic categories, namely the enriched and hybrid models. For each model we present information regarding its performance, the required number of iterations, as well as the number of neurons that are capable of solving the TSP problem. Based on the experimental results, the best model is identified for different occasions. The paper is a good starting point for anyone who is interested in solving TSP with SOM and desires to grasp a lot about this renowned problem.

Key-Words: - Self-Organizing Maps (SOM), Travelling Salesman Problem (TSP), Neural networks, Algorithms.

Received: April 29, 2022. Revised: January 19, 2023. Accepted: February 16, 2023. Published: March 7, 2023.

1 Introduction

The Traveling Salesman Problem (TSP) is among the most widely studied combinatorial optimization problems, and many studies have been conducted in an attempt to identify a reliable solution, a solution that is simple to state but very difficult to find. One of the first solutions was found during the 1950s by Dantzig, Fulkerson, and Johnson [1]. The objective of the problem is to find the shortest possible tour through a set of N vertices, and in such a way that each vertex is visited exactly once. This problem is classified as NP-hard problem [2] and there are two reasons for this. First, there are no quick solutions and second, the complexity of calculating the most optimum route increases when more destinations are inserted to the TSP. There are many exact and heuristic algorithms that have been devised in the field of operations research (OR) to solve the TSP. It is obvious that the number of possible tours increases explosively as the number of cities increases.

A Self-Organizing Map (SOM) is a type of Artificial Neural Network (ANN) that uses unsupervised learning, in order to build a 2D map of the problem space. The key difference between a self-organizing map and other approaches to problem-solving, is that a self-organizing map uses competitive learning rather than error - correction learning, such as the well-known backpropagation algorithm with gradient descent. A self-organizing map can generate a visual representation of data on a hexagonal or rectangular grid. Applications of SOM include among others, meteorology, oceanography, project prioritization, as well as, oil and gas exploration. A self-organizing map is also known as a self-organizing feature map (SOFM) or a Kohonen map [3], [78]. The Self-Organizing Map (SOM) networks, originally proposed by Kohonen, solve the TSP via unsupervised learning. The

application of the artificial neural network approach in solving TSP, was initiated by Bernard Angeniol, Gaël de La Croix Vaubois, and Jean-Yves Le Texier in 1988 [27]. The calculations are based on the lowest value obtained from the energy function and have been improved until the SOM algorithm is used to solve this problem.

A two-layer network with a two-dimensional input unit and m output units is utilized to apply SOM to the TSP. A ring expanding toward the locations of the cities can be used to represent how the network evolves. The coordinates of cities serve as the input data, and the coordinates of the ring's points serve as the weights of the nodes. The cities are presented in a random sequence to the network, and nodes compete based on Euclidean distance. The node that is closest to the city that is being presented is the winner node. Also, the winning node and its neighbors advance toward the city that is being presented in each iteration utilizing the neighborhood function in conjunction with an updated function. The winner's neighbor nodes' influence on a city is determined by the neighborhood function. It should be observed that updating nearby nodes in response to inputs exerts a force that preserves nodes close to one another and establishes a mechanism for condensing the created tour.

Fig.1 shows a schematic view of a SOM-like network for the TSP. A ring of output neurons, denoted by 1, 2, to M, is employed to characterize the feature map. The input neurons, receiving the data of the input city (say, coordinate values), are fully connected to every output neuron. **Fig.2** illustrates the initialization step, an intermediate step, as well as the final state of the training process; the solution is represented in the fourth graph.

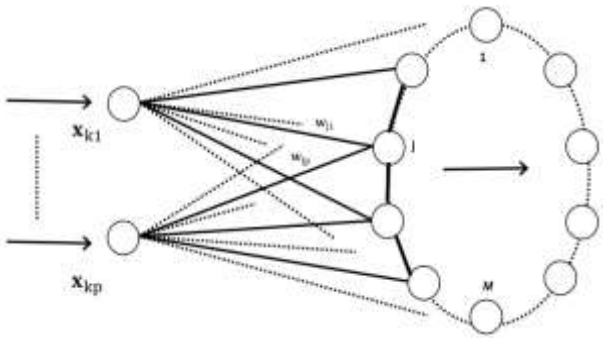

Fig. 1: A schematic SOM-like network for the TSP. M is the number of output neurons and p is the number of input neurons, say 2, for the 2-dimensional Euclidean TSP [50].

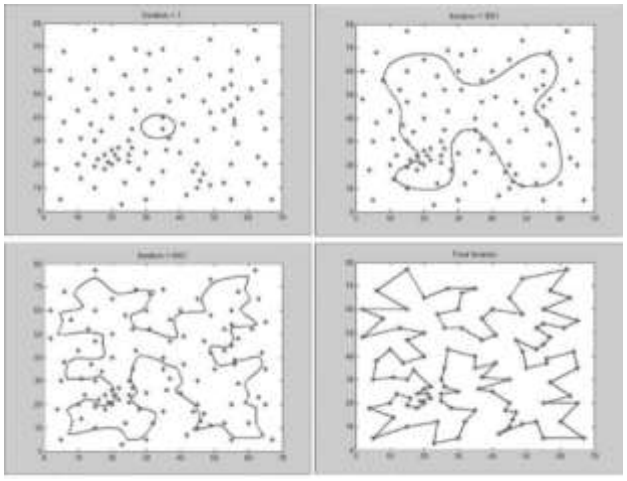

Fig. 2: The progress of applying the SOM algorithm can be displayed in two-dimensional planes as shown above. The dots are the cities of a TSP which are the coordinates of a location. The circle in the left–top picture, shows the ring that usually starts from a random location. After that, the main algorithm of SOM starts tracking the next location, and the convex-hull spreads across the axis x & y during the algorithm execution. The final iteration shows the best solution for the TSP map [45].

To solve a TSP, the algorithm needs the feeding of data in the input layer as a TSPLIB file. TSPLIB is a library of sample instances for the TSP (and related problems) emerged from various sources and of various types. An example of a library is ‘qa194.tsp’ with the filename identifies a country / dataset as well as the number of the cities / dots of this dataset (in this example the data are associated with the country of Qatar and a number of 194 cities). Other examples of these TSPLIF filenames are ‘pbc1173’ and ‘bier127’.

After this short description regarding the Self-Organizing Map and the Traveling Salesman Problem, the paper is organized as follows: In Section (2), the different categories of models that solve the problem are presented. On the other hand, in Section (3) the focus is given to the subcategories of enriched models. Section (4) presents the Hybrid models as well as their special types. The objective of Section (5) is the presentation of the Multiple Salesman TSP models. Section (6) is a brief review of a similar problem which is called the Vehicle Routing Problem (VRP) and is considered as a generalization of the Travelling Salesman Problem (TSP), and the TSP with drones (TSP-D) is also presented. Least, but not last, in Section (7), some computational results are presented.

2 Categories

In this section, the different models of SOM that solve the TSP are introduced. The models are classified according to the complexity of the underlying algorithm used by each one of them. Some models are more complicated to be constructed, and some others require fewer modifications. The literature search revealed two main categories of such models. The first one is the Enriched SOM model which consists of the basic SOM with some upgrades, while the second category is identified by the Hybrid SOM model.

A hybrid algorithm is defined as an algorithm that combines two or more other algorithms that solve the same problem. This statement does not describe a combination of multiple algorithms that solve different problems – in fact, many algorithms can be considered as combinations of simpler pieces – but, instead, a combination of algorithms that solve the same problem, but they differ in characteristics, such as the performance. The experience has shown that these hybrid methods lead to higher performance, better computational results, and more flexibility on large-scale data. On the other hand, the disadvantages of hybrid methods are mostly associated with the complexity of the algorithms regarding the construction of the primary model. The reason for this is that the network is a combination of two or more methods, and therefore, it is difficult to build. Also, we must note that some algorithms are characterized by poor performance regarding the time spent to finish. Hybrid algorithms are further divided into greedy, evolutionary, genetic, memetic, chaos, and fuzzy algorithms.

In a more detailed description, a greedy algorithm is an algorithmic paradigm that builds up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. This means that the class of problems in which the selection of the local optimum, also leads to the global optimum, are the best fit for greedy algorithms. The advantage of using this type of algorithm, is that the solutions associated with smaller instances of the problem, are generally straightforward and easy to understand. On the other hand, the disadvantage of greedy algorithms is that it is entirely possible that the most optimal short-term solutions may lead to the worst possible long-term outcome.

On the other hand, an evolutionary algorithm is nothing more than an evolutionary-based computer application that is capable of solving problems by employing processes that mimic the behaviors of living things. As such, it uses mechanisms that are typically associated with biological evolution, such as reproduction, mutation, and recombination. However, the drawback of this type of algorithm is that it requires a lot of computational power. It is interesting to note, that evolutionary algorithms are behaved in a Darwinian-like natural selection process; the weakest solutions are eliminated, while the stronger, more viable options, are retained and re-evaluated in the next evolution with the goal to arrive at optimal actions to achieve the desired outcomes.

A genetic algorithm is a heuristic search method used in artificial intelligence and computing. It is employed to find optimized solutions to search problems based on the theory of natural selection and evolutionary biology. Genetic algorithms are an excellent tool for searching through large and complex data sets. The most serious disadvantages of genetic algorithms, is the high cost of their implementation, as well as the difficulties associated with the debugging process. Furthermore, on some occasions, they can be difficult to understand.

The memetic algorithms can be viewed as a combination of a population-based global technique with a local search procedure made by each one of the individuals. These algorithms are special types of genetic algorithms with a local hill climbing. The memetic algorithms, like genetic algorithms, are a population-based approach.

A chaotic self-organizing map can be produced by replacing the linear neural units of the conventional self-organizing map with neural units capable of producing chaos.

Fuzzy logic is an interesting approach that allows the association of multiple possible truth values with the same variable. Fuzzy logic attempts to solve problems using an open, imprecise spectrum of data and heuristics that makes it possible to obtain accurate conclusions. Fuzzy logic is designed to solve problems by considering all available information and making the best possible decision given the input.

Simulated annealing (SA) is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is a metaheuristic that allows the approximation of global optimization in a large search space for an optimization problem. It is often used when the search space is discrete.

Elastic net linear regression uses the penalties from both the lasso and ridge techniques to regularize regression models. The technique combines both

the lasso and ridge regression methods, by learning from their shortcomings, to improve the regularization of statistical models.

Finally, the paper is also considered the multiple salesman problem, that solves the TSP with more than one salesman as the classic one.

3 Enriched Models

3.1 Convex-hull Property Models

The convex-hull of a set of points in a two-dimensional Euclidean space, is defined as the smallest (in terms of area) convex polygon that includes all the points in the set. This subsection presents models that are based on this feature.

Guan et al. [5] proposed the so-called Topology Preserving SOM (TOPSOM). The process of gaining knowledge in TOPSOM includes the appropriate arrangement of the output layer, the mapping of neurons to the input nodes that represent cities, as well as the identification of a minimum path that passes from all cities and then returns back to the starting point. The goal of this method is the satisfaction of two constraints, namely, the preservation of index topology and the convex hull. The main steps of TOPSOM are the following:

1. Normalization and Random Initialization.
2. Winner selection.
3. Improved Hebbian Learning process.
4. Final tour construction.

Zhang et al. [6] proposed the so-called overall - regional competitive SOM (ORC-SOM). In this model there are two rules that describe a competition process, namely, the overall competition, and regional competition. These rules are introduced into the circular standard SOM competitive learning algorithm: the purpose of the overall competition is to outline the tour, while regional competition is responsible for refining it in order to obtain the optimal and / or near-to-optimal solution of the TSP. The ORC-SOM model can be considered as a simple extension of a standard SOM, with an overall - regional competitive learning rule embedded. The ORC-SOM learning algorithm for solving the 2-D Euclidean TSP of N cities uses as input the coordinates of N cities in two-dimensional space, returns as output an optimal or near-to-optimal tour, and uses as parameter a circular SOM with two inputs and suitable M output neurons with M >N.

Xu et al. [7] proposed a Convex-Hull SOM (CHSOM) model. The proposed SOM is first initialized with cities on the convex-hull of a problem. In order to keep the convex-hull property of the problem, a principle of neuron creation and deletion is introduced into the proposed SOM. The algorithm is described as follows:

Initialization method: The SOM initially has k neurons where k is the number of cities on the convex hull. In other words, there is an 1-to-1 mapping of the set of neurons to the associated set of cities. Furthermore, the value associated with each neuron is the same as the value of the corresponding city.

Neuron creation: During learning, only those cities that are not located on the convex-hull are fed into the input neurons. First, one city is fed into the input neurons. Then, a winner is selected based on the competing rule. In this model, there are two cases where the creation of a neuron is required. The first one is when an initializing neuron becomes a winner. In this case, a neuron is created and is associated with the same value as the initializing neuron. In the next step, the newly created neuron is inserted into the ring as the left neighbor or the right neighbor of the winner. **Fig.3** shows that principle.

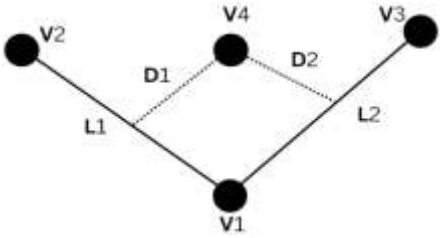

Fig. 3: The principle that allows the determination of the position for the insertion of a newly created neuron. In this figure, V1 is the winner, and V4 is the input city. Furthermore, V2 is the left neighbor of V1, while V3 is its right neighbor [7].

Neuron deletion: A node is deleted if it has not been chosen as the winner by any city during three complete surveys.

Neighborhood function: This function is described by a mathematical equation. Once a neighbor is an initializing neuron, the neighborhood updating process skips this neuron. In other words, only those neighbors that are not initializing neurons are updated.

Yang & Yang [8] proposed an Expanding Property SOM (EPSOM) model. The process used in this method is described as follows: Within each iteration, the excited neuron is drawn towards the input city as well as towards the convex hull. The former adaptation, together with the cooperative adaptation of neighbor neurons, will gradually

discover the topological neighborhood relationship of the input data. Meanwhile, the latter adaptation, together with the cooperative adaptation, can approach the convex hull property of the TSP. To do so, it is reasonable to push the excited neuron away from the center of the city. The procedure to implement the idea is described as follows:

1. Take the coordinates of the cities.
2. Randomly set the weight vectors and nullify the iterations.
3. Randomly select a city and feed it to the input neurons.
4. Find the winning neuron according to the Euclidean metric.
5. Train the neuron and its neighbors.
6. Update the width parameter and the learning parameter according to a predefined decreasing scheme, and if the learning is not terminated, go to Step 3.
7. Calculate the activity value of each city.
8. Order the cities by their activity values and then form a tour for the TSP.

Kwong-Sak et al. [9] proposed an Expanding SOM (ESOM) model. The basic idea of the ESOM is to include implicitly the convex hull property in its learning rule, with little additional computation. The convex hull property is acquired gradually, as the topological neighborhood among cities is being inspected and preserved. Therefore, this ESOM model tries to satisfy sufficient and necessary conditions for optimal tours. This is realized in the following way: In a single update iteration, besides drawing the excited neuron towards the input city, it is expanded towards the convex hull. It is worth noting that the approach of the convex hull can be approximated by moving away from the center of the cities. The motivation of the developed ESOM model is to approximate an optimal tour by trying to satisfy a sufficient condition (namely, the neighborhood preserving property), as well as a necessary condition (namely, the convex-hull property). The steps of the algorithm are the following:

1. Map all the city coordinates into a circle.
2. Randomly set the weight vectors.
3. Feed a random city to the input neurons.
4. Train neurons and their neighbors.
5. Update the parameters.
6. Calculate the activity value of each city.
7. Order the cities by their activity values.

3.2 Ring Topology Models

Bai et al. [10] proposed a Modified Growing ring SOM (MGSOM) model that initializes a random circular ring of neurons, it stretches the neurons to the city's location, and finds the closest node using Euclidean distance. The advantages of this model are the easy implementation, the fast computation, the robust applicability, and the production of good solutions. The MGSOM approach is briefly described as follows:

1. Initialization.
2. Randomizing.
3. Parameter adaptation.
4. Competition.
5. Adaptation.
6. Insertion of a node.
7. Step 4 or 8 depending on the result of a condition.
8. Convergence test.

The steps for a better-quality solution are the following:

1. Run 10 simulations with $t_{min} = 20$ iterations by using the MGSOM.
2. Select the best solution emerged from the simulation runs of Step 1.
3. Run a new simulation with $t_{min} = 10$ iterations that uses as initial weight matrix the weight matrix associated with the best solution found in Step 2.

The complexity of the improved algorithm (for each value of t) is evaluated as follows: for each city, there are 0.3m (average neighbor length) operations for the calculation of the neighborhood function. Therefore, the time complexity function for each iteration, can be defined as $T(m,n) = n(0.3m)$, while the complexity reduction is guided by the variance of the neighborhood function. The analysis of the variance evolution reveals that the algorithm processing time decreases fast (due to the selective update). This feature is mainly attractive when processing large datasets.

Bai et al [11] proposed an Efficient Growing ring SOM (EGSOM) model that uses a ring topology, as the previous one, namely, a network in which the nodes are connected in a closed loop configuration. The adjacent pairs of nodes are joined via direct connections, while the remaining pairs of nodes are indirectly connected, meaning that the data passes through one or more intermediate nodes. The structure of the EGSOM can grow according to each node's winning number and the states of its neighbors. Each city location corresponds to an input, and it is randomly applied to the EGSOM. The advantage of this model is that it is easy to implement, fast to compute, and produces good solutions. The number of nodes need only grow to

1.5*n* and the algorithm processing time is fast. The MGOM and the EGSOM models have similar specifications. Both models are based on a ring topology, and therefore, their computational results are close to each other.

Zhang et al. [12] proposed a modified heuristic approach (MSTSP) model. MSTSP has the same architecture and similar philosophy as the MGSOM and EGSOM. The main difference is that in the MSTSP the increase of the number of nodes does not trigger the insertion of a new node. The operation of the MSTSP model, is briefly described as follows:

1. Initialization.
2. Randomizing.
3. Parameter adaptation.
4. Competition.
5. Adaptation.
6. Step 4 or 7 depending on the result of a condition.
7. Convergence test.

The complexity of the MSTSP algorithm for each iteration (each value of k) may be evaluated as follows: For each city, there are m operations to find the closest node to that city, and 0.2m (average neighbor length) operations for the calculation of the neighborhood function. Therefore, the time complexity function for each iteration may be given by $n(m + 0.2m)$, and the time complexity function of the MSTSP algorithm for each simulation may be given by equation (1).

$$T(m,n) = (210) \cdot n(m + 0.2m). \quad (1)$$

Furthermore, the complexity reduction is guided by the neighborhood function variance. The analysis of the variance evolution reveals that the algorithm processing time decreases fast (due to the selective update). This feature is mainly attractive when processing large data sets.

3.3 Heuristic Enriched Models

Dantas et al. [13] proposed a SOM (Enhanced SOM Solution) model. This model is an improvement of the classic SOM in the sense that it employs hyperparameter tuning to adapt the algorithm and create two additional features to find solutions with better results. In the first step the primary hyperparameters (population size, number of iterations, learning rate, and discount rates for the latter two) are identified and then used to evaluate and their effects on the final scores. To understand the influence of each one of these factors and find their best configuration, a single-factor design is employed to tune the proposed technique, meaning that each feature is varied individually, one feature at a time. The search for the hyperparameters that better suit the dataset finally leads to the baseline algorithm. The first improvement to the baseline algorithm is to change the way of choosing the first node considered. In addition to the randomly chosen initial city, the algorithm is forced to start from the city in the centermost position and the furthest position from the centroid of all cities. As the second modification, after employing the hyperparameters tuning and identifying the most significant feature as the population size, the algorithm improved based on the variation of this hyperparameter in each SOM iteration. The solution is evaluated using the F1 score of the predicted adjacency matrix compared to the optimal solution. The F1 score is the harmonic mean of precision and recall. The best value of this score is the value of 1, while the worst value is the value of 0. [14]. It is chosen as the baseline for the following configuration: 100000 iterations, 0.9997 for the discount rate of the 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.

Modares et al. [15] proposed a SOM algorithm for solving a relatively efficient TSP problem. By simply inspecting the input city data for regularities and patterns and then adjusting itself to fit the input data through cooperative adaptation of the synaptic weights, such a network creates the localized response to the input data, thus reflecting the topological ordering of the input cities. In this way, this neighborhood preserving map, results in an expected tour of the TSP under consideration. From each city, the resultant tour tries to visit its nearest city. The shortest sub-tour can intuitively lead to a good tour for the TSP. This algorithm is one of the oldest SOM algorithms that solves the TSP.

Matsuyama [16] proposed another SOM model. This model uses a competitive learning process that causes just one neuron, or alternatively, a small group of neurons to respond to a given input. In this way, the self-organization of entire neural networks can be achieved. When this self-organization process is applied to various kinds of traveling salesman problems in a Euclidean space, a good approximation or the true solution is obtained. The algorithm is used as the training method for a neural network arranged in a closed loop, a sequential update that looks at the position vector of each city, one at a time. In this case, it uses symmetrical connections between neurons. The required number

of neurons is approximately a linear function of the number of cities.

Schabauer et al. [17] proposed a Structural Data Parallel approach (SDP). This approach allows users, who are inexperienced in high-performance computing, to increase the performance of neural network simulation by parallelization. The paper describes an unsophisticated approach, whose results are used then in the main sophisticated approach. The unsophisticated approach is based on the execution of an application written in Fortran 90 that tries to parallelize it right away. It uses the sequential program as the base for parallelization. It just tells the system how the data has to be distributed. Based on the SDP method for parallelizing, the computation of the TSP-SOM program consists of three parts:

1. Get coordinates of cities by random.
2. Get starting weights of the Kohonen network by constructing a circle.
3. Train the network.

The training phase, which is the main loop of the program, consists of the following steps:

1. Calculate a vector of distances.
2. Find the winner-neuron.
3. Update the matrix of weights.

In the sophisticated approach, there are identified the following five issues for specific attention and analysis:

1. Local cache misses.
2. Workload balancing.
3. Latency of communication.
4. Communication throughput.
5. Locality of data.

E.M. Cochrane and J.E. Beasley proposed a co-adaptive neural network (CAN). The CAN [18] is another approach to solving the TSP using the SOM. Its main feature is using a so-called cooperation phase. Cooperation means that during the learning process, the identification of the winning neuron is based on the cooperation of various inputs. The competition and the cooperation phases differ in how neurons react, when one neuron is the winner for various inputs. In the competition phase, if the neuron is a winner for just one time, only the neighborhood is moved. On the other hand, if the neuron has been chosen more than once, none of the neurons is moved. In the cooperation phase, none of the winning neurons are allowed to move more than once. The neighbors are not allowed to move as well. The learning method is based on both phases, beginning with the competitive phase, and switching at some point to the cooperation phase. In this algorithm, the neuron initialization process is the same as in the basic SOM (initially, the neurons lie on a ring). The winner selection is also similar, except one difference that is supposed to improve the speed: the winning neuron is searched for in the vicinity (as measured around the ring) of the neuron that was the winner in the previous iteration. In every βth iteration, the set from which the winning neuron is searched for, is expanded to the whole set of neurons. In reference [19], which is the original paper on the CAN algorithm, this model was compared to many other models. The main conclusion of this comparison is that the CAN algorithm is far superior with respect to the works of Matsuyama, Guilty Net [20] and Elastic Net [21], with regard to all measures of solution deviation from optimal and computation time. Also, the CAN algorithm produces results that are superior to Somhom with regard to solution deviation from optimal. Moreover, the results are produced in a faster time. The CAN is approximately 10 times faster than Somhom.

Faigl [22] proposed a modified CAN model. This model is similar to the original CAN, but with some modifications that give better results. The first one is an initialization method similar to the one used in the original model, in the sense that a ring topology is initialized as a small circle around the centroid of the cities. The original adaptation rule is also modified to consider the b-condition, and it is combined with the Multi-Scale Neighborhood Functions (MSNF) used by Murakoshi and Sato [23]. The co-adaptive net requires a higher number of adaptation steps, while the insertion of MSNF in the neighborhood function, increases the solution quality and the number of required steps. This modification of the adaptation rule to support the b-condition, can be used without decreasing the neighborhood size. The terminating condition of the network adaptation procedure in all algorithm variants is defined as $G < 0.01$. The gain-decreasing rate $a = 0.1$ provides almost the same solution quality (about one or two percent worse) as the original algorithm, being more than four times faster, compared with the original one associated with the value $a = 0.02$. For problems with less than fifty cities, the modified co-adaptive net algorithm provides better results. The co-adaptive net uses the number (id) of the winning neuron. The algorithm avoids adaptation of the winners and neighboring

nodes, meaning that the nodes are moved with less frequency.

Aras et al. [24] proposed a Kohonen network that incorporates explicit statistics (KNIES). More specifically, the KNIES method takes full advantage of statistics. The network consists of a fixed number of M neurons. The input to the network is the coordinates of each city node, which belongs to the set of cities. The city nodes are mapped to the neurons of the ring structure, and the order of the neurons in the ring represents the order of the city node's traversal. Since KNIES makes full use of the advantages of SOM, it can maintain the neighborhood structure between city nodes. The primary difference between the SOM and the KNIES, is the fact that every iteration in the training phase includes two distinct modules — the attracting module and the dispersing module. In the attracting module, a subset of the neurons migrates toward the data point that has been presented to the NN. This phase is essentially identical to the learning phase of the SOM. However, in what follows, the rest of the neurons which have not been involved in the attracting module, participate in a dispersing (repellent) migration. Indeed, these neurons now move away from their current positions in a manner that ensures that the global statistical properties of the data points are resident in the neurons. Thus, although in the SOM the neurons individually find their places both statistically and topologically in an asymptotic way, in the KNIES they collectively maintain their mean in such a way that it represents the mean of the involved data points. The KNIES was compared to ESOM and ORC-SOM.

Aras et al. [25] proposed a Kohonen-like Decomposition model (KNIES DECOMPOSE). KNIES DECOMPOSE is based on the foundational principles of Kohonen's SOM, and on its variant named KNIES described above. The final tour is obtained by combining Hamiltonian paths that KNIES HPP (KNIES Solution to the Hamiltonian Path Problem) constructs for the clusters determined according to Kohonen's clustering method. The Euclidean traveling salesman problem (TSP) is a close cousin of the Euclidean Hamiltonian path problem (HPP). The decomposition into subproblems, is a widely used strategy in solving large mathematical programs. It is easier to solve the subproblems, because the size of each subproblem is much smaller, and their solutions can usually be combined to approximate the solution of the original problem. The strategy to reduce the complexity of a large-scale traveling salesman problem instance, is the decomposition or partitioning into smaller HPP subproblems, which are easier to solve. In essence, it solves a large TSP by spawning smaller HPPs. Once these smaller HPP instances are individually solved, the solution to the original TSP is obtained by patching the solutions to the HPP subproblems. The partitioning into smaller HPP subproblems is achieved by clustering the cities of the original problem, in a way that the structural properties of the problem instance are preserved. The steps of the KNIES DECOMPOSE are:

1. Partition the cities into clusters for a given number of clusters.
2. Determine an order for visiting the clusters.
3. Identify an entering and a leaving city for each cluster, obeying the ordering obtained in Step 2 and forming the bridges between clusters.
4. Find a Hamiltonian path in every cluster between the entrance and exit cities.
5. Connect the Hamiltonian paths by using the bridges obtained in step 3.

Faigl [26] proposed a Growing Self-Organizing Array (GSOA), a novel unsupervised learning procedure for routing problems. Its main principles follow the existing work on SOM for the TSP, but it is mostly motivated by data collection planning, where a robotic vehicle is requested to collect data from a given set of sensing sites. In this type of problem, it may not be necessary to visit the sites precisely, and the robotic vehicle may use remote sensing or wireless communication to collect the required data. The procedure consists of three steps:

1. Winner node determination.
2. Adaptation of the winner.
3. Extraction of the solution after.

Angeniol [27] proposed another SOM model, with the following features: Firstly, the total number of nodes and connections in a connectionist parallel implementation, is proportional only to the number of cities in the problem, thus scaling very well with problem size, Secondly, the model requires the tuning of only one parameter, which directly controls the total number of iterations, and thirdly, the typical values of this parameter ensure a good, near-optimum solution in a reasonable time for the performed simulations. More specifically, only the gain decrease parameter a must be adjusted, with the obtained results to be almost insensitive to it. A low value of the parameter a gives a better average. However a higher value has the advantage that the optimum is sometimes reached, while, at the same

time, leads to a good solution after several tries, in less time than the case in which only one try is performed with a low value of a. In either case, a good average solution (less than 3% greater than optimum) may be obtained in 2 seconds using classical hardware. Simulations were performed on small sets of cities taken from Tank and Hopfield and from Durbin and Willshaw.

Kitaori et al. [28] proposed methods that enforce the ability of Kohonen's self-organizing features map (SOFM) to solve optimization problems, with the focus on the solution of the traveling salesman problem (TSP). As is well known, the conventional SOFM can solve the TSP. However, the solution found, is not the optimum solution, because the path intersects itself. On the other hand, the proposed methods of Kitaori et al, keep the path from always intersecting itself. The use of these methods to the weight adaptation rule, improves the length of the identified path, in the sense that the iteration execution time is decreased, while, at the same time, the convergence rate is increased. One is called constraint condition, which prevents the tour from crossing itself all the time. The other method is called active area and refresh, and its usage is for creating and deleting cells that will put the proper number of cells in the right place. A disadvantage of this approach is that as the number of cities increases, the convergence rate of the model gets worse, leading to inaccurate results regarding the path length.

Budinich [29] applied unsupervised learning to an unstructured neural network to get approximate solutions to the traveling salesman problem. For a training set of 50 cities, this algorithm performs like the elastic net of Durbin and Willshaw (1987), while, if the number of cities increases, the algorithm leads to better results than the ones associated with simulated annealing for problems with more than 500 cities. Furthermore, in all the tests, this algorithm requires only a fraction of the execution time taken by simulated annealing.

Kim et al. [30] proposed another efficient SOFM algorithm, for solving large-scale TSP. In this algorithm, a winning neuron for each city is not duplicated but instead, it is excluded in the next competition. This approach to TSPs using SOFM is very promising due to the following features. First, the total number of neurons and connections is small and linearly proportional to the number of cities of the TSPs, thus giving good scalability for the problem size, and makes the algorithm suitable for parallel hardware implementation. Second, the proposed algorithm requires only N output neurons and 2N connections, where N is the number of cities, while it does not require a dedicated procedure for the creation and deletion of neurons, giving thus 30% faster convergence than the conventional algorithm for 30-city TSPs. Finally, the actual potential of the proposed algorithm leads to near-optimal solutions in 92% of the total trials within a reasonable time. Simulation for the 1000-city TSPs also gives good and promising results for the proposed algorithm.

Mohamed [31] proposed a cost minimization approach. In this approach, a route-first cluster-second heuristic uses a SOM network with varying parameters, as well as a greedy algorithm to minimize the cost or distance of the TSP-D (TSP with drones) solution and measure the impact that the one parameter (cost or distance) has on the other. The study was conducted using benchmark TSP data sets of varying sizes that were adapted to fit the TSP-D model. The study shows that while a cost-minimization approach can yield distance savings of up to 21%, a distance-minimization approach can actually result in higher costs by up to 54%. Additionally, the SOM algorithm employed in this study, was shown to generate significant cost savings of up to 10.4% for networks consisting of 194 customers or more.

Markovic et al. [32] proposed a Waste Disposal SOM for solving the waste disposal problem with a size of 20 (WDS). This is a real life problem, solved for the city of Nis. On the other hand, Lobo [33] proposed a SOM on Marine patrol. This method is based on a Self-Organizing Map (SOM) solution for the Travelling Salesman Problem (TSP), even though there are significant changes. The locations of reported Search and Rescue (SAR) requests, together with the locations of reported occurrences of illegal fishing activities, are used as guidelines for designing the path to be followed. However, instead of forcing the patrol routes to pass exactly in those locations, as would happen in a TSP, the proposed method uses the locations as density estimators, in an attempt to identify the places where the patrol effort should be focused. In the next step the algorithm obtains a patrol route that passes through the areas with greater density. This algorithm uses some data from the Portuguese Navy.

Faigl et al. [34] proposed an application on Obstacles + SOM, namely, a SOM algorithm that solves the TSP with obstacles in the map. The difficulty of SOM application to the non-Euclidean

TSP has been noted by several authors of SOM algorithms for the TSP. The presented experimental results show that even a simple approximation of the shortest path among obstacles can be used, with the SOM algorithm to be able to find a solution with competitive quality to a solution of the Euclidean TSP of similar problem size. The approximation of the shortest path used here, is more computationally intensive than the computation associated with the Euclidean distance. The algorithms used in this work include a visibility graph algorithm, a convex polygon partitioning algorithm, a point location algorithm and a straight walk procedure in a convex partition.

Faigl & Preucil [35] proposed a Multi-Goal Path, namely, a SOM approach for the multi-goal path planning problem with polygonal goals. The objective of the problem is to find the shortest closed collision-free path for a mobile robot operating in a planar environment represented by a polygonal map W. The requested path must visit a given set of areas, where the robot takes measurements to find an object of interest. Neurons' weights are considered as points in W and the solution is found as the approximate shortest paths connecting the points (weights). The proposed SOM has fewer parameters than a previous approach based on the self-organizing map for the traveling salesman problem. Moreover, the proposed algorithm provides better solutions within less computational time for problems with a high number of polygonal goals. The planning problem for point goals can be formulated as the well-known traveling salesman problem (TSP).

Faigl et al. [36] proposed an approach on Orienteering Problem with Neighborhoods + SOM. This work addresses the Orienteering problem (OP) by the unsupervised learning of a self-organizing map (SOM). This approach is based on the solution of OP with a new algorithm based on SOM for the Traveling salesman problem (TSP). Both problems are similar in finding a tour visiting the given locations; however, the OP is focused on the determination of the most valuable tour that maximizes the rewards collected by visiting a subset of the locations, while keeping the tour length under the specified travel budget. The proposed stochastic search algorithm is based on unsupervised learning of SOM and it constructs a feasible solution during each learning epoch. The reported results support the feasibility of the proposed idea and show that the performance is competitive with existing heuristics. Moreover, the key advantage of the proposed SOM-based approach is the ability to address the generalized OP with Neighborhoods, where rewards can be collected by traveling anywhere within the neighborhood of the locations. This generalization of the problem fits better the data collection missions with wireless data transmission and it allows to save unnecessary travel costs to visit the given locations.

Faigl [37] also proposed another approach on Orienteering Problem + SOM. In this case, the proposed approach is able to directly utilize a non-zero communication radius, and thus it is capable of finding solutions with higher rewards than the demanding metaheuristics for the Orienteering Problem (OP), as well as and Team Orienteering Problem (TOP), and more importantly, without considering the neighborhoods. Therefore, the proposed SOM-based approach can be considered as a construction heuristic and combined with evolutionary techniques and existing metaheuristics to improve solutions not only in the OP and TOP, but mainly in orienteering problems with neighborhoods which are suitable formulations for robotic information gathering scenarios with single or fleet of robotic vehicles. The main difference between the ordinary OP and its generalization, namely, the Orienteering Problem with Neighborhoods (OPN), is that, in addition to the determined subset of the sensors providing the most valuable measurements, it is also required to determine the most suitable waypoints from which the measurements (rewards) from the sensors can be collected. Both formulations (the OP and OPN) share the main challenge of orienteering problems, which is the determination of the subsets of the locations according to the tour visiting them with respect to the given travel budget.

Faigl & Vana [38] proposed an approach on Dubins traveling salesman problem (DTSP + SOM). The main difficulty of Dubins TSP arises from the fact, that it is necessary to determine the sequence of visits to the locations, together with headings of the vehicle at these locations. The proposed paper is the first SOM-based solution of the Dubins TSP, that includes challenges of the underlying combinatorial TSP with the continuous optimization of the headings at the locations. Even though the results do not reveal significantly better solutions of SOM, than a more computationally demanding memetic algorithm, however, the results support the feasibility of the proposed idea and lead to better scalability for larger and denser problems.

Wang et al. [39] proposed a massively parallel cellular SOM to address large-scale Euclidean

TSPs. The implementation of the model is done on a GPU CUDA platform, with evaluations performed on 52 large-size TSP instances with up to 85900 cities. The proposed model deals with a very simple procedure close to the original standard SOM. The key point is the massive parallelism that should allow a substantial reduction in execution time in the future, as thousands of multi-cores are available in a single chip.

4 Hybrid Models

After the short presentation of the enriched models let us now present the hybrid models found in the literature. These models are the following:

4.1 Greedy Models

Nuriyev et al. [40] proposed a Self-Organizing Iterative approach. The proposed algorithm is in fact, a combination of a Nearest Neighbour Algorithm from Both End Points (NND) [41] as well as a Greedy Algorithm [42]. In the first algorithm, the priority values of the edges are determined, and an initial solution is found. Then the NND algorithm is used from selected vertices, and the priority values of the edges are updated by considering how many times an edge is used in a solution. In the second part of the algorithm, an iteration algorithm is used to improve the existing solution. This process includes the update of the priority values of the edges and their subsequent sorting in descending order. After that, the greedy algorithm is run. The steps of the NND algorithm are as follows:

1. Choose an arbitrary vertex in the graph.
2. Visit the unvisited vertex that is nearest to this vertex.
3. Visit the unvisited vertex that is nearest to these two vertices and update the end vertices.
4. Is there any unvisited vertex left? If yes, go to Step 3.
5. Go to the end vertex from the other end vertex.

The steps of the Greedy algorithm are as follows:

1. Sort all edges in descending order.
2. Select the shortest edge and add it to the tour if it does not violate any of the above constraints.
3. Do we have any edges in the tour? If not, go to Step 2.

Mueller & Kiehne [43] proposed a different hybrid approach. Ant Colony Optimization (ACO) is a population-based metaheuristic that can be used to find approximate solutions to difficult optimization problems. In ACO, a set of software agents called artificial ants, search for good solutions to a given optimization problem. To apply ACO, the optimization problem is transformed into the problem of finding the best path on a weighted graph. The artificial ants (hereafter ants) incrementally build solutions by moving on the graph. The solution construction process is stochastic and is biased by a pheromone model, namely, a set of parameters associated with graph components (either nodes or edges), whose values are modified at runtime by the ants. The general idea of combining ACO and ANN, is to let the ants construct a tour which is then improved by applying a Self-Organizing Map. As the ACO algorithm is faster in converging towards a good (but not a very good) solution, the idea is to use the ANN as a kind of local search. The procedure is as follows:

1. Initialize ACO and SOM with the given parameters.
2. Solve the given TSP with the initialized ACO.
3. Extract the best-found tour in ACO and insert it into the SOM.
4. Solve the SOM.
5. Return the solution when SOM training is finished.

4.2 Evolutionary Models

Majdoubi et al. [44] proposed a Corona Virus Optimization Algorithm and Self-organizing Maps (CVOA+SOM). This work consists of applying a new hybrid method that uses both CVOA and SOM, to solve the Euclidean TSP. The conceived approach is based on considering subsequently a random path of n cities as an input pattern. For each one of the generated tours, the adaptation parameters are calculated to select the winner node with criterion the minimum distance from the located point of the tour. The adaptation procedure is then applied to detect the neighbors, to create a new candidate solution. The implementation of the proposed method leads to an evolutionary algorithm based on probabilistic procedures that requires a definition of the replicate function, which is used to insert new individuals to participate in CVOA evolution. The replicate function is therefore responsible for the generation of a candidate solution to use with the SOM algorithm. The main properties of CVOA are as follows: The defined probabilities and parameters are updated by the scientific community; the exploration of search space is possible, as long as the infected population is not null; the high expansion rate, guarantees the better use of search space, leading to the intensification of the resolution. The concept of parallel strains is reformulated by applying the processing algorithms in different processors, to generate diversification of the resolutions. The ob-

tained average deviation for data sets Pr76 and Rat99 is 0.3 and 0.5 respectively, solved by the present combined algorithm and where the learning parameters are adapted (The learning rate and the neighborhood function).

Tinarut & Komgrit [45] proposed a hybrid SOM. It uses local search which is a solution to a specific problem. The map is definitely fixed, and the solution is an estimated solution, which cannot be guaranteed to be the optimal solution or the same solution every time. A typical local search, after finding an initial feasible solution result, generates a mechanism to change the answer, and then stops the answer when an acceptable value is received. The mechanism of changing the results for this TSP problem can be achieved by changing the position of the path that connects the two present cities to the brand-new route that connects the city to the new point. If these changes lead to a shorter distance with respect to the old solution, the old solution is replaced with the new one. This procedure is performed repeatedly, until the required number of rounds is reached, or until acceptable conditions are met. In this study, the following three routes are used: 2Opt exchange operator, Relocate operator, and Exchange operator. 2Opt Exchange Operator for TSP aims to reduce the distance without crossing routes. Relocate Operator will change two routes connected to each other at a given point, without passing through it again. Then it will create a new route to that point, by changing the one or the other route into two routes connected to that point. The Exchange Operator will change two routes connected to each other at a given point into two routes connected to a new point. The two old routes connected to the new point will be transformed into two routes connected to the old one. The proposed algorithm can be outlined as follows:

- Initialization.
- Result improvement by local search.
  - 2Opt exchange operator.
  - Relocate operator.
  - Exchange operator.
  - Solution examination.
  - Processing cycle increases.

Brocki [46] proposed a 2opt+SOM approach that combines the Kohonen network with the 2opt algorithm to lead to an improved model. More specifically, a neuron and a city are assigned to each other. All neurons are organized in a vector that represents a sequence of cities that must be visited. Unfortunately, the Kohonen SOM model without some modifications is unable to solve TSP. The reason for this, is that if the neural weights are used as the coordinates of the cities, they may not equal to the coordinates of cities that are given. To solve the problem, an algorithm that would modify Kohonen's solution to one that is valid, has been created. In this algorithm the positions of cities and the positions of neurons may not be equal. However, adequate neural weights and cities' coordinates are very close to each other. An algorithm that modifies neural weights so they are equal to cities' coordinates can be applied. The 2opt algorithm selects a part of the tour, reverses it, and inserts it back into the cycle. If the new tour is shorter than the original cycle, then it is replaced. The algorithm stops when no improvement can be done. This algorithm was improved by Ahmad & Kim [79], via a modification to the neighborhood method, using a Gaussian method.

Vieira et al. [47] proposed a SOM Efficiently applied to the TSP (SETSP). The proposed modifications to the SOM, lead to a reduction in the algorithm complexity from execution on n^2 steps to execution on, at least, n steps, where n is the input data set size (number of cities), as the algorithm evolves in time. Thus, even for a large data set, the algorithm executes faster as the number of iterations evolves. Furthermore, the complexity reduction is guided by the neighborhood function variance which decreases. This feature is attractive, especially in the case of processing large data sets. The applied modifications include a change to the network structure (a ring network is proposed) and initialization, a definition of a neighborhood function threshold, as well as a definition of new parameter adaptation laws. The main important aspects of the SETSP are algorithm complexity reduction, according to the number of iterations and its faster convergence.

4.3 Genetic Models

Nourmohammadzadeh & Voss [48] proposed a clustered SOM model that uses the SOM as well as the Genetic Algorithm (GA) and consists of three steps. In the first step, a SOM of a given size is applied to a data set of node coordinates. During the training process, the SOM neurons are moved on the plane and the shape of the network changes. According to the final shape of the map, the cities are clustered based on the SOM neuron that is nearest to each one of them. Therefore, the cities of each cluster are known and the sub-TSPs are constituted. In the next step, the GA is applied to each sub-problem to minimize the length of the sub-tours. Furthermore, another GA is used that tries to find a good order for connecting the sub-tours. The GA codes the order of

visiting the cities and clusters as strings of unique numbers as it is shown in **Fig.4**, where the square shapes are the unique numbers and the rest is the order of the cities. After this step, a solution for the entire original TSP can be found. The number of clusters is equal to 1 for every 20 cities. It is observed that the proposed SOM with the complementary GA is superior in terms of the objective value as well as and the computation time with their competitors. Before each GA application, its parameters are tuned by the Taguchi design of experiments [49] and trial and error method. In Taguchi, three levels (meaning three variables: nPoP which is the a population of random initial solutions, Cross_rate for doing crossovers on the solutions and nMutat which is a proportion of the population is chosen randomly from the population for mutation) are considered as the potential values of each parameter and a number of experiments are conducted with some designed combinations of the levels. For each level the value of a signal-to-noise ratio (SNR) ratio is calculated. The smaller SNR values are better in this minimization problem. However, in the trial-and-error approach used to determine the maximum number of iterations, the experiments are conducted using increasing values for the maximum number of iterations, starting from 50 iterations. **Fig.4** depicts a solution built according to this method for a small TSP, by connecting the cities of each cluster, and then, the clusters themselves.

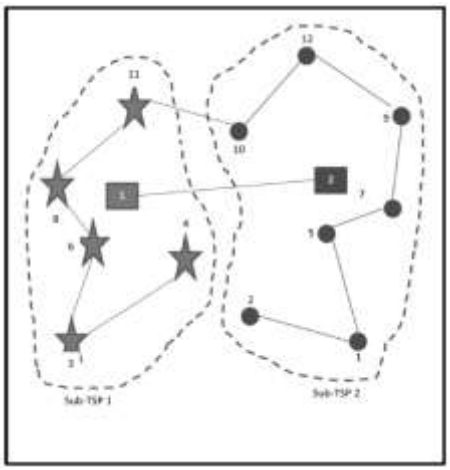

Fig. 4: An example solution [48].

Jin et al. [50] proposed an Integrated Self-Organizing Map (ISOM), whose name is due to the fact that its learning rule integrates the three learning mechanisms found in the SOM literature. More specifically, within a single learning step, the excited neuron is first dragged towards the input city, then pushed to the convex hull of the TSP, and finally drawn towards the middle point of its two neighboring neurons. The evolved ISOM (eISOM) uses a genetic algorithm to handle inherently hard or time-consuming problems. In this algorithm, every individual represents a learning scheme. For each learning scheme, the parameters include the type of formula to calculate the expanding coefficient and the parameters a_i (i =1,2,3,4). They also include other parameters in the ISOM, such as the radius R, the total learning loop L, the initial values, and the decreasing schemes of the effective learning width σ(t), as well as the learning parameters $\eta_1(t)$ and $\eta_2(t)$. To evaluate the solution quality, the relative difference between the generated tour length and the optimal tour length as used. The eISOM is one of the most accurate SOMs for the TSP with quadratic computation complexity.

Vishwanathan & Wunsch [51] proposed an Adaptive Resonance Theory + SOM (ART/SOFM) approach. The adaptive resonance theory in its basic form uses an unsupervised learning technique. The terms “adaptive” and “resonance”, mean that they are receptive to new learning (adaptive) without discarding the previous or the old information (resonance). The ART networks are widely known to solve the stability-plasticity difficult choice with the stability to describe their nature of memorizing the learning and the plasticity to describe the fact that they are versatile to achieve new information. Due to the nature of ART, they are always able to learn new input patterns without forgetting the past. The model uses a combination of Adaptive Resonance Theory (ART) [52] and Self Organizing Feature Maps (SOFM) to solve large-scale TSPs. The use of this combination, allows the number of neurons to be a steady one. This approach is divided into the following modules:

- Clustering.
- Finding Hamiltonian paths for each cluster.
- Finding a Hamiltonian loop for the inter-cluster tour.
- Linking the clusters, and applying heuristics to improve the solution.

Note that clustering techniques have been frequently used to divide the TSP into smaller subproblems and combine the sub-solutions separately.

4.4 Memetic Models

Creput & Koukam [53], proposed a MEMETIC SOM model. This memetic SOM uses an evolutionary loop embedding the SOM. The main operator is the SOM algorithm applied to a graph network. The memetic loop applies a set of operators sequentially to a population of Pop

individuals, at each memetic iteration, called a generation. A loop executes a constant number of generations. The number of individuals is also constant. At each generation, a predefined number of SOM basic iterations are performed allowing the long decreasing run to be interrupted and combined with the application of other operators that can be other SOM operators with their own parameters, mapping, fitness evaluation, and selection operators. A construction loop, as well as an improvement loop, are instantiated based on the memetic loop structure presented in the code of **Fig.5.**

```
Initialize population with Pop individuals.
g = 0.
While g < Gen
  Apply one or more standard SOM processes, with their own parameter
    settings, to each individual in population.
  Apply mapping operator MAPPING to each individual in population.
  Apply fitness evaluation operator FITNESSTSP to each individual in
    population.
  Save the best individual encountered.
  Apply selection operator SELECT.
  Apply elitist selection operator SELECT_ELIT.
  g = g + 1.
End while.
Report best individual encountered.
```

Fig. 5: The generic memetic loop embedding SOM [53].

The details of the operators are as follows:

1. Self-organizing map operator: It is the standard SOM applied to the ring network.
2. Mapping operator: This operator, denoted MAPPING, generates an admissible TSP solution and modifies the shape of the ring accordingly, at each generation. The operator maps cities to their nearest neuron, not already assigned to a city. This generates a valid tour. Then, the operator moves neurons to the location of their single assigned city (if it exists) and dispatches regularly (by translation) other neurons along edges formed by two consecutive cities in the tour.
3. Fitness operator: For each individual, this operator, denoted FITNESSTSP, evaluates a scalar fitness value that has to be maximized and used by the selection operator. It returns the fitness value fitness = – L, where L is the length of the tour defined by the ordering of cities mapped along the ring.
4. Selection operators: Based on fitness maximization, the operator denoted SELECT, replaces the worst individuals associated with the lowest fitness values in the population, with the same number of the best individuals, associated with the highest fitness values in the population. An elitist version SELECT_ELIT, replaces the worst individuals with the single best individual encountered during the run.

Avsar & Aliabadi [54], proposed a Parallelized Memetic Self-Organizing Map (PMSOM). In this approach the cities are organized into municipalities, the most appropriate solution from each municipality is used to find the best overall solution and finally the neighboring municipalities are joined by a blending operator to identify the final solution. The proposed system is a more generalized form of the previous memetic SOM. It uses different techniques in the following order:

1. Divine map into municipalities by K-Means clustering algorithm.
2. It uses an algorithm that shows the learning mechanism of each municipality. If the municipality has not yet converged, it will start to randomly choose a subset of assigned cities and apply KNIES learning rule over each member of the population.
3. Merge the neighborhood municipalities.
4. Remove kinks (kinks make tours more complicated and long) of mixed clusters with an algorithm similar to the famous 2opt algorithm.
5. Update Adjacency Matrix.
6. Termination.

4.5 Chaotic Models

Matsushita & Nishio [55] proposed an interesting Chaotic SOM (CHAOSOM) model, associated with the Hodgkin-Huxley equation [56], a mathematical model used to simulate action potentials in giant squid axons. The basic algorithm of CHAOSOM is the same as SOM; however, the important feature of CHAOSOM is the ability to refresh the learning rate as well as the neighboring coefficient at the timing of the spikes generated chaotically by the Hodgkin-Huxley equation. The neighboring coefficient is a set of neighboring neurons of the winner neuron which includes a parameter of the number of spikes and an increasing value depending on a chaotic signal spike.

Ryter et al. [57] proposed an application of different Chaotic Maps on the SOM in order to solve the TSP. As it is well known from mathematics, a chaotic map is a map (and more specifically, an evolution function) that exhibits some sort of chaotic behavior. These maps may be parameterized using a discrete-time or a continuous-time parameter. Discrete maps usually take the form of iterated functions. Chaotic maps often occur in the study of dynamical systems and are able to generate fractal structures. Although a fractal may be constructed by an iterative procedure, some fractals are studied on their own, rather than in terms of the

map that generates them. This is a usual approach, because there are several different iterative procedures that generate the same fractal. The maps that are used and the associated results are given in Table 1 [57]. The results differ from each other, depending on the scale of the problem (big or small scale) and the iterations. Whether a chaotic map performed just better denoted with (+) or worse denoted with (-) and both are statistically insignificantly better/worse. If a map is denoted with (++) or (- -) have better/worse significance within a 95% confidence interval. The Henon map corresponds to a time-discrete dynamical system [58]. The Burgers Map is a discretization of a pair of coupled differential equations that were used by Burgers to illustrate the relevance of the concept of bifurcation to the study of hydrodynamics flows [59], [60]. The results shown in Table 1 [57], are scaled from 1000 to 1000000 iterations (1K to 1M) and they are divided into small and big-scale problems.

Table 1. Computational Experiments of Chaotic Maps

	Small Problem			Big Problem		
Iterations	1K	10K	1M	1K	10K	1M
Arnold Cat Map	-	+	-	+	-	-
Burgers Map	+	+	-	++	++	- -
Delayed Logistics	-	-	- -	+	++	- -
Dissipative Map	-	-	-	-	+	+
Henon Map	+	++	- -	-	+	-
Ikeda Map	- -	+	-	+	+	-
Lozi Map	-	-	- -	-	+	-
Sinai Map	-	+	+	-	+	-
Tinkerbell Map	-	-	- -	++	++	- -

The experiments showed that significantly better results are obtained for large-size problems with a small or medium number of iterations when the pseudorandom number generator is replaced by Burgers Map, Delayed Logistics Map [61], or Tinkerbell Map [62]. For smaller problems, the Henon Map showed significantly better results for a medium number of iterations.

4.6 Fuzzy Model

Chaudhuri et al. [63] proposed a Fuzzy SOM approach. The Fuzzy Self Organizing Map [64] introduces the concept of membership function in the theory of Fuzzy Sets to the learning process. Some network parameters related to the neighborhood in the Self Organizing Map, are replaced by the membership function. Also, the learning rate parameter is omitted. The Fuzzy Self Organizing Map is more effective at reducing oscillations and avoiding dead units, because they take into account all input data at each iteration step. The Fuzzy Self Organizing Map used here, is a combination of the Self Organizing Feature Map and the Fuzzy C Means clustering algorithm. The more complex the problem is, the more output neurons are required. The number of output neurons is manually selected. The weight components are initialized randomly and adjusted gradually using a Self-Organizing learning algorithm, and ultimately a mapping is done from input to output. The learning algorithm consists of the following steps:

1. Randomize the weights for all the neurons.
2. Input all the patterns.
3. Take one random input pattern and calculate the Euclidean distances.
4. Compute the memberships of each pattern to all neurons.
5. Find the winning neuron and the neighbors of the winner.
6. Adjust the synaptic weights of each neuron according to the computed memberships.
7. Reduce the values of the parameters.
8. Determine the stability condition of the network.

4.7 Simulated Annealing Model

Takano et al. [65] proposed modified self-organizing map and Simulated Annealing (SOM + SA) approach. Regarding the SOM network, the fact that a map is formed, with data having similar features to be arranged nearby, while other data are arranged at a distant place, the route will be identified according to the order of cities which is determined at random. Therefore, the obtained route, quickly suggests a route pattern, close to the global optimal. However, because the accuracy deteriorates locally, SA (Simulated Annealing), which is excellent for local searching, is then applied to compensate for this area. With this procedure, SA is performed using the solution obtained by SOM, as the initial solution of SA to obtain a higher accuracy route. The algorithm of the SOM+SA procedure is the following:

1. Execute the algorithm of SOM.
2. Perform crossing judgment and crossing removal, for the solution in the previous step.
3. Execute the algorithm of SA using the solution obtained in Step 2 as its initial solution.
4. Perform crossing judgment and crossing removal in order for the solution obtained in Step 3, to be used as the final solution.

In the proposed SA+SOM solver, the determination of an initial temperature for simulated annealing, influences the search performance. To obtain higher solution accuracy, the appropriate parameters of the SA process have to be selected. The estimated computational complexity as well as the accuracy of the solution for large-scale TSP were estimated, revealing the possibility that an approximate solution for a problem of tens of thousands of cities, is obtainable using an ordinary PC.

4.8 Elastic Net

Chen et al. [66] proposed an Elastic Net with SOM (EN+SOM) model. This paper presents an Elastic Net (EN) algorithm, by integrating the ideas associated with SOM. The proposed solution is based on a SOM structure, initialized with as many artificial neurons as the number of targets to be reached. In the competitive relaxation process, the information regarding the trajectory connecting the neurons, is combined with the distance of neurons from the target. The gradient ascent algorithm attempts to fill up the valley, by modifying parameters in a gradient ascent direction of the energy function. The EN is based essentially on how the constraints are represented in its energy function, and thus can be used in the solution of non-geometrical problems. In each iteration of the EN algorithm, the coordinates of all nodes are calculated even if there is no change in the coordinates. However, in SOM, one or a few weights are updated. In this paper, SOM is used to construct the network, and quantify the criteria. EN is used to derive the TSP route under a set of weights and display the results.

5 Multiple Salesman Models

Multiple salesman problem is an extension of the well-known Traveling Salesman Problem (TSP), where a number of m salesmen are used to serve a set of n locations / cities, restricted to the original constraint that each location must be visited only once. The agents / salesmen share a single-depot case known as Single-Depot Multiple-TSP or use multiple depots, a variant known as Multiple-Depot Multiple-TSP.

Lupoaie et al. [67] proposed MinMax Multiple-TSP. In this model the evolutionary algorithm as well as the ACO approach are used. Furthermore, the algorithms are tested individually but also, they combined with each other. In the case of SOM for Multiple-TSP, the topology of the output layer must be modified such that instead of a single route that has the standard SOM, a number of m routes must be generated. **Fig.6** illustrates the initialization step, an intermediate step, as well as the final state of the training process; the solution is represented in the fourth graph. The evolutionary algorithm for solving Multiple-TSP was the two-part chromosome representation. Because of the requirement to perform more complex mutations within an individual, a requirement imposed by the necessity of balancing the tours, the multi-chromosome technique [68] is used. This technique reduces the size of the search space, by eliminating redundant solutions and it can be improved using the 2opt local search heuristic approach. The ACO approach uses g-MinMaxACS which is a variation of the Ant Colony System. More specifically, instead of using a single ant to construct one tour as in TSP, a set of m ants is used instead, to generate a complete solution to the Multiple-TSP problem. Initially, all ants start their tours from the depot, and the salesman to visit the next city is selected at random. A more elaborate description can be found in [69]. In the hybridization stage, the emerged models are created via the combination of SOM and g-MinMaxACS (SOM-ACO), SOM and EA (SOM-EA), and SOM, EA, and 2opt (SOM-EA-2opt). In the MinMax Multiple-TSP, the number of salesmen that were used is 2, 3, 5, and 7. Ant Colony System achieves better results than both SOM and EA. When comparing the performance of the hybrid version SOM-ACO with the standalone ACO is slightly the same in some cases. In the case of the evolutionary algorithm, it is evident that the hybridization between EA and SOM improves greatly over the simple EA in all cases. Regarding the comparative performance of SOM-ACO versus SOM-EA, the latter achieves statistically significant better results in most cases. Generally, hybridization not only improves results but also reduces the variance. When enhancing SOM-EA with the 2opt local search heuristic, the results also improve significantly.

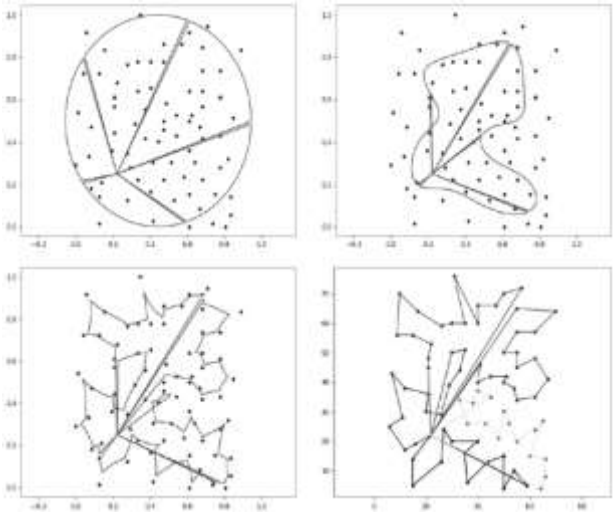

Fig. 6: SOM for Multiple-TSP for the data set eil76 with 5 salesmen: initialization, intermediate iteration, final iteration, and final solution. The points correspond to cities, while the curves correspond to the trajectory given by traversing the neurons using the ring topology [67].

Zhu & Yang [70] proposed a Multiple Traveling Salesman Problem (MTSP). In this model there are k rings (where k is the number of salesmen) and M nodes that represent the cities. This model has 2 nodes in the input layers and kM nodes in the output layer (or equivalently k rings with each ring containing M nodes). The rule for identifying the winning neuron is modified with a new term that increases the possibility of a node to be a winner when it is far away from the home city. The algorithm can be easily implemented for parallel computation. It is compared to conventional SOM from Modares et al. [15]. The comparison of MTSP with the conventional TSP, was performed using 2, 3, and 4 salesmen for each one of them. The quality of the results associated with the improved SOM approach, is better than the quality associated with the conventional SOM approach. In addition, the improved SOM approach converges faster than the conventional SOM approach. The differences between the two approached are: the use of M nodes in the output layer instead of 2M nodes, to reduce the calculation workload, the modification of the neighborhood function by adding an equation to speed up the convergence, and lastly, the modification of the update rules, by adding an equation to increase the convergence.

A summary of all models presented in the previous sections, can be found in Table 2 (enriched models) and Table 3 (hybrid models). In both tables, the first column presents the type of the model. The second and the third columns include the name and the release date of the model, while the other columns present for each model, the number of the cities that were tested, the number of iterations performed, and many other useful information and comments. Most of the models are capable of giving better results for large-scale problems and others for small datasets. Last but not least, the tables show the number of neurons that are used in the first phase of the algorithm which is usually the initialization phase. Usually, the number of neurons that are used is equal to the number of the cities of the TSP, but sometimes it is either 1.5 or 2 times more. Selecting the number of nodes that is equal to the number of cities makes the process of node separation more delicate and results in the sensitivity of the algorithm to the initial configuration. All of the models have a common 2 layers in the input. On the input layer, they take as entry the Cartesian coordinates of the 2D point (city). On the other Tables 4 and 5 present the models that we discussed before and the models or algorithms that competed against during their experiments. With the **X** sign, we can perceive their competitors while Table 6 presents a summary of the Multiple TSP models. Table 7 shows the competition of the Multiple TSP models as they were mentioned in the previous tables. Table 8 is an assemblage of all the models.

6 Vehicles & Drones

The vehicle routing problem (VRP) is a well-known combinatorial optimization problem concerned with the identification of the optimal routing of goods between a central depot and a set of customers. The Capacitated Vehicle Routing Problem (CVRP) [71] introduces a restriction on a vehicle's capacities. To extend the SOM application from the TSP to the CVRP, there are two factors that must be considered, and more specifically, the number of vehicles in the problem as well as the vehicle capacity constraint. Applying the SOM to the CVRP, there are K rings of neurons (one ring for each of the K vehicles / routes), where each ring consists of M neurons. The output neurons must now be indexed using the ring (route) as well as the position within the route. To account for the capacity constraint of the CVRP, there are two approaches that have been followed, according to the literature, both of which incorporate a mechanism to penalize routes that are over capacity when choosing the winning node for an input city X_i. One of the earliest applications of the SOM to the CVRP uses a probabilistic approach for choosing the winning node J [72]. In this algorithm, each time an input city, X_i, is presented to the network, the closest node (measured by Euclidean

distance) on each route is identified. From these K nodes, the winning node is chosen in such a way that the probability of the winning node being chosen from an overloaded route, tends towards zero as the network evolves in time. The second approach incorporates a bias term to penalize overloaded routes during the competition phase of the algorithm. The earliest variation of this approach multiplies the Euclidean distance by a 'handicap' factor as the nodes compete [73]. In this method, a larger 'handicap' indicates that the current route is near or over the capacity constraint. In subsequent SOM approaches to the VRP, a bias term is added to the Euclidean distance as the nodes compete. In this approach, a winning node has the smallest combination of Euclidean distance and bias term. The published results of the application of the SOM to the CVRP that utilize benchmark problem sets, surpass the results from previous SOM approaches [72]–[75].

Tairi et al. [76] proposed a traveling salesman problem with drones solution with SOM (TSP-D). The goal of the traveling salesman problem with drones (TSP-D) is to efficiently route a vehicle equipped with a drone from a source depot to deliver packages to a set of customers and return to the source depot with minimal distance traveled. The TSP-D has been widely studied since first being introduced in 2015 and it has been shown to significantly reduce travel time in comparison to a pure TSP. The model proposed in this study uses a self-organizing map to determine an initial solution to the underlying TSP. In the next step, a greedy algorithm is used to determine ideal customer nodes to be serviced by the drone, to reduce the total distance traveled by the vehicle. The data used for this study were retrieved from a National TSP library. After determining a feasible route for the vehicle by setting the SOM network, the next phase in the solution is associated with the determination of the ideal candidates for drone insertion. Drone insertion refers to the process of dropping one of the nodes from the truck route and instead, adding it to the drone route. The objective of this algorithm is the determination of nodes (if such nodes selected) for drone delivery that reduce as much as possible the total distance traveled by the truck. This process of inserting drone nodes is iterated, until there are no more potential candidates for drone delivery to minimize the distance traveled by truck. The network size used for this study was eight times the population size and the number of iterations was initialized to 100,000, even though the algorithm stopped after 24,487 iterations. The algorithm tested on 194 customer nodes and the results showed that the initial solution generated using the SOM was 7% higher than the optimal solution, while the solution to the TSP-D was 39% less than the optimal TSP solution and was computed in 28.8 seconds. The final solution to the large TSP-D, suggests that pairing a drone with a delivery truck, can help significantly reduce the transportation distance as well as time, if the procedure is planned efficiently.

7 A Comparison of the Presented Algorithms

In this section, comparisons between the models are performed. The presented results are associated with the similarities as well as the difference between the presented models. These models are also compared and some experimental results are given.

Compared with traditional SOM-based TSP methods, TOPSOM relaxes the stronger 'closed' constraint at initialization stage, and is able to satisfy the convex-hull constraint by using an elastic competitive Hebbian learning mechanism during optimization. TOPSOM has an average PDM (i.e. the percent deviation of the mean solution value to best Known Solution) of 4.16% and 8.02% in small-scale ch130 and medium-scale uy734, while the second best one (i.e. eISOM) only gets an average PDM of 6.37% and 9.87%, respectively. As a result, TOPSOM outperforms eISOM by up to 2.21% and 1.85%. For the large-scale instances like ro2950 and fi10639, TOPSOM has an average PDM of 14.53% and 21.94%, respectively, while the second-best eISOM gets an average PDM of 15.67% and 23.45%, respectively. Thus, TOPSOM outperforms eISOM by up to 1.14% and 1.51%. The superiority of TOPSOM becomes prominent from the perspective of the speed for problem-solving, by 14.08% faster than traditional SOM on average.

The Memetic SOM is superior to the co-adaptive net as well as the ORC-SOM in generating the best tours on average, indicating that Memetic SOM is not only fast but also superior in getting more precise solutions. However, this is true only for small-scale instances. On the other hand, for large-scale instances, ORC-SOM has 6.178% on average, which is the smallest compared with those from the co-adaptive net and memetic SOM, making 1.76% and 1.34% improvement respectively. This indicates that ORC-SOM is suitable for more precise solutions with higher computation complexity for large-scale problems, compared with the memetic SOM and co-adaptive net.

The CHSOM obtained better results than Leung's algorithm for problems with 70, 107, 124, 195, and

442 cities. Although CHSOM did not find better solutions for KROA200 and ATT532, the solution quality is very near to that of Leung's algorithm.

The Expanding Property SOM competed against Leung's ESOM and the performances of both shows that it is very close to the theoretical bounds. The main reason for which the error of the Expanding Property SOM is slightly larger than Leung's ESOM, is that the proposed learning rule is comparatively more conservative than Leung's one. Thus, the proposed learning rule limits the range of convex-hull expansion of the excited neuron. More specifically, it has a faster average execution time, but the solution quality is close.

The solution qualities of the ESOM range from 4.05% to 11.35%. Therefore, its performance, even though it is a bit worse as the number of cities increases, is still stable. The ESOM takes slightly longer in execution time than Budinich's SOM.

The fact that MGSOM and EGSOM have similar specifications, was the reason for which some of the TSPs that were tested, had the same outcome. However, the MGOM is characterized by a better average deviation from the optimal tour length, with a total average of 2.3215%, compared with EGOM with 2.4925%, and MSTSP with 2.4372%.

In the case of the Enhanced SOM Solution, a set of additional modifications performed after choosing the baseline hyperparameter configuration, led to slightly better results with a 0.07891 F1 score.

The CAN competed against MEMETIC SOM, ORC-SOM, eISOM, ESOM, and SETSP and gave the worst average results among all the others. The Modified CAN competed against the original one and the results have shown that the processing time was reduced in half and in the PDM (the percentage deviation to the optimum tour length of the mean solution value) has better results only where the number of cities is between 200 and 300. The original one (CAN) has better results for less than 200 cities but when it is more than 300 both of them have similar results.

The experimental results showed that on small-scale TSP, only the ESOM algorithm is more prominent, and the deviation rates of KNIES and ORC-SOM are not much different. But once the scale of the experimental data is relatively large, the ORC-SOM algorithm has the lowest deviation from the optimal rate. This shows that for medium and large-scale TSP problems, the use of ORC-SOM algorithm makes the identification of the globally optimal path easier. When the TSP data size is greater than or equal to 500, compared with the other two algorithms, ORC-SOM has the smallest optimal deviation rate. When the TSP data size is less than or equal to 1000, the running times of various algorithms are very close. Finally, when the TSP data size is greater than 1000, the ORC-SOM algorithm has the smallest running time, showing its own advantages. After the above comparison, it is not difficult to see that for small-scale TSP data, ORC-SOM has no outstanding features compared to other algorithms, while for larger-scale data, ORC-SOM can not only find a good path, but also has a relatively small running time.

Furthermore, KNIES is more complicated than ESOM. Consequently, the performance of ESOM is better than KNIES. A comparison of these algorithms reveals that KNIES-TSP executes at least on $n \times n$ steps, while SETSP executes at least on n steps so SETSP has shown superior performance compared to KNIES. The success of KNIES DECOMPOSE increases as the problem size increases. If att532, pcb442, kroA200, and rat195 data sets are considered, the average relative deviations from the optimal tour lengths, become 7.03, 8.94, and 8.93 for KNIES DECOMPOSE, KNIES_TSP, and KNIES_TSP_Global, respectively. Besides accuracy, KNIES DECOMPOSE cuts the running time necessary for KNIES_TSP_Global to solve att532, from 49,888.68 seconds down to 3.02 seconds, with no deterioration in the solution quality.

Bernard's Angeniol approach was compared with the elastic net method presented by Durbin and Willshaw. The results for five sets of 50 cities show similar characteristics. On average, both approaches are equivalent. The results compared to Tank and Hopfield demonstrates that a low value of the parameter 'a' gives a better average, but a high value has the advantage that the optimum is sometimes reached. It also gives a good solution in several tries in less time than one try with a low value of a. It seems the 2opt + SOM hybrid is not a very powerful algorithm for the TSP. It has been outperformed by EA as well as Lin-Kerninghan algorithms.

The approach of Mueller & Kiehne outperforms ACO and SOM on a level of 5%. In the hybrid algorithm, this method competed against Zhou and it was faster about 84% considering the same TSPs, and 11% faster than Peker's algorithm. In comparison to the approach which uses the K-means and the Firefly algorithm, it is observed that Clustered SOM with the complementary GA is

superior in terms of both the objective value and the computation time.

The ART/SOFM model has a 10% excess over the optimal tour length on TK2392 and the eISOM on the same TSP has a 6.44% excess. Also, in the same amount of cities, on PR2392 TSP Memetic SOM has a 7.34%, The ESOM has 8.49%, the eISOM has 6.44%, and the ORC-SOM is 7.16%. The Eli51 is one of the common TSPs which is used many times. The SETSP has a computational result of 435, close to the optimum length.

Furthermore, Fuzzy has the same computational result as SETSP, and on the MGSOM it has 431 so as EGSOM. The Hybrid has 440. The Iterative has 470. On the excess %, the results of the Eli51 on the eISOM is 2.56%, on SETSP is 2.22%, on the MEMETIC it is 2.52%, on the EGSOM it is 1.39% as the MGSOM too and the ORC-SOM 3.00%. The results above are associated with small-scale TSP problems.

Now, let us present results associated with a large scale of problems, meaning more cities. The TSP called fi10639 has 10639 total cities. TOPSOM as it was presented previously, has a percentage Deviation of the Mean solution equal to 21.94%, compared with eISOM which has 23.45% and the traditional SOM with 27.02%. The MEMETIC has 6.93%. The TSP berlin52 has 52 cities so it is a small-scale problem. The optimal tour length of the traditional SOM is 7544. The result of the Self-Organizing Iterative's length on that TSP is 7959. The eISOM has a mean length of 8148 and the TOPSOM has a value of 7995. Also, in the hybrid model, the average value is 8208. Furthermore, in the Mueller & Kiehne approach, the mean value is 7669.

One common characteristic of most of the models, is the Euclidean distance, that is used to find the nearest neighbor. Only the approach of Mueller & Kiehne took the TSP from the Welch Two Sample t-test. Other than that all of them used TSPLIB instances. Furthermore, TOPSOM used National TSP's instances too.

CHAOSOM compared with SOM, and the percentage error of the optimal distance on the CHAOSOM has shown better results.

Fuzzy compared to the Evolutionary Algorithm and the Lin Kernighan Algorithm. This algorithm provides significantly better results for TSP as the number of cities increases.

The average improvement rate of the solution accuracy (average error for optimal) in the SOM+SA process is estimated at about 2%, but it is scattered depending on the size of the problems. The average CPU time on SOM+SA is $O(n^{2.3})$ for the number of cities equal to n.

The PMSOM was compared with the MEMETIC SOM and the results show that the computational speed is faster, because it uses clusters, while the number of cities increases and the PDM% is 4.53 compared to 5.87 of the MEMETIC SOM.

To sum up, the objective of this work was the study of a set of papers, presenting an interesting SOM and the evaluation and comparison regarding their use to the TSP. The mathematical equations of the algorithms can be found in the papers describing them. ORC-SOM had the best average results from the optimal tour length, considering all the other enriched SOM models for small-scale problems. Regarding the large datasets, TOPSOM produces the fastest results with a good performance. On the Hybrid models, for small datasets, the eISOM model produces the best results, while Fuzzy seems to produce the fastest time but not as decent results as eISOM. On the large scale of problems such as a 10000 of a TSP on average, the PMSOM has the best score and fast execution time. The MinMax Multiple-TSP produces the best results in the category of multiple salesmen.

8 Conclusion

In this survey paper, we gathered and studied models of the well-known Travelling Salesman Problem (TSP) using Self-Organizing maps (SOM). More specifically, a set of models that implement the SOM on TSP instances, are presented. These models were classified according to the algorithms they use. The comparison shows that some models tend to perform better on large-scale problems while some others on small-scale. The performance is good, not only regarding the time spent to finish, but also with respect to the optimality of the tour length. These models maximize their efficiency to provide us with general knowledge of different kinds of algorithms, so we can choose which is better for our problem. In future works, we will keep track of new methods that solve the TSP using SOM in order to create a newer version of this paper with models that have different approaches and superior computational results if possible.

References:

[1] Dantzig, G., et al. "Solution of a Large-Scale Traveling-Salesman Problem." *Journal of the Operations Research Society of America*, vol. 2, no. 4, 1954, pp. 393–410. JSTOR, http://www.jstor.org/stable/166695. Accessed 9 Nov. 2022.

[2] Papadimitriou, C.H., "The Euclidean Traveling Salesman Problem is NP-complete", *Theoretical Computer Science*, 4, 1978, pp. 237-244. https://doi.org/10.1016/0304-3975(77)90012-3.

[3] Kohonen, T., 1990. The self-organizing map. https://doi.org/ 10.1109/5.58325. 1464 – 1480. Proceedings of the IEEE (Volume: 78, Issue: 9).

[4] Hopfield, J., and Tank, D.W. 1985. Neural computation of decisions in optimization problems. *Biological Cybernetics*. 52(3): 141-152. https://doi.org/10.1007/BF00339943.

[5] Qingshu Guan, Xiaopeng Hong, Wei Ke, Liangfei Zhang, Guanghui Sun, Yihong Gong. 2021. Kohonen Self-Organizing Map based Route Planning: A Revisit. 2021 IEEE/RSJ *International Conference on Intelligent Robots and Systems (IROS)*. doi: 10.1109/IROS51168.2021.9636025. 7969-7976.

[6] Junying Zhang, Xuerong Feng, Bin Zhou, Dechang Ren. An overall-regional competitive self-organizing map neural network for the Euclidean traveling salesman problem. https://doi.org/10.1016/j.neucom.2011.11.024. *Neurocomput*. 89 (July 2012), 1-11.

[7] Xinshun Xu, Zhiping Jia, Jun Ma, Jiahai Wang. A Self-organizing Map Algorithm for the Traveling Salesman Problem. *Fourth International Conference on Natural Computation*. 2008. doi: 10.1109/ICNC.2008.569. 431-435.

[8] Haiqing Yang and Haihong Yang. An Self-organizing Neural Network with Convex-hull Expanding Property for TSP. 2005 *International Conference on Neural Networks and Brain*. doi: 10.1109/ICNNB.2005.1614637.

[9] Kwong-Sak Leung, Hui-Dong Jin, Zong-Ben Xu. 2004. An expanding self-organizing neural network for the traveling salesman problem. https://doi.org/10.1016/j.neucom.2004.02.006. *Neurocomputing*, Vol. 62, December 2004, pp. 267-292, ISSN 0925-2312.

[10] Yanping Bai, Wendong Zhang, Zhen Jin. 2006. An new self-organizing maps strategy for solving the traveling salesman problem. *Chaos, Solitons & Fractals*, 28(4), 1082-1089. https://doi.org/10.1016/j.chaos.2005.08.114.

[11] Yanping Bai, Wendong Zhang, Hongping Hu. 2006. An Efficient Growing Ring SOM and Its Application to TSP. *Proceedings of the 9th WSEAS International Conference on Applied Mathematics*, Istanbul, Turkey, May 27-29, (pp351-355).

[12] Wen Dong Zhang, Yan Ping Bai, and Hong Ping Hu. The incorporation of an efficient initialization method and parameter adaptation using self-organizing maps to solve the TSP. *Applied Mathematics and Computation* 172 (1 January 2006) 603–623. https://doi.org/10.1016/j.amc.2005.02.025.

[13] Joao P. A. Dantas, Andre N. Costa, Marcos R. O. A. Maximo, Takashi Yoneyama. 2021. Enhanced Self-Organizing Map Solution for the Traveling Salesman Problem. *Neural and Evolutionary Computing* (cs.NE). https://doi.org/10.48550/arXiv.2201.07208.

[14] 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. https://doi.org/10.1145/2808194.2809488.

[15] Somhom, S., Modares, A., & Enkawa, T. (1997). A self-organising model for the travelling salesman problem. *Journal of the Operational Research Society*, 48, 919–928 https://doi.org/10.1057/palgrave.jors.2600439.

[16] Matsuyama, Y. (1990). Competitive self-organization and combinatorial optimization: Applications to traveling salesman problem. *In IJCNN. International Joint Conference on Neural Networks* (pp. 819-824). Publ by IEEE. doi: 10.1109/IJCNN.1990.137937.

[17] Hannes Schabauer, Erich Schikuta, Thomas Weishaupl. 2005. Solving Very Large Traveling Salesman Problems by SOM Parallelization. doi: 10.1109/PDCAT.2005.223. *Sixth International Conference on Parallel and Distributed Computing Applications and Technologies* (PDCAT'05).

[18] Roman Sushkov. 2015. Self-Organizing Structures for the Travelling Salesman Problem in a Polygonal Domain. CZECH

TECHNICAL UNIVERSITY IN PRAGUE. Faculty of Electrical Engineering.

[19] EM Cochrane and JE Beasley. The co-adaptive neural network approach to the euclidean travelling salesman problem. *Neural Networks* Volume 16, Issue 10, December 2003, Pages 1499-1525. https://doi.org/10.1016/S0893-6080(03)00056-X.

[20] Burke, L. I., & Damany, P. (1992). The guilty net for the traveling salesman problem. *Computers and Operations Research*, 19, 255–265. https://doi.org/10.1016/0305-0548(92)90047-9.

[21] Durbin, R., & Willshaw, D. (1987). An analogue approach to the travelling salesman problem using an elastic net method. Nature 326, 689-691. https://doi.org/10.1038/326689a0.

[22] Jan Faigl. On the performance of self-organizing maps for the non-Euclidean Traveling Salesman Problem in the polygonal domain. *Information Sciences* Volume 181, Issue 19, 1 October 2011, Pages 4214-4229. https://doi.org/10.1016/j.ins.2011.05.019.

[23] Kazushi Murakoshi, Yuichi Sato, Reducing topological defects in self-organizing maps using multiple scale neighborhood functions, *Biosystems* 90 (1) (2007) 101–104. https://doi.org/10.1016/j.biosystems.2006.07.004.

[24] N. Aras, B.J. Oommen, I.K. Altınel. The Kohonen network incorporating explicit statistics and its application to the travelling salesman problem. *Neural Networks* Volume 12, Issue 9, November 1999, 1273-1284. https://doi.org/10.1016/S0893-6080(99)00063-5.

[25] N. Aras, I. K. Altinel and J. Oommen, "A Kohonen-like decomposition method for the Euclidean traveling salesman problem-KNIES/spl I.bar/DECOMPOSE," in *IEEE Transactions on Neural Networks*, vol. 14, no. 4, pp. 869-890, July 2003, doi: 10.1109/TNN.2003.811562.

[26] Jan Faigl. 2018. GSOA: Growing Self-Organizing Array - Unsupervised learning for the Close-Enough Traveling Salesman Problem and other routing problems. *Neurocomput*. 312, C (Oct 2018), 120–134. https://doi.org/10.1016/j.neucom.2018.05.079.

[27] BERNARD ANGENIOL, GAEL DE LA CROIX VAUBOIS AND JEAN-YVES LE TEXIER. Self-Organizing Feature Maps and the Travelling Salesman Problem. *Neural Networks* Volume 1, Issue 4, 1988, Pages 289-293. https://doi.org/10.1016/0893-6080(88)90002-0.

[28] K. Kitaori, H. Murakoshi and N. Funakubo, "A new approach to solve the traveling salesman problem by using the improved Kohonen's self-organizing feature map," *Proceedings of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics*, 1995, pp. 1384-1388 vol.2, doi: 10.1109/IECON.1995.484152.

[29] M. Budinich, "A Self-Organizing Neural Network for the Traveling Salesman Problem That Is Competitive with Simulated Annealing," in *Neural Computation*, vol. 8, no. 2, pp. 416-424, 15 Feb. 1996, doi: 10.1162/neco.1996.8.2.416.

[30] S. . -J. Kim, J. . -H. Kim and H. . -M. Choi, "An efficient algorithm for traveling salesman problems based on self-organizing feature maps," [Proceedings 1993] *Second IEEE International Conference on Fuzzy Systems*, 1993, pp. 1085-1090 vol.2, doi: 10.1109/FUZZY.1993.327363.

[31] Tairi, Mohamed. A Cost Minimization Approach to the Traveling Salesman Problem with Drones Using a Self Organizing Map. State University of New York at Binghamton ProQuest Dissertations Publishing, 2020. 27993460. https://www.proquest.com/openview/8b5256814e3fefbaaf2e6b85f90046ca/1?pq-origsite=gscholar&cbl=18750&diss=y.

[32] Danijel MARKOVIĆ, Miloš MADIĆ, Vojislav TOMIĆ, Sonja STOJKOVIĆ. 2012. SOLVING TRAVELLING SALESMAN PROBLEM BY USE OF KOHONEN SELF- ORGANIZING MAPS. ACTA TECHNICA CORVINIENSIS – *Bulletin of Engineering*, Tome V(2012). ISSN 2067-3809. 21-24.

[33] Victor Sousa Lobo. 2005. One dimensional Self-Organizing Maps to optimize marine patrol activities. *Europe Oceans* 2005. doi: 10.1109/OCEANSE.2005.1511777.

[34] Faigl, Jan et al. "An application of the self-organizing map in the non-Euclidean Traveling Salesman Problem."*Neurocomputing* 74 (2011): 671-679. https://doi.org/10.1016/j.neucom.2010.08.026.

[35] Faigl, J., Přeučil, L. (2011). Self-Organizing Map for the Multi-Goal Path Planning with Polygonal Goals. In: Honkela, T., Duch, W.,

Girolami, M., Kaski, S. (eds) Artificial Neural Networks and Machine Learning – ICANN 2011. ICANN 2011. *Lecture Notes in Computer Science*, vol 6791. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-21735-7_11.

[36] Faigl, Jan et al. "Self-organizing map-based solution for the Orienteering problem with neighborhoods." 2016 IEEE International Conference on Systems, *Man, and Cybernetics* (SMC) (2016): 001315-001321. doi:10.1109/SMC.2016.784442.

[37] J. Faigl, "On self-organizing maps for orienteering problems," 2017 *International Joint Conference on Neural Networks* (IJCNN), 2017, pp. 2611-2620, doi: 10.1109/IJCNN.2017.7966175.

[38] Faigl, J., Váňa, P. (2016). Self-Organizing Map for the Curvature-Constrained Traveling Salesman Problem. In: Villa, A., Masulli, P., Pons Rivero, A. (eds) Artificial Neural Networks and Machine Learning – ICANN 2016. ICANN 2016. *Lecture Notes in Computer Science*(), vol 9887. Springer, Cham. https://doi.org/10.1007/978-3-319-44781-0_59.

[39] Hongjian Wang, Naiyu Zhang, and Jean-Charles Crput. 2017. A massively parallel neural network approach to large-scale Euclidean traveling salesman problems. Neurocomput. 240, C (May 2017), 137–151. https://doi.org/10.1016/j.neucom.2017.02.041.

[40] Urfat Nuriyev, Onur Ugurlu, Fidan Nuriyeva. 2018. Self-Organizing Iterative Algorithm for Travelling Salesman Problem. https://doi.org/10.1016/j.ifacol.2018.11.299. IFAC PapersOnLine 51-30 (2018) 268–270.

[41] Kızılateş G., Nuriyeva F. (2013). On the Nearest Neighbour Algorithms for Travelling Salesman Problem. Advances in Computational Science. Engineering and Information Technology, Springer (AISC,volume 225), pp. 111-118. https://doi.org/10.1007/978-3-319-00951-3_11.

[42] Appligate, D.L., Bixby, R.E., Chavatal, V., Cook, W.J., (2006). The Travelling Salesman Problem, A Computational Study, Princeton University Press, Princeton and Oxford, 593p. ISBN 978-0-691-12993-8.

[43] Dr. Carsten Mueller, Niklas Kiehne. 2015. Hybrid Approach for TSP Based on Neural Networks and Ant Colony Optimization. 2015 *IEEE Symposium Series on Computational Intelligence*. 1431-1435. doi: 10.1109/SSCI.2015.203257094.

[44] EL Majdoubi, O., Abdoun, F., Abdoun, O. (2021). A New Optimized Approach to Resolve a Combinatorial Problem: CoronaVirus Optimization Algorithm and Self-organizing Maps. 947–957. In: Motahhir, S., Bossoufi, B. (eds) Digital Technologies and Applications. ICDTA 2021. Lecture Notes in Networks and Systems, vol 211. Springer, Cham. https://doi.org/10.1007/978-3-030-73882-2_86.

[45] Pongpinyo Tinarut, Komgrit Leksakul. 2019. Hybrid Self-Organizing Map Approach for Traveling Salesman Problem. https://doi.org/10.12982/CMUJNS.2019.0003. CMU J. Nat. Sci. (2019) Vol. 18(1). 27-37.

[46] Brocki, Ł., Koržinek, D. (2007). Kohonen Self-Organizing Map for the Traveling Salesperson Problem. In: Jabłoński, R., Turkowski, M., Szewczyk, R. (eds*) Recent Advances in Mechatronics*. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-73956-2_24.

[47] Frederico Carvalho Vieira, Adriao Duarte Doria Neto and Jose Alfredo Ferreira Costa, "An Efficient Approach to the Travelling Salesman Problem Using Self-Organizing Maps*", International Journal of Neural Systems*, vol. 13, no. 2, pp. 59-66, 2003. doi: 10.1142/S0129065703001443.

[48] Nourmohammadzadeh, A., Voß, S. (2021). Hybridising Self-Organising Maps with Genetic Algorithms. In: Simos, D.E., Pardalos, P.M., Kotsireas, I.S. (eds) Learning and Intelligent Optimization. 265–282. LION 2021. *Lecture Notes in Computer Science()*, vol 12931. Springer, Cham. https://doi.org/10.1007/978-3-030-92121-7_22.

[49] Grice, J.V., Montgomery, D.C.: Design and analysis of experiments. Technometrics 42(2), 208-209 (2000). https://doi.org/10.2307/1271458.

[50] Hui-Dong Jin, Kwong-Sak Leung, Man-Leung Wong and Zong-Ben Xu. 2003. An Efficient Self-Organizing Map Designed by Genetic Algorithms for the Traveling Salesman Problem. doi: 10.1109/TSMCB.2002.804367. IEEE Trans. on Systems, Man, and Cybernetics Part B: Cybernetics, vol. 33, pp. 877-888.

[51] Narayan Vishwanathan, Donald C. Wunsch. 2001. ART/SOFM: A Hybrid Approach to the TSP. https://doi.org/10.1109/IJCNN.2001.938771. IEEE. IJCNN'01. *International Joint Conference on Neural Networks*. Proceedings (Cat. No.01CH37222). 2254-2257.

[52] Teresa Serrano- Gotarredona, Bernabe Linares – Barranco and Andreas G.Andreou. "Adaptive Resonance Theory Microchips." Kluwer Academic Publishers. 737-746. 1999. https://doi.org/10.1007/BFb0098232.

[53] Jean-Charles Creput, Abderrafiaa Koukam. A memetic neural network for the Euclidean traveling salesman problem. *Neurocomputing* 72 (4-6) (2009) 1250–1264. https://doi.org/10.1016/j.neucom.2008.01.023.

[54] Bihter Avşar, Danial Esmaeili Aliabadi. Parallelized neural network system for solving Euclidean traveling salesman problem. *Applied Soft Computing* Volume 34, September 2015, Pages 862-873. https://doi.org/10.1016/j.asoc.2015.06.011.

[55] Haruna MATSUSHITA, Yoshifumi NISHIO. 2006. CHAOSOM and its Application to Traveling Salesman Problem. Tokushima University.

[56] A. L. Hodgkin and A. F. Huxley, "A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve," *Journal of Physiology*, vol. 117, pp. 500-544, 1952. doi: 10.1113/jphysiol.1952.sp004764.

[57] Remo Ryter, Michael Stauffer, Thomas Hanne, Rolf Dornberger. Analysis of Chaotic Maps Applied to Self-Organizing Maps for the Traveling Salesman Problem. 2015 IEEE *Congr. Evol. Comput*. CEC 2015 - Proc., pp. 1717–1724, 2015. doi: 10.1109/CEC.2015.7257094.

[58] M. Henon, "A two-dimensional mapping with a strange attractor," *Communications in Mathematical Physics*, vol. 50, pp. 69–77, feb 1976. https://doi.org/10.1007/BF01608556.

[59] E. ELabbasy, H. Agiza, H. El-Metwally, and A. Elsadany, "Bifurcation analysis, chaos and control in the burgers mapping," *International Journal of Nonlinear Science*, vol. 4, no. 3, pp. 171–185, 2007.

[60] J. Burgers, "Mathematical examples illustrating relations occurring in the theory of turbulent fluid motion," in Selected Papers of J. M. Burgers, F. Nieuwstadt and J. Steketee, Eds. *Springer Netherlands*, 1995, pp. 281–334. http://dx.doi.org/10.1007/ 978-94-011-0195-0 10.

[61] Verhulst, "Recherches mathmatiques sur la loi d'accroissement dela population." *Nouveaux mmoires de l'Acadmie Royale des Sciences et Belles-Lettres de Bruxelles*, vol. 18, pp. 14–54, 1845. http://eudml.org/doc/182533.

[62] K. Alligood, T. Sauer, and J. Yorke, Chaos: An Introduction to Dynamical Systems, ser. Chaos: An Introduction to Dynamical Systems. Springer, 1997. http://books.google.ch/books?id=48YHnbHGZ AgC.

[63] Arindam Chaudhuri, Kajal De, Dipak Chatterjee. A Study of the Traveling Salesman Problem Using Fuzzy Self Organizing Map. 2008 *IEEE Region 10 and the Third international Conference on Industrial and Information Systems*. doi: 10.1109/ICIINFS.2008.4798469.

[64] Applegate D., Bixby R., Chvatal V., and Cook W., On the solution of traveling salesman problems. *Doc.Math.J.DMV* Extra Volume ICM III, pages 645-656, 1998. crpc-tr98744.

[65] Hiromi Takano, Yutaka Shirai, Naofumi Matsumoto. 2010. PERFORMANCE EVALUATION OF HYBRID PROCEDURE OF SELF-ORGANIZING MAP AND SA FOR TSP. *International Journal of Innovative Computing, Information and Control* Volume 7, Number 5(B), May 2011. 2931-2944.

[66] J. Chen, J. Chen and X. Zhang, "Solving the Traveling Salesman Problem Using Elastic Net Integrate with SOM," 2007 *IEEE International Conference on Automation and Logistics*, 2007, pp. 2234-2237, doi:10.1109/ICAL.2007.4338947.

[67] Vlad-Ioan Lupoaie, Ivona-Alexandra Chili, Mihaela Elena Breaban, and Madalina Raschip. 2019. SOM-Guided Evolutionary Search for Solving MinMax Multiple-TSP. 2019 IEEE *Congress on Evolutionary Computation (CEC*). doi: 10.1109/CEC.2019.8790276. 73-80.

[68] A. Kiraly, J. Abonyi. Optimization of multiple traveling salesmen problem by a novel representation based evolutionary algorithm. *In Intelligent Computational Optimization in Engineering*, pp. 241-269, Springer, 2011. https://doi.org/10.1007/978-3-642-21705-0_9.

[69] Necula, R., Raschip, M., Breaban, M. (2018). Balancing the Subtours for Multiple TSP

Approached with ACS: Clustering-Based Approaches Vs. MinMax Formulation. In: Tantar, AA., Tantar, E., Emmerich, M., Legrand, P., Alboaie, L., Luchian, H. (eds) EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI. *Advances in Intelligent Systems and Computing*, vol 674. Springer, Cham. https://doi.org/10.1007/978-3-319-69710-9_15.

[70] Anmin Zhu and Simon 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. doi: 10.1109/RISSP.2003.1285655.

[71] Meghan Steinhaus, Arash Nasrolahi Shirazi, and Manbir Sodhi. Modified Self Organizing Neural Network Algorithm for Solving the Vehicle Routing Problem. 2015 IEEE *18th International Conference on Computational Science and Engineering*. doi: 10.1109/CSE.2015.56.

[72] H. E. Ghaziri, "Solving routing problems by a self-organizing map," in Artificial Neural Networks, 1991, *International Conference on Artificial Neural Networks*. ICANN, 1991, pp. 829–834.

[73] Y. Matsuyama, "Self-organization via competition, cooperation and categorization applied to extended vehicle routing problems," in Neural Networks, 1991., IJCNN-91-*Seattle International Joint Conference on*, vol. 1. IEEE, 1991, pp. 385–390. doi: 10.1109/IJCNN.1991.155208.

[74] Ghaziri, H. (1996). Supervision in the Self-Organizing Feature Map: Application to the Vehicle Routing Problem. In: Osman, I.H., Kelly, J.P. (eds) Meta-Heuristics. Springer, Boston, MA. https://doi.org/10.1007/978-1-4613-1361-8_39.

[75] A. I. Vakhutinsky and B. Golden, "Solving vehicle routing problems using elastic nets," in Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 *IEEE International Conference on*, vol. 7. IEEE, 1994, pp. 4535–4540. doi: 10.1109/ICNN.1994.375004.

[76] M Tairi, Y Wang, and SW Yoon. A Dynamic Approach to the Traveling Salesman Problem with Drones Using a Self- Organizing Map Proceedings of the 9th Annual World Conference of the Society for Industrial and Systems Engineering, 2020 SISE Virtual Conference September 17-18, 2020. ISBN: 97819384961-9-6.

[77] Jaradat, A., Matalkeh, B., Diabat, W.: Solving traveling salesman problem using firefly algorithm and k-means clustering. In: 2019 IEEE *Jordan International Joint Conference on Electrical Engineering and Information Technology* (JEEIT). IEEE, pp. 586–589 (2019). https://doi.org/10.1109/jeeit.2019.8717463.

[78] Kohonen, T., "Self-organized formation of topologically correct feature maps," Biol. *Cybern*. 43, pp. 59–69, 1982, doi: 10.1007/BF00337288.

[79] Rashid Ahmad and Dohyeun Kim, "An Extended Self-Organizing Map based on 2-opt algorithm for solving symmetrical Traveling Salesperson Problem," *Neural Comput*. Applications, vol. 26, issue 4, pp. 987–994, May 2015, doi: 10.1007/s00521-014-1773-z.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

TABLE 1. ENRICHED SOM MODELS

Type	Method name	Date	Dataset & Iterations	Performance	The number of neurons depends on the cities	Useful Comments
Convex-Hull	TOPSOM	2021	52 to 16862 cities TSPLIB instances National TSP's instances 30000 iterations	Big Data	Neurons = 8 × Cities	In comparison with traditional SOM, TOPSOM gets better routes with up to a 7.67% decrease in terms of PDM, and a 3.19% reduction in average in terms of PDB.
Convex-Hull	ORC-SOM	2011	51 to 2392 cities TSPLIB instances 200 iterations	Big Data	Neurons = 2 or 3 × Cities	ORC-SOM is superior in average percentage to its counterparts in solution quality.
Convex-Hull	CHSOM	2008	70 to 532 cities TSPLIB instances 200 iterations	Not referred	Neurons = Cities	Obtained better solution quality (%from optimum) than Leung's algorithm on some TSPs, but not on all.
Convex-Hull	Expanding Property SOM	2005	2400 cities No library found 100 * n iterations	Big Data	Neurons = Cities	The width is decreased linearly to 1 in the first 65% of the iterations.
Convex-Hull	ESOM	2004	50 to 2400 cities TSPLIB instances 100 * n iterations	Small Data	Neurons = Cities	4.18% solution quality on average on 20 TSPs Compared to KNIES-global, KNIES-local, SA, and Budinich.
Growing Ring	MGSOM	2005	6 to 442 cities TSPLIB instances 20 iterations	Big Data	Neurons = 2 × Cities	2.3215 average deviations from the optimal tour length compared to KL, KG, and SETSP.
Growing Ring	EGSOM	2006	51 to 106 cities TSPLIB instances No iterations found	Big Data	Neurons = 1.5 × Cities	2.4925 percent difference from the optimal tour and 233.8 average processing time (secs.)
Growing Ring	MSTSP	2006	51 to 442 cities TSPLIB instances 20 iterations	Big Data	Neurons = 2 × Cities	The average deviation of 2.4372% from the optimal tour length.
Heuristic	Enhanced SOM	2021	50 to 200 cities No library found 100000 iterations	Not referred	Neurons = Cities	0.07891 of F1 score achieved.
Heuristic	SDP	2005	100000 cities TSPLIB instances 50 to 500 iterations	Big Data	Neurons = 5 × Cities	The results show a very nice speedup behavior and make the solution of very large TSPs viable, up to hundreds of thousands of cities.
Heuristic	CAN	2003	51 to 85900 cities TSPLIB instances Cities / 2 iterations	Big Data	Neurons = 2.5 ×Cities	CAN and ORCSOM seem to be more robust.
Heuristic	Modified CAN	2011	6 to 574 cities TSPLIB instances Cities / 2 iterations	Small Data	Neurons = 2.5 × Cities	The error in the CAN algorithm is negligible for small problems.
Heuristic	GSOA	2018	51 to 2392 cities TSPLIB instances 150 iterations	Big Data	Neurons = 2.5 × Cities	Faster computation time and better results than ORC-SOM.
Heuristic	KNIES	1999	51 to 532 cities TSPLIB instances No iterations found	Small Data	Neurons = Cities	2.73% relative deviation from the optimal tour length. Better than PKN, GN, AVL, KG.
Heuristic	KNIES DECOMPOSE	2003	51 to 2392 cities TSPLIB instances No iterations found	Big Data	Neurons = Clusters	It is not the number of clusters, but the quality of the clustering which is crucial; this results in both a higher speed and accuracy at the same time.
Heuristic	Modares et al.	1997	51 to 1400 cities TSPLIB instances No iterations found	Big Data	Neurons = 2 × Cities	The deviation from the best-known solutions are 2.17% in the worst case and 1.22% on average.
Heuristic	Bernard Angeniol	1988	1000 cities No library found No iterations found	Big Data	Neurons = Cities	Compared with the elastic net method presented by Durbin and Willshaw.

TABLE 2. HYBRID MODELS

Type	Method name	Date	Dataset & Iterations	Performance	The number of neurons used depends on the cities	Useful Comments
Greedy	Self-Organizing Iterative	2018	51 to 200 cities TSPLIB instances No iterations referred	Small Data	Neurons = Cities	5-10% solution quality (%from optimum).
Greedy	Mueller & Kiehne	2015	52 to 561 cities Welch Two Sample t-test iterations > 2000	Not referred	Neurons = Cities	The hybrid approach outperforms ACO and SOM at a level of 5%.
Evolutionary	CVOA + SOM	2021	14 to 99 cities TSPLIB instances 2500 or 3000 iterations	Big Data	Neurons = Cities	Competed against classic SOM.
Evolutionary	Hybrid	2018	51 to 442 cities TSPLIB95 instances 500000 iterations	Big Data	Neurons = 2 × Cities	Less processing time but not a better quality solution on most of them.
Evolutionary	2opt + SOM	2007	51 to 10000 cities TSPLIB instances 25000 iterations	Big Data	Neurons = Cities	Its speed might be impressive, but it still is slow.
Evolutionary	SETSP	2003	70 to 442 cities TSPLIB instances iterations > 30	Big Data	Neurons > Cities	3.69 Average deviation from the optimal tour length compared to KL, KG.
Genetic	Clustered SOM	2021	100 to 10000 cities TSPLIB instances iterations > 50	Big Data	Neurons = Cities	Compared with SOM-Firefly [77], K-means-GA, and K-means-Firefly.
Genetic	EvolvedISOM (eISOM)	2003	30 to 2393 cities TSPLIB instances 160 * n iterations	Big Data	Neurons = Cities	3.24% solution quality on the optimum length.
Genetic	ART/SOFM	2001	1000 to 14000 cities TSPLIB instances No iterations referred	Big Data	Neurons = Cities	Percentage excess over the Lin Kernighan tour.
Memetic	MEMETIC	2008	29 to 85900 cities TSPLIB instances No iterations referred	Small Data	Neurons = 2 × Cities	Better %PDM and %PDB on most of its competitors especially on small scale.
Memetic	PMSOM	2015	29 to 10639 cities TSPLIB instances 60 to 480 iterations	Big Data	Neurons = 5 × Cities	This methodology is more than 4 times faster than those of non-parallel systems on average.
Chaotic	CHAOSOM	2006	48 cities TSPLIB instances 100 iterations	Not referred	Neurons = 2 × Cities	2.02% error from the optimal distance.
Chaotic	Chaotic maps	2015	50 to 10000 cities OpenOpal framework 1000 to 1000000 iterations	Depending on the model	Neurons = Cities	Burgers, Delayed Logistics, or Tinkerbell Map are for big problems and Henon Map is for small sets.
Fuzzy	Fuzzy	2008	51 to 1200 cities TSPLIB instances 25000 iterations	Big Data	Neurons = Cities	It has better average percentage results than Lin Kernighan.
Simulated annealing	SOM + SA	2010	10 to 2000 cities TSPLIB instances Iterations = cities * 5	Big Data	Neurons = Cities	Its computational complexity is approximately $O(k \times n^{1.5})$.
Elastic Net	EN + SOM	2007	1000 to 2000 cities No library found	Not referred	Neurons = 2.5 × Cities	The integration of SOM and EN has proven to be a very efficient way to

			10 million iterations			optimal route planning.

TABLE 3. ENRICHED SOM COMPETITION

Method name	Kohonen	CAN	eISOM	ESOM	Elastic net	AVL	GN	KL	ORC-SOM	Somhom's	Matsuyama
TOPSOM	***X***	***X***	***X***								
ORC-SOM	***X***			***X***							
CHSOM				***X***		***X***					
Expanding Property SOM				***X***							
ESOM					***X***						
MGSOM	***X***					***X***	***X***	***X***			
MSTSP						***X***	***X***	***X***			
EGSOM									***X***		
CAN				***X***					***X***		
KNIES					***X***						
Bernard Angeniol		***X***							***X***		
GSOA		***X***								***X***	
Modified CAN								***X***			
KNIES DECOMPOSE					***X***	***X***	***X***				***X***
Modares et al.	***X***	***X***	***X***								

TABLE 4. HYBRID COMPETITION (Hybrid models against the algorithms/models/authors of a model taken from their papers)

Method name	Greedy	ACO	Kohonen	Peker	Yongquan Zhou	AVL	GN	KL	MEMETIC
Self-Organizing Iterative	***X***								
Mueller & Kiehne		***X***	***X***						
CVOA + SOM			***X***						
Hybrid				***X***	***X***				
SETSP			***X***			***X***	***X***	***X***	
SOM + SA			***X***						
PMSOM									***X***

The continuation of the TABLE 5.

Method name	ESOM	SA	Budinich's SOM	CEN	Lin Kernighan	CAN	eISOM	Kohonen	Evolutionary	K-means-Firefly (combination)
Clustered SOM										***X***
EvolvedISOM (eISOM)	***X***	***X***	***X***	***X***						
ART/SOFM					***X***					
MEMETIC	***X***					***X***	***X***			
CHAOSOM					***X***			***X***		
Fuzzy					***X***				***X***	***X***
2opt + SOM					***X***				***X***	

TABLE 5. MULTIPLE TSP MODELS

Method name	**Date**	**Dataset & Iterations**	**Performance**	**The number of neurons depends on the cities**	**Useful Comments**
MinMax Multiple-TSP	2019	51 to 99 cities TSPLIB instances 5000 iterations	Not specific	Neurons = 3 × Cities	The optimum solution is obtained on some problem instances.
MTSP	2003	29 to 561 cities TSPLIB instances 100 iterations	Not referred	Neurons = Cities	Better length of the average solution and running time than conventional SOM.

TABLE 6. MULTIPLE TSP COMPETITORS

Method name	ACO	SOM	SOM-ACO	EA	SOM-EA	SOM-EA-2opt	Modares et al.
MinMax Multiple-TSP	***X***	***X***	***X***	***X***	***X***	***X***	
MTSP							***X***

TABLE 7. MODELS CATEGORIZED

Heuristic	Enhanced SOM	SDP	CAN	Modified CAN	KNIES	KNIES DECOMPOSE	Modares et al.	GSOA	Bernard Angeniol
Convex-Hull	ORCSOM	ESOM	TOPSOM	CHSOM	Expanding Property SOM				
Evolutionary	CVOA + SOM	SETSP	Hybrid	2opt + SOM					
Genetic	Clustered SOM	eISOM	ART/SOFM						
Multiple Salesman	MTSP	MinMax Multiple-TSP							
Greedy	Iterative	Mueller & Kiehne							
Chaotic	Chaotic Maps	CHAOSOM							
Memetic	MEMETIC	PMSOM							
Fuzzy	Fuzzy								
Simulated Annealing	SOM + SA								
Elastic Net	EN + SOM								