Solving the Capacitated Vehicle Routing Problem with a Self Organizing Map Using Fuzzy Logic for Parameter Control

Meghan Steinhaus[a,*], Arash Nasrollahishirazi[b], Manbir Sodhi[b]

[a] *United States Coast Guard Academy, 15 Mohegan Ave, New London, CT 06320*
[b] *University of Rhode Island, 92 Upper College Rd, Kingston, RI 02881*

Abstract

Although a significant amount of research has focused on the use of Artificial Neural Networks (ANNs) for solving the Traveling Salesman Problem (TSP), there is considerably less attention on the application of ANNs to the Vehicle Routing Problem (VRP). This paper proposes an updated Self Organizing Map approach to solving the Capacitated Vehicle Routing Problem (CVRP). The proposed algorithm extends existing research by strengthening the bias term and introducing a *restricted* mechanism in order to encourage the network to explore a greater number of feasible solutions. In addition to the bias term update, the proposed algorithm overcomes parameter sensitivity issues with the incorporation of fuzzy logic for automatic parameter control. Numerical results from benchmark problems demonstrate that the proposed algorithm without automatic parameter control is superior to the existing SOM approaches in the literature, and the incorporation of automatic parameter control eliminates the need for manual parameter tuning with minimal impact on results. The proposed algorithm is a constructive heuristic and, in accordance with the literature, is benchmarked against two other well-known constructive heuristics: the Sweep algorithm and the Clarke and Wright algorithm with superior results across all benchmark problems. T

Keywords: Vehicle Routing Problem, Self Organizing Map, Combinatorial Optimization, Artificial Neural Networks

1. Introduction

Since the introduction of the Hopfield Neural Network (HNN) for solving the Traveling Salesman Problem (TSP) three decades ago [1, 2, 3], Artificial Neural Networks (ANN) have been applied to nearly every type of combinatorial optimization problem (COP) [4]. Following the introduction of the HNN, two additional ANN approaches for solving the TSP were proposed: the Elastic Net (EN) method [5], and the Self Organizing Map (SOM) [6, 7, 8]. The EN and SOM both offer a geometric approach to solving the TSP. Subsequent research on the use of neural networks for solving COPs can be related to either the HNN approach or the SOM approach [4], where the EN is grouped with the SOM as a result of their similarities. The vast majority of research into the use of ANNs for combinatorial optimization has focused on the HNN [9] since it is easily generalized to a wide range of COPs. The geometric nature of the SOM limits its application across all COPs [10]. For COPs that are geometric in nature, however, such as the Euclidean TSP and Vehicle Routing Problem (VRP); the SOM approach appears to surpass the performance of the HNN [11, 12].

This paper is concerned with the use of a SOM to solve the Capacitated Vehicle Routing Problem (CVRP). The CVRP is the most basic and widely studied VRP [13]. It is an NP-hard combinatorial optimization problem which is closely related to the TSP, yet it is significantly more difficult to solve [14]. The original SOM approaches to solving a TSP [5, 6, 7, 8] have been extended to the VRP [15, 16, 17, 18, 19, 20, 21, 22].

Questions regarding the effectiveness of ANNs for combinatorial optimization problems date back to the Wilson and Pawley [23] critique of the HNN for the TSP. Despite continued research with competitive results, the application of ANNs to COPs was never able to overcome the early criticism [4]. The limited amount of ANN research into the VRP was described in 2007 by Laporte [14], as 'rather unsuccessful and this line of research seems to have been abandoned.' This criticism is likely rooted in two issues with the existing literature. First, not all research into the use of ANNs for the TSP and VRP have made use of publicly available benchmark data sets [12], therefore it is difficult to draw solid conclusions about the effectiveness of these ANN approaches. Second, the results of the application of ANNs for the TSP and VRP do not appear to be competitive with the best known Operations Research (OR) heuristics [24, 12, 25]. Burke points out however [24], that the SOM approach is a constructive heuristic, and should only be compared with similar approaches to draw conclusions about its performance. Contradictory to Laporte's description, other researchers encourage a continued focus into the use of ANNs for combinatorial opti-

*Corresponding Author.
Email address: mksteinhaus@gmail.com (Meghan Steinhaus)

mization problems, specifically the use of the SOM for the TSP and VRP [12, 25] for the following reasons:

- Over time, the incremental improvements to the SOM might lead to more competitive approaches to the VRP.
- The SOM algorithm is easily parallelized, and might become competitive as massively parallel computers become more widely available.
- Insights from the application of the SOM to the VRP might result in improvements in heuristics for other combinatorial optimization problems.
- Unlike the HNN, the SOM can be applied to very large problem instances.

The purpose of this paper is two-fold. First, this paper will present an updated SOM algorithm for solving the CVRP. The proposed algorithm will improve upon the best known SOM results for solving the CVRP, and the proposed algorithm will incorporate fuzzy logic for automatic parameter control. The second goal of this paper is to provide a side by side comparison of the proposed SOM algorithm to two other constructive heuristics for the CVRP: the Sweep algorithm [26] and the Clarke and Wright algorithm [27]. This comparison will help provide clear insight into the effectiveness of the SOM approach to solving the CVRP.

2. Background

2.1. Vehicle Routing Problem

The vehicle routing problem (VRP) is a well-known combinatorial optimization problem concerned with finding the optimal routing of goods between a central depot and a set of customers [13]. The CVRP introduces a restriction on the vehicle capacities. This paper is concerned with the symmetric CVRP, which is described as follows. Let $G = (V, A)$ be an undirected, complete graph, where $V = \{0, \ldots, n\}$ is the set of nodes, and A is the set of arcs which connect the nodes. A cost, c_{ij}, is associated with each arc, $(i, j) \in A$, where $i \neq j$. In this research the cost associated with each arc is the Euclidean distance between the nodes i and j, and because the problem is symmetric, $c_{ij} = c_{ji}$. Node 0 corresponds to the depot, whereas the remaining nodes, $V' = V \setminus \{0\}$, correspond to the customers. Each customer $i \in V'$ has a deterministic, non-negative demand, d_i, for goods. The goods are transported between the depot and the customers by one of the vehicles in the known, homogeneous set of K vehicles. Each vehicle has a maximum capacity, Q, that is identical across the set of vehicles, and $d_i \leq Q$ for each $i = 1, \ldots, n$. The objective of the CVRP is to find K vehicle routes that start and end at the depot, with the minimum cost, where the demand of each customer is met, and the capacity of each vehicle is not violated.

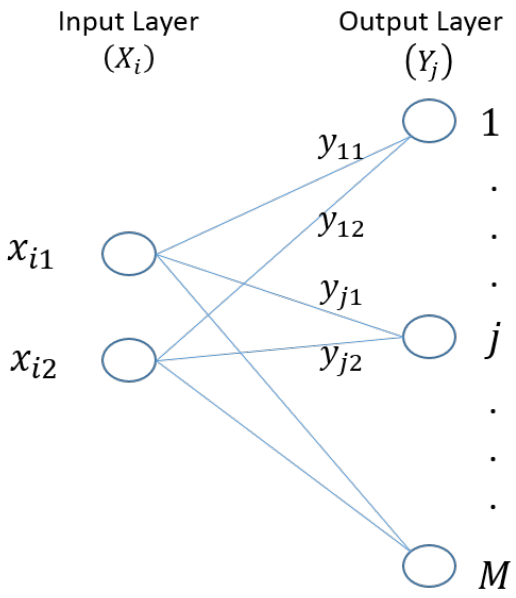


Figure 1: A two layer SOM network for TSP

The TSP is a special case of the CVRP, where the capacity of a vehicle is unbounded, $Q = \infty$, and there is only one vehicle in the fleet, $K = 1$. First, the general application of the SOM to the TSP will be presented, followed by the application of the SOM to the CVRP.

2.2. Self Organizing Map

A general description of the SOM approach to the TSP is based upon the 'elastic band' idea that originates from the EN [5]. The neurons in the SOM are ordered along a circular band, and as the network evolves, the band of neurons is stretched toward the cities until each city is eventually associated with one neuron on the band. This association between the neurons and cities corresponds to a solution. The extension of the SOM algorithm from the TSP to the VRP requires the modification of the learning process. For the VRP, the rules which govern how the nodes are updated must now incorporate some type of penalties which will account for the capacity constraints in the VRP.

SOM for the TSP. The SOM is applied to an N city TSP using a two layer network with M output nodes where $M \geq N$. The input node, X_i, represents the coordinates of a city x_{i1}, x_{i2}, whereas the coordinates of the output nodes, Y_j, are represented by the connection weights between the layers y_{j1}, y_{j2}. The input and output layers are fully connected, as in Figure 1, and the weights between the layers are initialized to small values such that the neurons form a small circular ring near the center of the cities.

Each city i, where $i = 1, \ldots, N$, is iteratively input to the network, and the output nodes, Y_j, compete based on Euclidean distance to be declared the winner. If X_i represents the coordinates of the i^{th} input city, and Y_j represents the weights of the j^{th} node, the winning node, J, is

found to be the node with the smallest Euclidean distance to the input city, according to the following equation.

$$J = \text{ArgMin}_j\{\|X_i - Y_j\|\} \quad (1)$$

After the winning node J is found for a city, the position (weights) of J and its neighbors are updated as follows:

$$Y_j^{new} = Y_j^{old} + \mu F(d_j)(X_i - Y_j^{old}), j = 1, \ldots, M \quad (2)$$

In (2), μ is the learning rate and d_j is the number of nodes between node j and the winning node J. This distance is found as follows:

$$d_j = min[|j - J|, M - |j - J|] \quad (3)$$

The function $F(\cdot)$ is defined in such a way that as the network evolves, the influence of the winning node on its neighbors decreases. The algorithm continues to loop through the set of cities, presenting one city at a time to the network, causing the weights of the winning node and its neighbors to be updated. The algorithm terminates when there is one node sufficiently close to each city, or when a max number of iterations has been reached.

SOM for the CVRP. To extend the SOM application from the TSP to the CVRP, two factors must be considered: 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 by both ring (route) and position within the route, Y_j^k.

To account for the capacity constraint of the CVRP, two approaches have been followed in the literature. Both of these approaches 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 to choosing the winning node J [16]. 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 toward zero as the network evolves.

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 using (1) [15]. With 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 in (1) as the nodes compete [19, 20, 21]. With this approach, a winning node has the smallest combination of Euclidean distance and bias term. Amongst the published results of the application of the SOM to the CVRP that utilize benchmark problem sets, it is clear that results from the bias term approach in [19, 20, 21] surpass the results from previous SOM approaches [16, 17, 18].

The SOM approach to the CVRP is usually initialized with K small petaloids that start and end at the depot [19, 20, 21]. The network evolves as a winning node, J, is identified for each input city, X_i, and the positions of the winning node and its neighbors are updated. Figure 2 graphically illustrates how a SOM network evolves. A solution is found when each city has a node within a specified distance, or when a maximum number of iterations has occurred.

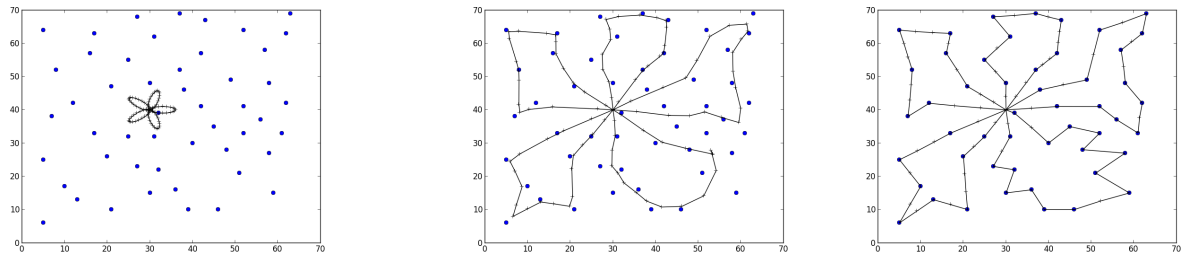
There are numerous variations of the SOM approach to the TSP and VRP, and a complete literature survey of these SOM algorithms can be found in [12, 25]. Cochrane and Beasley summarize four general areas where the SOM approaches differ from one another:

- number of output neurons, Y_j, used in the network
- order in which the cities are presented to the network
- equation for finding the 'winning' neuron J
- the number of neighbors updated, as well as the equation for updating neurons

Shortcomings of SOM for the CVRP. The results of the bias term approaches of applying the SOM to the CVRP of Torki, et al [19] and Modares, et al [20], appear to be promising. Although these approaches differ slightly in their bias term calculations, their results are identical, and all solutions are with 7% of the best known solution at the time of publication. In an attempt to replicate these results, however, it became clear that details regarding parameter settings and the bias term updates were critical to the quality of the published results. Despite using various parameter settings, and multiple methods for updating the bias term, the results could not be replicated. Many of the results were either infeasible, or feasible but of poor quality. This suggested the criticality of the parameter settings in order to strike a balance between the competing demands of finding a short route while not violating the capacity constraint.

The results of Schwardt et al [21] also seemed promising, but the authors indicate that different parameter settings were used for the various test problems, and details of these parameter settings were not provided. Schwardt does, however, address the fact that the SOM can result in infeasible solutions, which is why the parameter settings are adjusted for each problem. Schwardt also introduced additional mechanisms into the SOM in order to achieve feasible solutions more often.

Reviewing these previous results, it is clear that the parameter settings are critical in the solution quality for the bias term approach of using the SOM for the CVRP. Success of this approach can only be found if there is an appropriate balance between the competing demands of cost



(a) Initial SOM (b) after 60 iterations (c) Final solution, 178 iterations

Figure 2: Evolution of SOM network for CVRP with N=50 and K=5

and feasibility for a solution. The shortcomings of the current literature for the SOM application to the CVRP are that the few published results using benchmark data sets do not offer enough details and insight into finding the appropriate parameter settings for the algorithm.

3. Updated SOM for CVRP

This section proposes an update to the bias term approach to the SOM for application to the CVRP. The proposed algorithm improves upon these previous results by strengthening the bias term as well as introducing a new *restricted* mechanism to help increase the feasibility of solutions. When developing the proposed algorithm, careful consideration was given to the four general areas where SOM approaches differ as described in Section 2.2. The parameter settings for this updated approach are explored.

3.1. Proposed Algorithm

For a CVRP problem with N cities and K vehicles, where each city has a demand q, and each vehicle has a capacity Q, the proposed algorithm is initialized with M neurons divided amongst K rings. These K rings of neurons are initialized into small petaloids, where each is centered at a random point in the plane that is bound by the maximum and minimum X and Y coordinates amongst all of the N cities. Randomly initializing the location of the rings, instead of centering them at the depot, injects more stochasticity into the network and can lead to a more robust search of the solution space.

After the network is initialized, it moves into the competition phase. During this phase the nodes repeatedly compete to be declared 'winner' for each city including the depot. One epoch consists of all cities and depot being presented to the network once. At the beginning of each epoch, the cities are randomly ordered and presented to the network one at a time. Randomly ordering the cities is another means to increase the robustness of the algorithm [20]. When a city is presented, one winner is declared amongst all of the nodes, whereas when the depot is presented, a winner is declared on each route. An *inhibit* term is used to ensure that a node may only be declared a winner once per epoch, as described in [20]. This helps force the separation of nodes on each ring. The competition rules are discussed next.

Accounting for the capacity constraint is the most critical factor to ensure the feasibility of solutions when extending the SOM from the TSP to the CVRP. The proposed algorithm incorporates a bias term in the competition rule in order to account for the capacity constraint. When the nodes compete to be declared the 'winner' for a city, X_i, the competition is based on a combination of a Euclidean distance and the bias term as follows:

$$J = \text{ArgMin}_j\{\|X_i - Y_j\|\} + \nu B_k \tag{4}$$

where

$$\nu = \frac{\nu}{1 + \delta\nu} \tag{5}$$

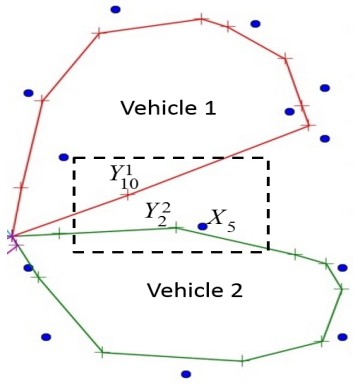
and

$$B_k = \left(1 + \frac{\sum_l q_l^k}{Q}\right)^2 \tag{6}$$

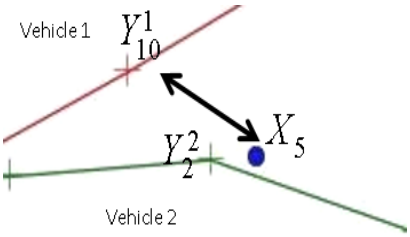
Where ν is initialized to a value that is proportional to the average distance between each node and the depot; and δ controls how quickly ν decreases and is initialized to a small value.

The Euclidean distance in (4) is used to help ensure a short route, whereas the second term, the bias term, is used to help find a feasible route. The bias term is calculated according to (6), and its purpose is to penalize overloaded routes. The term $\frac{\sum_l q_l^k}{Q}$ in (6) provides a ratio of the sum of each demand l assigned to route k, to the available capacity on the route. The proposed bias term differs from the existing literature as it provides a larger difference in the penalties for routes that are near or over capacity.

Figure 3 shows the competition between nodes Y_{10}^1 and Y_2^2 for city X_5. Assuming that vehicle 2 is over capacity and has a high bias term, node Y_{10}^1 is declared the winner despite having a larger Euclidean distance from city X_5. At the end of each epoch, the bias term is updated. In order to update the bias term for each route, the demand q_l for each city, l, is assigned to the route that corresponds



(a) Nodes Y_{10}^1 and Y_2^2 compete for city X_5.



(b) Node Y_{10}^1 is declared winner.

Figure 3: Example of node competition with bias term; assume that vehicle 2 is overloaded and vehicle 1 is not.

to the closest node (measured only by Euclidean distance) at the end of the epoch.

As the network evolves, the Euclidean distance between nodes and cities decreases, therefore it is critical that the weight of the bias term also decreases. If the weight is not decreased accordingly, the bias term will dominate (4) resulting in solutions that are feasible, but of poor quality. The parameter ν serves to decrease the bias term weight, and ν is updated according to (5) after each epoch.

During the competition phase of algorithm, after the winning node J is found for a city, X_i, the position of the winning node and its neighbors are updated as follows:

$$Y_j{}^{k,new} = Y_j{}^{k,old} + \Delta Y_j{}^k \tag{7}$$

$$\Delta Y_j{}^k = \mu F(d_j)(X_i - Y_j{}^k) + \lambda(Y_{j+1}^k - 2Y_j{}^k + Y_{j-1}{}^k), \quad \{\forall j | F(d_j) \geq 0\} \tag{8}$$

The magnitude and direction that a node is updated is determined by (8). The first term in (8) moves the node, Y_j^k toward the current city, X_i. The magnitude of this movement is determined by the learning rate, μ; the neighborhood function, $F(\cdot)$; and the current distance between the node and city, $(X_i - Y_j{}^k)$. The second term in (8) originated from the Elastic Net algorithm. This term helps strengthen the force between neighboring nodes, and helps reduce the possibility of a route crossing over itself. The weight of this second term, λ, decreases at the end of each epoch in order to allow the network to converge.

As previously stated, the function $F(\cdot)$ is defined as a decreasing function, so that as the network evolves the winning node has less influence on its neighboring nodes.

$$F(G, d_j) = \begin{cases} \exp^{((-d_j{}^2)/(G^2))}, & d \leq H \\ 0, & otherwise \end{cases} \tag{9}$$

Where $H = .2 * M$, and G is the gain parameter that decreases as the network evolves.

At the end of each epoch, an intermediate solution is found by determining the closest node to each city. Each node cannot be assigned to more than one city. This intermediate solution is used to update the bias term B_k according to (6). The parameters ν, λ, and G are also decreased at this time.

Although the proposed algorithm provides a strong bias term for penalizing routes that are near or over capacity; it is possible for the solution to be infeasible. In order to overcome this, a *restricted* term is introduced to encourage the network to evolve toward a feasible solution. When the intermediate solution is found at the end of each epoch, any route that is over capacity Q becomes *restricted*. During each epoch, a *restricted* route cannot be declared a 'winner' for cities beyond its capacity Q. This *restricted* mechanism forces the network to explore more feasible solutions, and evolve the network accordingly. Once *restricted*, a route can remain *restricted* for a certain number of iterations, or until the route is no longer overloaded in the intermediate solution. An outline of the algorithm follows.

Step 1: Initialization

Initialization: Initialize parameters μ, δ, λ, G, and α; set $\nu = c *$ (average distance of all cities to the depot). Let $N =$ the number of cities plus the depot, and $M =$ the number of nodes; and let *closeEnough* be the maximum distance required between a city and node for termination. Construct the nodes into K small petaloids placed randomly in the plane.

Step 2: Randomization

Randomization: Randomize the order of cities. Let i be the index of each city. Set $i = 1$. Set *inhibit* = *False* for each node; *restricted* = *False* and *capacityCount* = 0 for each route.

Step 3: Competition

Competition: Present city X_i to the network. Using (4), all qualified[1] nodes compete to determine the

Table 1: Parameter settings tested

Settings		
run	δ	c
1	.01	1
2	.01	10
3	.01	100
4	.005	1
5	.005	10
6	.005	100

winning node J. If the depot is presented, one winning node is found on each of the K routes. Set *Inhibit* = *True* for the winning node(s), and update the $capacityCount_k$ for the route that corresponds to node J.

Step 4: Adaptation

Adaptation: Move node J and its neighbors using (7).

Step 5: Increment

Increment: Set $i = i + 1$. If $i \leq N$, go to Step 2; otherwise, go to Step 6.

Step 6: Test Convergence

Test Convergence: If each city has a node within a *closeEnough* distance, STOP. Otherwise, set $\lambda = \lambda(1 - \alpha)$; $G = G(1 - \alpha)$; update ν according to (5); update B_k according to (6); update *restricted* routes; and go to Step 2.

3.2. Parameter Sensitivity

Computational experiments were conducted with the updated SOM algorithm on ten standard CVRP instances [28, 29] from the literature. The same problem instances were used as Torki et al, and Modares et al, in order to make a clear comparison between the algorithms. Additionally, three parameters were varied in this experiment: the number of epochs that a route is *restricted*, c, and δ. Two settings were used for the number of epochs that a route remains *restricted*. The first setting keeps the route *restricted* until the route is no longer overloaded in the intermediate solution; the second setting keeps a route *restricted* for ten epochs from when it is first categorized as *restricted*. Within each of these two settings, the parameters c and δ varied according to Table 1.

Each of the parameters that were chosen to vary have a significant influence on the ability of the algorithm to find a feasible solution. The parameter c is used to initialize the weight of the bias term at the beginning of the algorithm. The algorithm updates and uses δ at the end of each epoch to decrease the weight of the bias term. The remaining parameters were initialized as follows: $\mu = .6$; $\lambda = .3$; $G = .4 * M$; $\alpha = .03$; and $M = 4 * N$. These initial parameter settings are largely based upon the works of Torki, et al, and Modares, et al [19, 20].

For each parameter setting, 100 replications of the SOM were conducted, and the best solution amongst all parameter settings was recorded in Table 2, along with the percent deviation from the best known solution (PDB). The results of the proposed algorithm are compared to the best known results from the CVRP SOM literature [19, 20, 21]. These results are found in Table 2.

One parameter setting did not lead to all of the best solutions. Each problem appeared to be uniquely sensitive to the parameter settings, and a few parameter and problem pairings led to no feasible solutions. This observation reinforces the importance of the parameters in maintaining a balance between the competing demands of finding a short solution and the feasibility of the solution. This finding is in line with the results of Scwardt and Dethloff [21] who reported having to adjust the parameter settings for each new problem.

Given the sensitivity of the algorithm's success to the parameters c and δ, it is desirable to control the algorithm's parameters automatically, instead of having to tune the parameters for each new problem. The intention is to identify those parameters that are most influential in keeping the balance between a short, yet feasible solution. Once these parameters are identified, all problems will start out with the same initial settings. As the network evolves, the automatic parameter control will tune these parameters in a manner that results in good quality, feasible solutions across all problems.

4. Fuzzy Logic Control

Background. A fuzzy logic controller (FLC) can be described as a model which can produce a mapping between an input and output where the mapping is defined using linguistic variables [30]. The linguistic variables are a means to incorporate expert, human knowledge into an automatic control strategy. Fuzzy logic has been used extensively for controlling parameters in the Genetic Algorithm [30, 31] and a similar approach can be applied to the current SOM algorithm for the CVRP.

The generic form of an FLC is shown in Figure 4. In the current application, the SOM algorithm is the controlled system. Performance measures (Process Output) will be extracted from the SOM as it is evolving, these measures will be fuzzified and then passed through the fuzzy decision making logic. The results of the fuzzy decision making logic will then be defuzzified and the SOM parameters will be updated accordingly.

The purpose of incorporating an FLC into the SOM is to provide parameter control based on how the algo-

[1] A node is qualified to compete if it is not inhibited, and if it is *restricted*, it cannot have exceeded the capacity Q for 'winning' nodes in one epoch.

Table 2: Best solutions found in 100 replications of the proposed algorithm

Name	Size	Best Known (no. vehicles)	Torki, et al [a]	Schwardt, et al	Proposed	
					Result	PDB
eil22	21	375.3 (4)	390 (4)	375.3 (4)	375.3 (4)	0
eil23	22	569 (3)	585 (3)	570.2 (3)	569.7 (3)	0.1
eil30	29	534 (3)	557 (3)	537.1 (3)	539.9 (3)	1.1
eil33	32	835 (4)	889 (4)	915.4 (4)	846 (4)	1.3
C1	50	524.6 (5)	537 (5)	526.3 (5)	524.6 (5)	0
C2	76	835.3 (10)	8763 (10)	902.73 (10)	860.3 (10)	3.0
C3	100	826.13 (8)	863 (8)	838.9 (8)	829.6 (8)	0.42
C11	120	1042.1 (7)	1066 (7)	1111.4 (7)	1049.7 (7)	0.73
C4	150	1028.4 (12)	1082 (12)	1074.4 (12)	1071.1 (12)	4.2
C5	199	1291.3 (16)	1386 (17)	1367.5 (17)	1365.3 (17)	5.73

[a] The results from Torki, et al, and Modares, et al are identical, therefore, they are combined into one column.

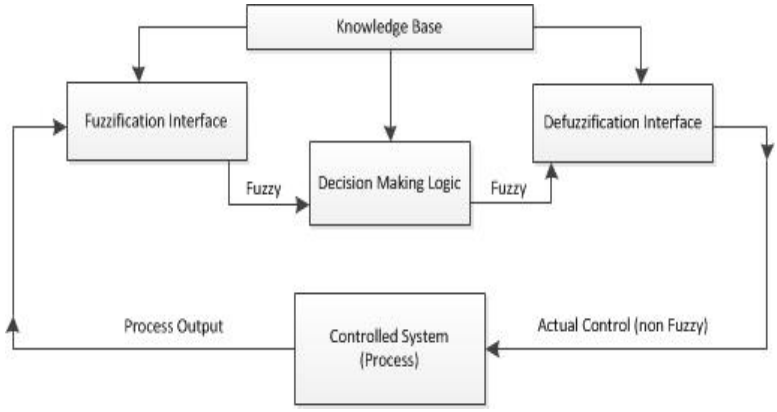


Figure 4: General structure of fuzzy logic controller [32]

rithm is performing. If the algorithm is converging toward a feasible solution, the appropriate parameters should be adjusted to relax the weight of the bias term. On the other hand, if the SOM is converging toward an infeasible solution, these parameters must be adjusted to increase the weight of the bias term with intent of encouraging the algorithm to find a feasible solution. The first step in designing the FLC is to determine what metrics will be used to assess the algorithm's performance.

Input. In order to measure how well the SOM is converging toward a feasible solution, two metrics are chosen. Each metric is evaluated at the end of an epoch, and are based on the current, intermediate solution of the SOM. The first metric x_1 (10) is the ratio of the maximum demand assigned to a route, to the truck capacity Q. If $x_1 \leq 1$, this indicates that all routes are feasible. If $x_1 \geq 1$, however, this indicates that at least one route is infeasible. The second metric x_2 (11) is a ratio of the current and previous x_1 values. This ratio provides an indication as to whether the feasibility of the solution is getting better, worse or staying the same.

$$x_1 = \frac{\text{Max}_k\{capacityCount_k\}}{Q} \tag{10}$$

$$x_2 = \frac{x_1^{curr}}{x_1^{last}} \tag{11}$$

Output. The next step is to identify which parameters will be controlled by the FLC. In the experiment described in Section 3.2, only the parameters c, *delta*, and the length of *restricted* were adjusted, but the algorithm consists of many other parameters that may be controlled. Table 3 provides a list of all the parameters in the SOM algorithm with a brief description, and if/how they are currently updated.

Two parameters were chosen to be updated by the FLC: and δ and α. In the current version of the algorithm, the value of δ is initialized at the beginning of the algorithm,

Table 3: Parameter Descriptions

Parameter	Description	Update Frequency	Update Rule
μ	learning rate	N/A	N/A
δ	used to decrement ν	N/A	N/A
λ	weight of neighbor positions in (8)	end of each epoch	$\lambda = \lambda * (1 - \alpha)$
G	gain parameter; input to (9)	end of each epoch	$G = G * (1 - \alpha)$
α	used to decrement λ and G	N/A	N/A
c	coefficient for initial value of ν	N/A	N/A
ν	weight of bias term	end of each epoch	$\nu = \frac{\nu}{1 + \delta\nu}$
closeEnough	min dist between node and city in final solution	N/A	N/A
restrictedLength	how long a route is restricted if it is overloaded	N/A	N/A

and held constant. δ is used to decrement the weighting factor ν at the end of the each epoch. The weight of the bias term must decrease as the algorithm evolves in order to ensure that the algorithm converges, however, how quickly the weight decreases has a significant impact on the quality of the solution. If the weight decreases too quickly, then there is a greater chance that the capacity constraint will be violated and the solution will be infeasible. However, if the weight decreases too slowly, this is likely to result in a solution that is feasible, but of poor quality with regard to route length. By allowing the FLC to control the value of δ, δ can be updated based on how well the algorithm is converging toward a feasible solution. If the intermediate solution is infeasible, the value of δ should be reduced, thus making the weighting factor ν decrease more slowly. If the algorithm is moving toward a feasible solution, however, δ can be increased to allow the algorithm to converge to a solution more quickly. Increasing and decreasing the value of δ will not change the fact that ν is a monotonically decreasing function, it will only change how quickly ν decreases.

The second parameter chosen to be controlled by the FLC is α. The role of α is to reduce the values of both G and λ, which are parameters related to the influence of neighboring nodes on each other. When a winning node is chosen, this winning node and its neighbors' positions are updated. The number of neighboring nodes which are updated decreases as the SOM evolves and is determined by 9. G is an input to 9, and as G decreases, so does the value of 9. The parameter λ is used in 8. When a node's position is updated, the movement is a combination of the node's distance to the current city, as well as the position of it's two immediate neighboring nodes. λ serves as the weight of the neighboring nodes positions for this update. When the neighboring nodes have a stronger influence in a node's positional update, this not only helps to ensure a shorter route, it also provides a more robust search of the solution space.

Table 4: Fuzzy Rules

Rule	IF		THEN	
	x_1	x_2	change α	change δ
1	low	better	increase	increase
2	low	same	increase	increase
3	low	worse	increase	steady
4	medium	better	same	same
5	medium	same	same	same
6	medium	worse	same	same
7	high	better	same	same
8	high	same	decrease	decrease
9	high	worse	decrease	decrease

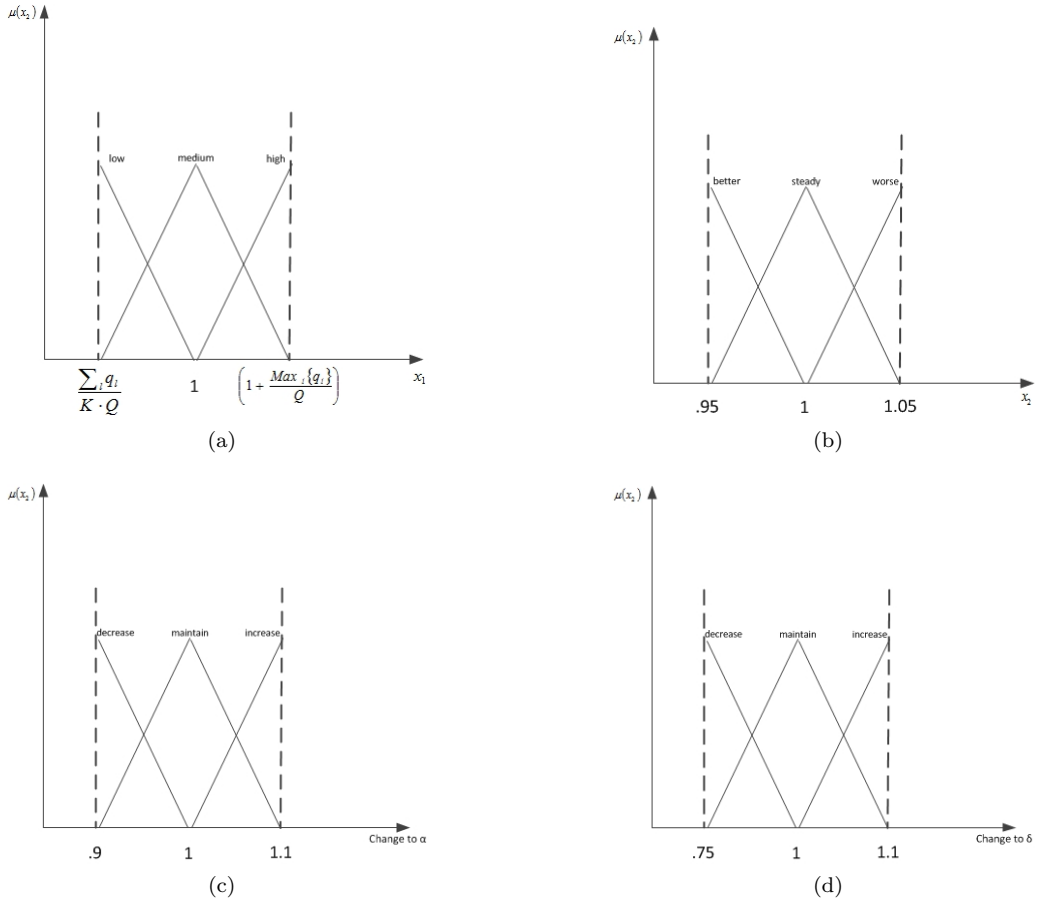
Fuzzifier/Inference/Defuzzifier. This experiment used triangular membership functions, the *AND* operator for fuzzification; the *OR* operator and the *clipping* method to determine the consequence; and the weighted average defuzzification strategy. The triangular membership functions can be found in Figure (5). With two inputs, x_1 and x_2, as well as two outputs, y_1 (α) and y_2 (δ), a total of 18 fuzzy rules were used in the FLC; 9 for each output. These fuzzy rules are found in Table (4).

4.1. SOM with Fuzzy Logic Control

The FLC is incorporated into Step 6 of the updated algorithm outlined in Section 3.1. The FLC is used to update the parameters α and δ every 10 epochs. In order to incorporate this update into the previously described, updated algorithm , Step 6 is be revised as follows:

Step 6: Test Convergence

Test Convergence: If each city has a node within a *closeEnough* distance, STOP. Otherwise, if the epoch number % 10 is equal to 0, update α and δ with the



(a) (b) (c) (d)

Figure 5: Triangular membership functions used in FLC. The fuzzification process uses 5a and 5b, and the defuzzification process uses 5c and 5d.

output of the FLC: $\alpha = y_1 * \alpha$; $\delta = y_2 * \delta$. Set $\lambda = \lambda(1-\alpha)$; $G = G(1-\alpha)$; update ν according to (5); update B_k according to (6); update *restricted* routes; and go to Step 2

The parameters λ, G and ν are updated at the end of each epoch regardless of whether the FLC is used. Using the FLC to update α and λ more frequently than 10 epochs resulted in the algorithm taking longer for convergence, and it did improve the solution quality. By allowing 10 epochs to pass, the input into the FLC, x_1 and x_2, seem to provide a more stable indicator as to how the SOM is evolving as compared to a higher frequency.

5. Experimental Results

In order to evaluate the effectiveness of fuzzy logic for parameter control in the updated SOM, the same ten standard CVRP instances [28, 29] were used as in Section 3.2. The Fuzzy SOM is initialized with the same parameter values as outlined in Section 3.2; however, instead of varying the parameters c, δ, and the length that a route is *restricted*, these values are initialized as follow: $c = 1, \delta = .01$ and a route remains *restricted* until it is not overloaded in the intermediate solution. For each problem, the best solution out of 100 replications of the Fuzzy SOM are reported. The results of the Fuzzy SOM are compared to the results of the updated SOM with no fuzzy logic. A comparison is made in both the number of feasible solutions as well as the solution quality. Table 5 provides a side by side comparison of the number of feasible solutions found by the updated SOM with fuzzy logic for parameter control, with a portion of the parameters tested for the updated SOM without fuzzy logic (Section 3.2).

In order to understand the performance of the Fuzzy SOM as a constructive heuristic, the proposed algorithm is compared to two well-known constructive heuristics: the parallel Clarke and Wright algorithm [27, 13] and the Sweep algorithm [26]. The Sweep Algorithm implementation used in this research follows the algorithm outlined in [33]. Once the customers were separated into routes according to the Sweep algorithm, the routes were then improved using a 2-opt heuristic. This Sweep algorithm was repeated n times, with each city used as the starting vertex once. The best result of these n replications is reported. The results are found in Table 6.

Table 5: Number of feasible solutions found out of 100 replications

Name	Updated SOM (no FLC) *restricted* until not overloaded						Fuzzy SOM
	$\delta = .01$	$\delta = .01$	$\delta = .01$	$\delta = .005$	$\delta = .005$	$\delta = .005$	$\delta = .01$
	$c = 1$	$c = 10$	$c = 100$	$c = 1$	$c = 10$	$c = 100$	$c = 1$
eil22	34	37	30	42	43	54	100
eil23	67	66	62	87	78	82	100
eil30	13	23	15	20	23	37	73
eil33	42	43	48	54	53	59	95
C1	11	12	15	15	15	17	54
C2	0	0	1	0	2	1	8
C3	46	47	57	86	91	89	92
C11	0	1	0	8	9	7	32
C4	15	12	14	49	49	51	71
C5	2	9	5	23	19	32	58

Table 6: Comparison of the best solutions found by each algorithm

Name	Size	Best Known (no. vehicles)	Clarke & Wright	Sweep w/ 2 opt	updated SOM	Fuzzy SOM	
						Result	PDB
eil22	21	375.3 (4)	388.8 (4)	398.3 (4)	375.3 (4)	375.3 (4)	0
eil23	22	569 (3)	621.1 (3)	581.0 (3)	569.7 (3)	569.7 (3)	0.1
eil30	29	534 (3)	534.5 (4)	508.1 (4)	539.9 (3)	539.9 (3)	1.1
eil33	32	835 (4)	843.1 (4)	879.6 (4)	846.0 (4)	845.0 (4)	1.2
C1	50	524.6 (5)	584.6 (6)	531.9 (5)	524.6 (5)	524.6 (5)	0
C2	76	835.3 (10)	900.3 (10)	902.8 (11)	860.3 (10)	862.2 (10)	3.2
C3	100	826.13 (8)	886.8 (8)	865.2 (8)	829.6 (8)	834.1 (8)	1.0
C11	120	1042.1 (7)	1068.1 (7)	1272.5 (7)	1049.7 (7)	1051.8 (7)	0.9
C4	150	1028.4 (12)	1133.4 (12)	1098.6 (12)	1071.1 (12)	1069.1 (12)	4.0
C5	199	1291.3 (16)	1395.7 (17)	1419.64 (17)	1365.3 (17)	1372.3 (17)	5.73

5.1. Discussion

The results of the updated SOM in Table 2 surpass previous applications of the SOM to the CVRP. These preliminary results also provide insight into the sensitivity of the SOM algorithm to the parameter settings. Although one set of parameter values did not lead to all of the best solutions, examining the algorithm's performance under different parameter settings provided insight into the delicate balance between the competing demands of finding a short route and finding a feasible solution.

The purpose of introducing fuzzy logic to control the parameter settings is to alleviate the need to tune the parameters for each new problem. The need for parameter control is a result of the sensitive balance between finding a feasible solution and finding a short route. Comparing the results of the Fuzzy SOM to the updated SOM without fuzzy, it is clear that the FLC for parameter control finds feasible solutions more often, and the quality of these solutions is commensurate with the results of manually tuning the SOM parameters.

Table 5 provides the number of feasible solutions found in 100 replications for a portion of the parameter settings evaluated in Section 3.2. In general, as the value of c increased and the value of δ decreased, the algorithm found feasible solutions more often, however, these solutions had longer route lengths. As c increases and δ decreases, the weight of the bias term ν is initialized to a larger value (c is the weighting factor), and as the algorithm evolves, ν decreases more slowly (δ controls how quickly ν decreases). Under these circumstances, there is a greater emphasis on the bias term in 4, which places more emphasis on the algorithm finding a feasible solution versus a short route.

With a smaller value of c and a larger value of δ, however, the emphasis is on the first term in 4, which will likely result in more infeasible solutions. This trend is evident in Table 5. When $\delta = .005$, across all values of c, there is an overall increase in the number of feasible solutions found.

The Fuzzy SOM requires no manual parameter tuning, and it is initialized with only one setting of c and δ. The FLC in the Fuzzy SOM is designed to output two values, y_1 and y_2, which are update factors for the terms α and δ. By updating α and δ, the balance between finding a short route and a feasible solution is shifted, if necessary, based on how the algorithm is evolving. It is clear from Table 5 that the Fuzzy SOM is successful in finding feasible solution more often compared to the manually tuned SOM.

Finding more feasible solutions is only beneficial if the solution quality remains competitive. It is clear from Table 6 that the solution quality is nearly unchanged by incorporating fuzzy logic for parameter control. The Fuzzy SOM not only finds feasible solutions more often, but the solution quality is nearly identical to the results of multiple attempts of manual parameter control.

Finally, the proposed Fuzzy SOM algorithm outperforms the parallel Clarke and Wright Algorithm as well as the Sweep Algorithm on nearly all of the 10 benchmark problems. The best known solution for 'eil30' is improved when the number of trucks is increased to 4, and both the Clarke and Wright as well as the Sweep Algorithm outperform the Fuzzy SOM by finding a better solution with an increased number of vehicles. Other than this problem, the results of the Fuzzy SOM are superior to the Sweep Algorithm on all remaining benchmark problems; and the Fuzzy SOM outperforms the Clarke and Wright Algorithm on 8 out of the 9 remaining problems.

6. Conclusion

Although the use of a SOM for solving a CVRP has not always been highly regarded in the literature, when directly compared to other construction heuristics, the SOMs performance cannot be overlooked. The performance of the proposed Fuzzy SOM is superior to the Sweep and Clarke and Wright algorithm results on nearly all of the benchmark problems, and the proposed algorithm improves existing SOM results on nine out of ten problems. This provides strong evidence that continued SOM research for the CVRP is promising and should persist.

This research demonstrates the credibility of the SOM as a constructive heuristic for the CVRP as well as the effectiveness of fuzzy logic for dynamic parameter control. Further, the simple bias term concept of the proposed algorithm allows for a straightforward extension to other VRP variants. As massively parallel computers continue to become more readily available, one of the most attractive aspects of the SOM algorithm is its inherent potential for parallelization. Now that the current results have clearly demonstrated the effectiveness of the Fuzzy SOM for finding competitive solutions, subsequent research will investigate the computing time of the parallelized Fuzzy SOM and compare it against other, well known VRP heuristics. Future research might also consider combining the existing SOM algorithm with local improvement heuristics or a hybridization with existing evolutionary techniques.

References

[1] J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proceedings of the national academy of sciences 79 (8) (1982) 2554–2558.

[2] J. J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons, Proceedings of the national academy of sciences 81 (10) (1984) 3088–3092.

[3] J. J. Hopfield, D. W. Tank, neural computation of decisions in optimization problems, Biological cybernetics 52 (3) (1985) 141–152.

[4] K. A. Smith, Neural networks for combinatorial optimization: A review of more than a decade of research, INFORMS Journal on Computing 11 (1) (1999) 15–34.

[5] R. Durbin, D. Willshaw, An analogue approach to the travelling salesman problem using an elastic net method, Nature 326 (6114) (1987) 689–691.

[6] J. Fort, Solving a combinatorial problem via self-organizing process: an application of the kohonen algorithm to the traveling salesman problem, Biological Cybernetics 59 (1) (1988) 33–40.

[7] B. Angeniol, G. de La Croix Vaubois, J.-Y. Le Texier, Self-organizing feature maps and the travelling salesman problem, Neural Networks 1 (4) (1988) 289–293.

[8] H. Ritter, K. Schulten, Kohonen's self-organizing maps: Exploring their computational capabilities, in: Neural Networks, 1988., IEEE International Conference on, IEEE, 1988, pp. 109–116.

[9] U.-P. Wen, K.-M. Lan, H.-S. Shih, A review of hopfield neural networks for solving mathematical programming problems, European Journal of Operational Research 198 (3) (2009) 675–687.

[10] M. N. Syed, P. M. Pardalos, Neural network models in combinatorial optimization, in: Handbook of Combinatorial Optimization, Springer, 2013, pp. 2027–2093.

[11] J.-Y. Potvin, State-of-the-art surveythe traveling salesman problem: A neural network perspective, ORSA Journal on Computing 5 (4) (1993) 328–348.

[12] E. Cochrane, J. Beasley, The co-adaptive neural network approach to the euclidean travelling salesman problem, Neural Networks 16 (10) (2003) 1499–1525.

[13] P. Toth, D. Vigo, The vehicle routing problem, SIAM Monographs on Discrete Mathematics and Applications, 2002.

[14] G. Laporte, What you should know about the vehicle routing problem, Naval Research Logistics (NRL) 54 (8) (2007) 811–819.

[15] 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.

[16] 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.

[17] H. Ghaziri, Supervision in the self-organizing feature map: Application to the vehicle routing problem, in: Meta-Heuristics, Springer, 1996, pp. 651–660.

[18] A. I. Vakhutinsky, 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.

[19] A. Torki, S. Somhon, T. Enkawa, A competitive neural network algorithm for solving vehicle routing problem, Computers & industrial engineering 33 (3) (1997) 473–476.
[20] A. Modares, S. Somhom, T. Enkawa, A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems, International Transactions in Operational Research 6 (6) (1999) 591–606.
[21] M. Schwardt, J. Dethloff, Solving a continuous location-routing problem by use of a self-organizing map, International Journal of Physical Distribution & Logistics Management 35 (6) (2005) 390–408.
[22] H. Ghaziri, I. H. Osman, Self-organizing feature maps for the vehicle routing problem with backhauls, Journal of Scheduling 9 (2) (2006) 97–114.
[23] G. Wilson, G. Pawley, On the stability of the travelling salesman problem algorithm of hopfield and tank, Biological Cybernetics 58 (1) (1988) 63–70.
[24] L. Burke, conscientious neural nets for tour construction in the traveling salesman problem: the vigilant net, Computers & operations research 23 (2) (1996) 121–129.
[25] J.-C. Créput, A. Koukam, The memetic self-organizing map approach to the vehicle routing problem, Soft Computing 12 (11) (2008) 1125–1141.
[26] B. E. Gillett, L. R. Miller, A heuristic algorithm for the vehicle-dispatch problem, Operations research 22 (2) (1974) 340–349.
[27] G. u. Clarke, J. W. Wright, Scheduling of vehicles from a central depot to a number of delivery points, Operations research 12 (4) (1964) 568–581.
[28] N. Christofides, S. Eilon, An algorithm for the vehicle-dispatching problem, OR (1969) 309–318.
[29] N. Christofides, A. Mingozzi, P. Toth, The vehicle routing problem, in: N. Mingozzi, P. Toth, C. Sandi (Eds.), Combinatorial Optimization, John Wiley and Sons, London, 1979.
[30] A. Michael, H. Takagi, Dynamic control of genetic algorithms using fuzzy logic techniques, in: Proceedings of the Fifth International Conference on Genetic Algorithms, Citeseer, 1993, pp. 76–83.
[31] F. Herrera, M. Lozano, Adaptation of genetic algorithm parameters based on fuzzy logic controllers, Genetic Algorithms and Soft Computing 8 (1996) 95–125.
[32] C. C. Lee, Fuzzy logic in control system: Fuzzy logic in controller part I, Systems, Man and Cybernetics, IEEE Transactions on 20 (2) (1990) 404–418.
[33] G. Laporte, M. Gendreau, J.-Y. Potvin, F. Semet, Classical and modern heuristics for the vehicle routing problem, International transactions in operational research 7 (4-5) (2000) 285–300.