

# Application of Neuro-Fuzzy System to Solve Traveling Salesman Problem

Suman Sangwan

*Received: 17 June 2011 Accepted: 9 July 2011 Published: 22 July 2011*

---

## Abstract

This paper presents the application of adaptive neuro-fuzzy inference system (ANFIS) in solving the traveling salesman problem. Takagi-Sugeno-Kang neuro-fuzzy architecture model is used for this purpose. TSP, although, simple to describe mathematically well characterized, is quite difficult to solve. TSP is called a NP-Hard problem, i.e. this problem is as hard as the hardest problem in NP-Complete space. Training of fuzzy system was performed by a hybrid Back-Propagation (BP) and Least-Mean-Square (LMS) algorithm and for optimizing the number of fuzzy rules, subtractive-clustering algorithm was utilized. Then the ANFIS was tested against a number of training data samples. More accurate and quick results were obtained by using ANFIS.

---

**Index terms**— MF, FIS, ANFIS, Subtractive Clustering, BP, LMS.

## 1 I. INTRODUCTION

The Traveling Salesman Problem (TSP) is one of the oldest combinatorial/optimization problems. TSP can be formulated as: A salesman visits 'n' number of city in such a way that the salesman starts journey by selecting one city among n cities, visits the other cities one by one and returns to the first city from where the journey was started. So the journey of the salesman provides a complete tour that consists of all cities. Now, the resultant of TSP is to find a tour with minimum cost.

TSP is a classic NP-Complete task requiring enormous amount of computational time, irrespective of the algorithm used in that the number of possible tours is extremely large even for problems containing a small number of cities. TSP is a constrained non-linear optimization problem. It is simple to describe, mathematically well characterized. But to find the actual best solution to TSP is computationally very hard, called a NP-complete problem. [7] The TSP is computationally intensive if an exhaustive search is to be performed comparing all possible routes to find the best one. For an n-city tour, there are n possible paths. Due to degeneracies, the number of distinct solutions is less than n. The term distinct in this case refers to tours with different total distances. For a given tour, it does not matter which of the n cities is the starting point, in terms of the total distance traveled. This degeneracy reduces the number of distinct tours by a factor of n. Similarly, for a given circuit, it does not matter in which of two directions the salesman travels. This fact further reduces the number of distinct tours by a factor of two. Thus, for an n-city tour, there are  $n!/(2*n)$  distinct circuits to consider. [19] A Neuro-Fuzzy system is a fuzzy inference system that uses a learning algorithm derived from neural network theory to determine the parameters: fuzzy sets, membership functions (MFs) and fuzzy rules by processing the data samples. The learning capabilities of neural networks make them a prime target for a combination with fuzzy systems in order to automate or support the process of developing a fuzzy system for a given task. To enhance the knowledge acquisition, neural networks are extended to extract automatically fuzzy rules from numerical data. The heuristic learning procedure operates on the local information; it takes the semantic properties of the underlying fuzzy system into account which results in constraints on the possible modifications applicable to the system parameters. A Neuro-Fuzzy system can be always (i.e. before, during and after learning) interpreted as a system of fuzzy rules. It is also possible to create the system out of training data from the scratch, as it is possible to initialize it by prior knowledge in form of fuzzy rules. The fuzzy rules encoded within the system represent vague samples and can be viewed as prototypes of the training data. [17] Neuro-fuzzy computing enables us to build

a more powerful intelligent decision-making system by combining the advantages of an artificial neural network with the fuzzy modelling of imprecise and qualitative knowledge. [18] Training of fuzzy system was performed by a hybrid back propagation (supervised learning) and least-mean-square algorithm, and for optimizing the number of fuzzy rules, a subtractive-clustering algorithm was adopted.

## 2 II. LITERATURE REVIEW

TSP has been studied well by using Elastic Adjusting method [3], Simulated Annealing [4], Ant Colony Optimization Algorithm [4, 2015], Hopfield Neural Network [5,7,16], Genetic Algorithm [5], Modified Particle Swarm Optimization algorithm [8], Expanding Self Organizing Map [8] etc. The brief study of how these techniques were applied to solve TSP is discussed below:

Global Journal of Computer Science and Technology Volume XI Issue XIV Version I 2011

## 3 August

In [3] the application of multi-pattern Elastic Adjusting method, to solve the TSP, is discussed. The method is enlightened by the synapse intensifying and adjusting mechanisms in biological neural network, which dynamically modulated the encoded gene patterns in chromosomes, guided GA to coordinate exploration and exploitation, therefore it tried to probe and evaluate the optimizing trends of static problem from its own dynamic change and the results of the experiments for TSP proved that this method has good optimizing performance. In [4] XuZhihong, SongBo and GuoYanyan discussed how Simulated Annealing-Ant Colony hybrid algorithm can be applied to solve the TSP. Based on the theoretical and practical study, the performance differences between parallel algorithms and traditional algorithms were analyzed, the feasibility of proceeding parallel algorithm to solve the TSP in cluster of PCs was analysed and some meaningful conclusions about parallel efficiency were achieved.

In [5] Junyan Liu, Zhuofu Wang, Honglian Yin and Wangling Qiu discussed the application of Genetic Algorithm and Hopfield Neural Network in solving the TSP. The optimization methods of GA in artificial neural network weight, structure and algorithm were introduced and GA-HNN was established. In GA-HNN, global property of GA and the parallelisms of artificial neural networks were combined, so the method retained the global stochastically searching ability of GA and the robustness and self-learning ability of HNN. The application of this method to the optimization of TSP has the advantages of calculation stabilization, prompt convergence and preferable precision. In [6] it is discussed how modified Particle Swarm Optimization algorithm can be applied to solve the TSP. This method involved the cooperative mechanism among the individuals such as the particles also learn from other individuals according to certain probability. Additionally, the mutation of velocity has been added according to the concept of the bird flying off at a tangent in the nature. This kind of tentative study behavior helped to find the global optimum solution more frequently. The numerical simulation results justified the effectiveness and efficiency of the proposed method.

In [7] it is discussed how Continuous Hopfield Network can be applied to solve the TSP. The energy function to be minimized was determined by both the constraints for a valid solution and total length of the touring path. Since setting of parameters in energy function was crucial to the convergence and performance of the network. The role of each parameter was analyzed and criteria for choosing these parameters were described. An iterative computation algorithm of CHN was used. In [8] it is discussed how the expanding self-organizing map (ESOM) can be used to handle the Euclidean TSP. By incorporating its neighborhood preserving property and the convex-hull property of the TSP, expanding SOM was introduced. In each learning iteration, the ESOM draws the excited neurons close to the input city, and in the meantime pushes them towards the convex-hull of cities cooperatively.

The ESOM may acquire the neighborhood preserving property and the convex-hull property of the TSP, and hence it can yield near-optimal solutions. Its feasibility is analyzed theoretically and empirically.

In [10] it is discussed how Neural Network Divide and Conquer strategies can be applied to solve the TSP. The concepts of Adaptive Resonance theory and Self Organizing Maps were applied. The results demonstrated that neural networks are capable of solving such an optimization problem in the quartermillion city, range with reasonable computational costs. In [11] it is discussed how an efficient genetic algorithm (GA) can be applied, to solve the TSP, with precedence constraints. The key concept of the proposed GA was topological sort. Also, a new crossover operation was developed for the proposed GA. The results of numerical experiments depicted that the proposed GA produced an optimal solution and exhibited superior performance compared to the traditional algorithms.

In [12] Marco Dorigo, Luca Maria Gambardella discussed the application of ant colony system (ACS), a distributed algorithm to solve the TSP. In the ACS, a set of cooperating agents called ants cooperate to find good solutions to TSP's. Ants cooperate using an indirect form of communication mediated by a phenomena, they deposit on the edges of the TSP graph while building the solutions. The results show that the ACS outperforms other nature-inspired algorithms such as simulated annealing and evolutionary computation, and it was concluded that ACS-3-opt, a version of the ACS, augmented with a local search procedure, to some of the best performing algorithms for symmetric and asymmetric TSP's. In [16] Taiji YAMADA, Kazuyuki AIHARA and Makoto KOTANI discussed the application of Deterministic Chaotic Neural Networks to solve the TSP. The TSP was

numerically analyzed using two types of model: (i) the devil's staircase model, which has relative refractoriness and the unit step output function and (ii) the chaotic neuron model, which has relative refractoriness and the continuous output function. Last, it was shown that CNN have high ability to solve TSP.

Traveling salesman problem is described as: given a finite number of cities along with the distance between each pair of them, finding the shortest close tour of visiting every city exactly once and returning to the starting point. Given a  $N \times N$  distance matrix 'D', in which  $D_{ij}$  represents the distance from  $i$ th city to  $j$ th city. 'D' is a symmetric matrix with all diagonal elements zeros, assuming that the traveling distance from  $i$ th city to  $j$ th city is equal to that from  $j$ th city to  $i$ th city. For figure 1, the distance matrix is shown as : b) Tour Matrix Each tour can be expressed in terms of a  $N \times N$  tour matrix 'V', of which  $x$ th row describes the position of the  $x$ th city in the current tour, the  $i$ th column describes which city is chosen to visit at the  $i$ th step. For figure 1, one possible trip A?C?D?B?A is shown as :A B C D A 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 B C D

#### 4 c) Constraints on Tour Matrix

Based on the above interpretation, two constraints must be valid for tour matrix 'V' that represents a feasible or valid solution, referring to the statement of TSP.

First, each city is visited once and only once in the tour. Hence, precisely one element must equal to '1' in each row and other elements in each row must equal to '0' of the matrix 'V'. Other elements in each row must equal to '0'. i.e.

$\sum_{j=1}^N V_{xj} = 1$  for  $x \in \{1, 2, \dots, N\}$  Second, one city must occur in any step in the tour. So, in each column of 'V', precisely one element must equal to '1' and every other element must equal to '0'. i.e.

$\sum_{i=1}^N V_{ii} = 1$  for  $i \in \{1, 2, \dots, N\}$

#### 5 d) Cost Function

The total length of a tour is the sum of distance between every adjacent pair of cities. Since the element of 'V' is either 1 or 0, the total length is represented by the cost function 'C' as:  $C = 0.5 \sum_{i=1}^N \sum_{x=1}^N \sum_{y=1}^N (V_{xy} + V_{yx}) D_{xy}$  Additionally,

$V_{y, N+1} = V_{y, 1}$  and  $V_{y, 0} = V_{y, N}$  The objective is to minimize the cost function under the constraints stated above.

### 6 IV. SOLVING TSP USING NEURO-FUZZY SYSTEM

The fuzzy system initially fuzzifies the inputs data to the values at interval [0 1] using the given set of membership functions (MFs). Next, it is inferred by the fuzzy logic through the IF-THEN rules. The basic part of the fuzzy system is the fuzzy inference engine that can be used for creating fuzzy rules. For TSP, the fuzzy rules can be represented as :  $R_i$  : If  $x_1$  is  $MF_{i1}$  and  $x_2$  is  $MF_{i2}$  and  $x_3$  is  $MF_{i3}$  and  $x_4$  is  $MF_{i4}$  ...  $x_j$  is  $MF_{ij}$  then  $z_i$  is  $z_i = s_i 0 + s_i 1 x_1 + \dots + s_i j x_j$  where  $R_i$  ( $i = 1, 2, \dots, L$ ) denotes the  $i$ th fuzzy rule,  $x_j$  ( $j = 1, 2, \dots, n$ ) is the  $j$ th input,  $z_i$  is the output of the  $i$ th fuzzy rule,  $s_i j$  coefficients are the constants that are determined after training the fuzzy system, and, finally,  $MF_{ij}$  is the  $j$ th fuzzy membership function of an antecedent for the  $i$ th rule, in sugeno form. [1] The output of this system can be described by the following function:  $y = \left[ \sum_{i=1}^L W_i z_i \right] / \left[ \sum_{i=1}^L W_i \right]$  and  $W_i = \prod_{j=1}^n MF_{ij}(x_j)$

Here membership function used is Gaussian MF which depends on two parameters  $\mu$  and  $\sigma$  given by:  $MF(x) = e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

Where  $\mu$  is the mean and  $\sigma$  is the variance of the function.

The basic task of the fuzzy system is to adjust the MF parameters where the output of each fuzzy rule and the estimating number of the rules are minimized with adequate precision. The ANFIS adapts the parameters of the TSK-type inference system using a neural network. [14] In the TSK-type systems, the output is a crisp number computed by multiplying each input by a constant followed by summing up the results. To train the fuzzy system, the ANFIS employs the BP algorithm. This approach is utilized for the parameters associated with the input MFs and further uses the Least Mean Square (LMS) estimation for parameters associated with the output MFs.

The gradient descent algorithm based on the LMS error between the actual and desired outputs is utilized in order to adjust for the parameters. The proposed combined methods may intensify the computational complexity, but the resulting performance is noticeably improved. Furthermore, weight updating is another important step for adjusting the model parameters which is based on the BP algorithm. [1] To increase the ability of the FIS in solving TSP, subtractive clustering [14] is utilized. This method partitions the data into clusters and generates the FIS with the appropriate number of rules required to distinguish the fuzzy qualities associated with each of the clusters. Subtractive clustering is based on a measure of the density of data points in the feature space. The idea is to find the regions in the feature space with high densities of data points. The point with the highest number of neighbors is selected as the center of a cluster. The data points within a prescribed fuzzy radius are then subtracted, and the algorithm looks for a new point with the highest number of neighbors.

When applying subtractive clustering [14] to a set of input-output data, each of the cluster centers represents a rule. To generate these rules, the cluster centers are used as the centers for the premise sets in a singleton type of rule base. We claim that our proposed method provides the appropriate number of fuzzy rules because,

after each selection of a cluster center by this method, the most similar data points are placed in a cluster with specified radii. Hence, such selection approach provides the system with a relatively better choice of clusters.

## 7 V. DESIGNING NEURO-FUZZY ARCHITECTURE

In this paper, we used the subtractive-clustering method to determine the number of fuzzy rules and the BP and LMS for the selection of the MF parameters and output of each rule. The designed network consists of several layers. The number of neurons in each layer depends on the dimension of the input vectors or the training feature set. For the system, we used four neurons in the input layer, six neurons in the hidden layer as a result of the proposed fuzzy rules, and, finally, one neuron in the output layer. Furthermore, through the implementation of subtractive clustering for each of the designed systems, the rule characteristics were determined.

As in the Sugeno type of order four, the outputs of each rule, namely  $z_i$  (for  $i = 1, 2, 3, 4, 5, 6$ ) were obtained by a combination of all mentioned inputs, where:  $z_i = s_{i0} + s_{i1}x_1 + s_{i2}x_2 + s_{i3}x_3 + s_{i4}x_4$ . Where  $x_j$  (for  $j = 1, 2, 3, 4$ ) is the  $j$ th input. The coefficients are adjusted through the LMS algorithm.  $s_{i0} = 0$  and  $s_{i1} = s_{i2} = s_{i3} = s_{i4} = 1$  for  $(i = 1, 2, \dots, L)$   $W_i = 1$  for  $(i = 1, 2, \dots, L)$

To solve a 4-city (figure), there are 4 inputs, namely 'trip1', 'trip2', 'trip3', 'trip4' respectively; the complete tour will be summation of all these subtrips. If the starting vertex is taken to be 'A', as in figure 5, there are 6 possible tours are available, hence 6 rules can be generated. Suppose that we are given an input-output pair  $(x, y)$ , where  $x = [x_1, x_2, \dots, x_j]$ . Our goal is to minimize the cost function,  $e = 0.5 \times \{y_{des} - y\}^2$  Where  $y_{des}$  is the desired output. The output of each rule  $z_i$  is defined by:  $z_i(t+1) = z_i(t) - k_z (\partial e / \partial z_i)$

Where  $k_z$  is a learning step. To adjust the parameters of MF, namely  $\mu$  and  $c$ , we start with TSK system. The derivative is given by:  $(\partial e / \partial z_i) = (\partial e / \partial y) \times (\partial y / \partial z_i)$  Where  $(\partial y / \partial z_i) = w_i / (\sum_{i=1}^L w_i)$  and  $(\partial e / \partial y) = y_{des} - y$  Similarly, for the  $j$ th MF of the  $i$ th fuzzy rule, the parameters  $c_{ij}$  and  $\mu_{ij}$  are calculated. The parameters for the output MFs are set to 0.2, 0.8, 0.9, 0.8, 0.9, 0.2 respectively. The range for output is set to  $[0, 1]$  and constant membership function is used for output.

The ANFIS model structure consisting of 4 inputs, 6 input MFs, 6 rules, 6 output MFs and finally 1 output is shown in figure 5. For the system that we designed, six fuzzy rules were obtained. Table depicts input values rules and their respective outputs. In this table, the use of MFs is represented with 6 linguistic expressions namely (A,B), (A,C), (B,C), (B,D), (C,D), (D,A). To design the system, we used "min" as the AND method and "max" as the OR method; "wtaver" as the defuzzification method. 3. The figure 7 shows that how the respective surface diagrams can be represented. Finally FIS structure was trained to result in a specific output and minimizing the error. For 10 training data samples, the minimum average testing error was found to be 1.05958e-007 which is significantly small. The optimal tour was found which found the output with this error. It is shown that neuro-fuzzy approach gives more accurate results with lesser error as compared to the other techniques. Furthermore, this approach can be used in solving TSP with large number of cities.



Figure 1: AugustFigure 1

<sup>1</sup>© 2011 Global Journals Inc. (US)

<sup>2</sup>© 2011 Global Journals Inc. (US)

<sup>3</sup>© 20 in this application. The FIS structures were trained against a number of vague training data samples.

<sup>4</sup>August

2

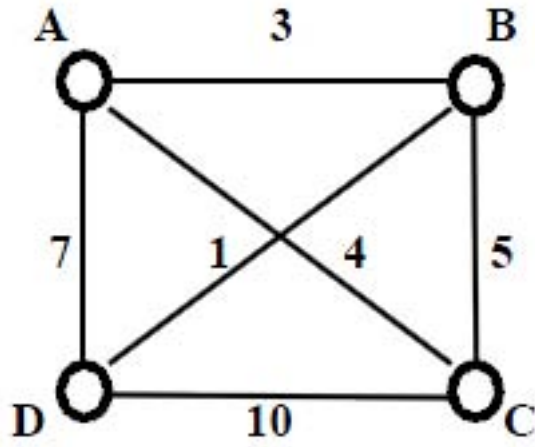


Figure 2: Figure 2 :

3

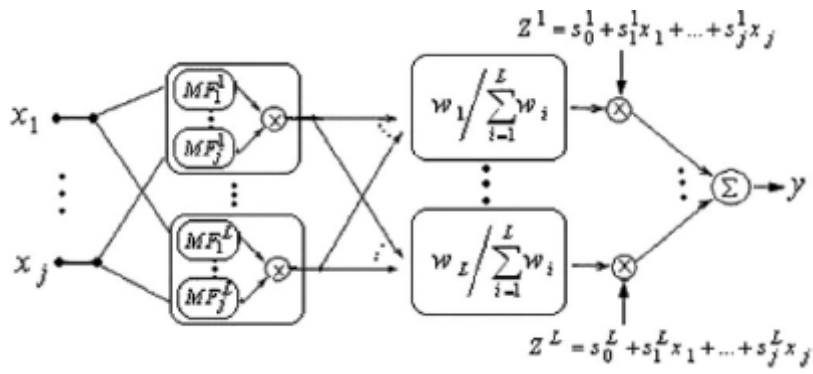


Figure 3: Figure 3 :

4

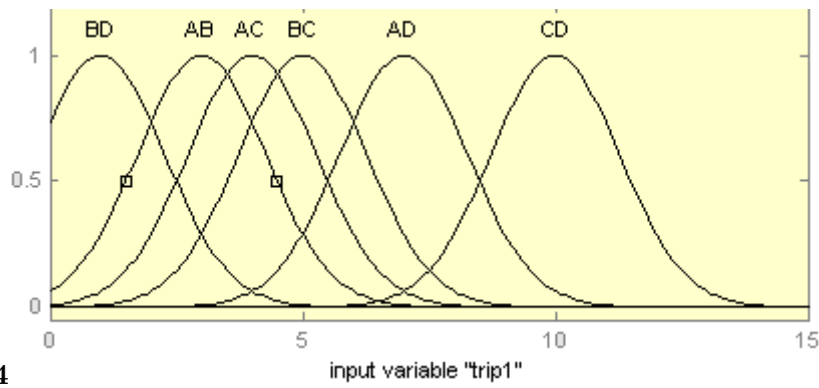


Figure 4: Figure 4 :

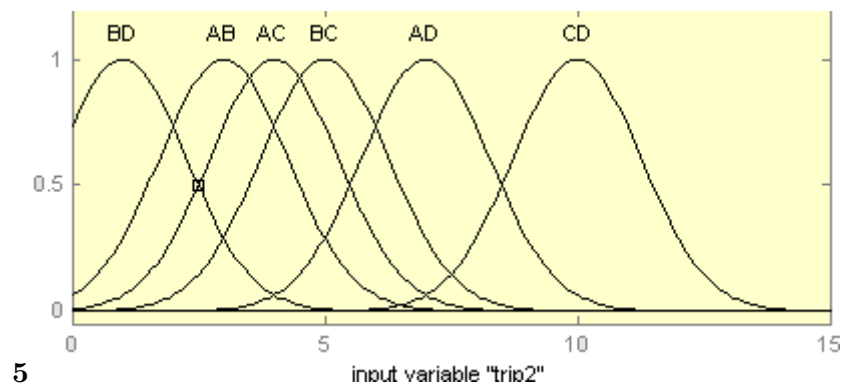


Figure 5: Figure 5 :

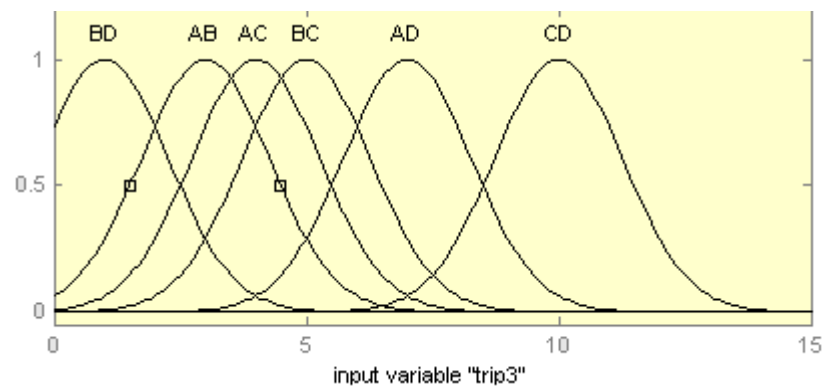


Figure 6:

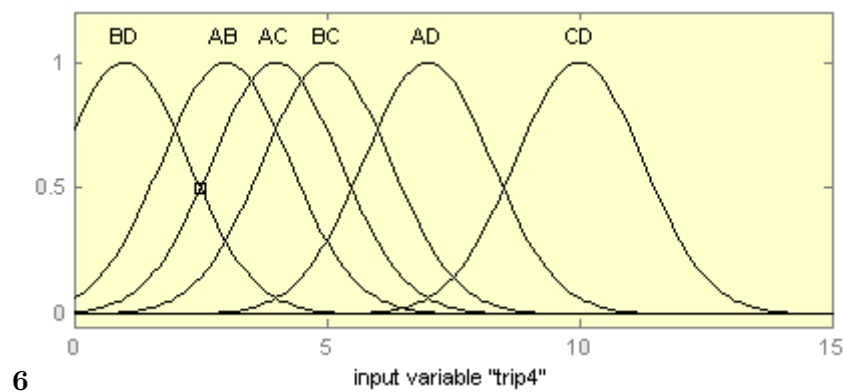


Figure 7: Figure 6 :

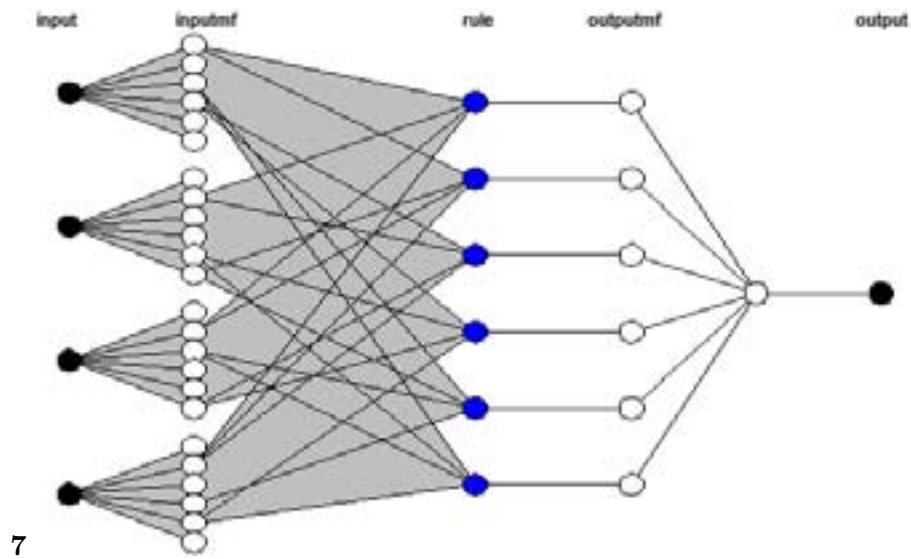


Figure 8: Figure 7 :

1

| Number of distinct tours<br>Cities | possible<br>sub-trips | distinct trips |
|------------------------------------|-----------------------|----------------|
| n                                  |                       | $n!/(2^n)$     |
| 4                                  | 6                     | 3              |
| 5                                  | 10                    | 12             |
| 6                                  | 15                    | 60             |
| 7                                  | 21                    | 360            |
| 8                                  | 28                    | 2,520          |
| 9                                  | 36                    | 20,160         |
| 10                                 | 45                    | 1,81,440       |

It is shown mathematically by table 1 and graphically by figure 2, how the number of possible sub-trips and distinct tours change with the number of cities.

Figure 9: Table 1 :

3

For testing FIS, the following matrices (4x5 & 5x6 respectively) are loaded as training data:

trainD4 =

|   |   |   |   |   |
|---|---|---|---|---|
| 0 | 3 | 5 | 4 | 3 |
| 3 | 0 | 6 | 9 | 6 |
| 5 | 6 | 0 | 7 | 7 |
| 4 | 9 | 7 | 0 | 4 |

trainD5=

|       |    |    |    |    |    |
|-------|----|----|----|----|----|
| 0     | 7  | 36 | 11 | 9  | 7  |
| 7     | 0  | 13 | 19 | 23 | 13 |
| 36 13 |    | 0  | 8  | 42 | 8  |
| 11 19 |    | 8  | 0  | 10 | 10 |
| 9     | 23 | 42 | 10 | 0  | 9  |

The training results are shown as in

Figure 10: Table 3 :

4

Figure 11: table 4 .

4

| Matrix  | Training Error | Average Testing Error |
|---------|----------------|-----------------------|
| trainD4 | 1.2273e-007    | 1.10293e-007          |
| trainD5 | 1.96643e-006   | 2.35423e-006          |

*[Note: ANFIS information: (4x5 training data) Number of nodes: 47 Number of linear parameters: 20 Number of nonlinear parameters: 32 Total number of parameters: 52 Number of training data pairs: 4 Number of checking data pairs: 4 Number of fuzzy rules: 4 ANFIS information: (5x6 training data) 12]*

Figure 12: Table 4 :

There is only 1 output namely 'trip\_distance' and 6 output MFs, namely 'output1', 'output2', 'output3', 'output4', 'output5', 'output6' respectively.

## .1 August

This page is intentionally left blank

[Wang et al. ()] *A Modified Particle Swarm Optimization Algorithm and its Application For Solving TSP*, Cuiru Wang , Jiangwei Zhang , Jing Yang , Chaoju Hu . Jun Liu. 2005. IEEE.

[Khezri et al. ()] *A Neuro-Fuzzy Inference System for sEMG-Based Identification of Hand Motion Commands*, Mahdi Khezri , Mehran Jahed , Member . 2010. IEEE.

[Verlag ()] *Advances in Soft Computing, Introduction to Neuro-Fuzzy Systems*, Physica Verlag . 2000. Springer Verlag Company.

[Kwong-Sak et al. ()] 'An expanding self-organizing neural network for the traveling salesman problem'. Kwong-Sak , Hui-Dong Leunga , Zong-Ben Jinb , Xuc . *Neurocomputing* 2004. 62 p. .

[Gambardella and Dorigo ()] 'Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem'. L M Gambardella , M Dorigo . *Proceedings of the Twelfth International Conference on Machine Learning*, (the Twelfth International Conference on Machine Learning) 1995. Morgan Kaufmann.

[Keng Ang et al.] *Application of rough set-based neuro-fuzzy system in NIRS-based BCI for assessing numerical cognition in classroom*, Kai Keng Ang , Cuntai Guan , Kerry Lee , Jie Qi Lee , Shoko Nioka , Britton Chance . ©2010 IEEE.

[Taiji et al. ()] *Chaotic Neural Networks and The TSP*, Yamada Taiji , Aihara Kazuyuki , Kotani Makoto . 1993. IEEE.

[Moon et al. ()] 'Discrete Optimization-An efficient genetic algorithm for the traveling salesman problem with precedence constraints'. Chiung Moon , Jongsoo Kim , Gyunghyun Choi , Yoonho Seo . *European Journal of Operational Research* 2002. 140 p. .

[Song et al. ()] 'Elastic Adjusting Method and its application to solve static TSP'. Yong Song , Xianfu Chen , Yongyuan Qing , Jingchuan You . *Fifth International Conference on Natural Computation*, 2009. 2009. IEEE.

[Freeman and Skapura] James A Freeman , David M Skapura . *Neural Networks -Algorithms, Applications, and Programming Techniques*,

[Liu et al. ()] *GA-Hopfield Network for Transportation Problem*, Junyan Liu , Zhuofu Wang , Honglian Yin , Wangling Qiu . 2008. IEEE.

[Mulder et al. ()] *Large Scale TSP via Neural Network Divide and Conquer*, Sam Mulder , C Dlonald , I I Wunsch . 2002. IEEE.

[Chiu ()] 'Method and software for extracting fuzzy classification rules by subtractive clustering'. S Chiu . *Proc. Biennial Conf. North Amer. Fuzzy Inf*, (Biennial Conf. North Amer. Fuzzy Inf) 1996. p. .

[Jang et al. ()] *Neuro-Fuzzy and Soft Computing', A Computational Approach to Learning and Machine Intelligence'* PHI, J S R Jang , C T Sun , E Mizutani . 2003.

[Lamba ()] *Neuro-Fuzzy Systems*, V K Lamba . 2008. University Science Press.

[Xuzhihong and Songbo ()] *Using Simulated Annealing and Ant Colony Hybrid Algorithm to Solve TSP*, Xuzhihong , Guoyanyan Songbo . 2009. IEEE.

[Wu and Yang ()] Helei Wu , Yirong Yang . *Application of Continuous Hopfield network to solve the TSP8th International Conference on Control, Automation, Robotics and Vision*, 2004. IEEE.