01 Jan 2001

ART/SOFM: A Hybrid Approach to the TSP

Narayan Vishwanathan

Donald C. Wunsch
Missouri University of Science and Technology, dwunsch@mst.edu

ART / SOFM : A hybrid approach to the TSP

Narayan Vishwanathan and Donald C. Wunsch II*

*Applied Computational Intelligence Laboratory
Dept. of Electrical and Computer Engineering
University of Missouri - Rolla
MO 65409-0249 USA
narayan@umr.edu,dwunsch@ece.umr.edu

Abstract:

We present a new method of solving large scale Traveling Salesman Problem (TSP) instances using a combination of Adaptive Resonance Theory (ART) and Self Organizing Feature Maps (SOFM). We divide our algorithm into three phases. Phase one uses ART to form clusters of cities. Phase two uses a novel modification of the traditional SOFM algorithm to solve a slight variant of the TSP in each cluster of cities. Phase three uses another version of the SOFM to link all the clusters. The experimental results show that our algorithm finds approximate solutions which are about 13% longer than those reported by the chained Lin Kernighan method for problem sizes of 14,000 cities.

1 Introduction

Neural Networks have been applied to solve numerous combinatorial optimization problems (COP). The TSP is probably the best known COP and is NP complete. Neural Networks were first applied to solve the TSP by Hopfield and Tank who used a neural network with N^2 neurons, where N is the number of cities in the TSP. However, this technique had problems even for small instances, and attempts to scale the method to larger problem sizes failed.

A different approach is based on the SOFM method of solving the TSP, and many researchers have applied this technique with modifications. The idea in all the methods is to perform a mapping of cities to neurons, which are located on a ring, where the ordering on the ring represents the traversal of the cities. They can be broadly divided into two classes, depending on whether they use a fixed or variable number of neurons.

In [1,2,12,13] the number of neurons at any stage of the algorithm is fixed and equal to the number of cities. In [3,4,5] the number of neurons at any stage is not fixed and neurons may be added and deleted as the algorithm proceeds. The advantage of using a fixed number of neurons is faster convergence and lesser bookkeeping, while techniques based on varying number of neurons seem to give a more accurate result for smaller instances. We combine the ART and the SOFM allowing the number of neurons at any stage to be fixed.

Clustering has been frequently used to divide the TSP into smaller sub-problems, and combine the sub-solutions separately [6,7,8,9,10]. References [6,8,10] have combined clustering and a Neural Network approach to solve the TSP. Noel and Szu [10] use genetic algorithms to solve the inter and intra cluster tours while [6] uses the Hopfield network for the same. In our approach we used the SOFM for the inter as well as intra cluster tours since the SOFM gives the best results among all neural network solutions.

2 ART / SOFM

Our complete ART/ SOFM algorithm can be divided into the following modules.

- Clustering
- Finding Hamiltonian paths for each cluster (Intra cluster tours)
- Finding Hamiltonian loop for inter cluster tour.
- Linking the clusters and applying heuristics to improve the solution.

2.1 Clustering

Adaptive resonance theory was developed by Carpenter and Grossberg, and a large number of ART architectures have been introduced in the last decade. A major separation among all the architectures is based on whether supervised or unsupervised learning is used. Unsupervised learning is implemented when a collection of input patterns is to be appropriately clustered into categories, while supervised learning is utilized when a mapping needs to be learned between inputs and corresponding output patterns. For our clustering of the input cities, we used the unsupervised ART1 algorithm given in [11], which discusses each of the ART variants and also supplies Matlab code for implementing each of them.

0-7803-7044-9/01/$10.00 ©2001 IEEE

To form the binary input patterns for ART1, first the x and y co ordinates are quantized. The number of quantization levels, **n**, is an input which can be entered by the user (we fixed it at 8). Then we append to each thermometer coded co-ordinate, its complement. Thus the input 'I' to the ART1 program is a vector of the following form

$$I \text{ (Size 4n by N)} = \left[I_1 \; I_2 \; \ldots\ldots\ldots\ldots\ldots I_N \right]$$

$$\begin{array}{c} I_1 \; I_2 \; \ldots\ldots\ldots\ldots\ldots\ldots\ldots I_N \\ \left[\begin{array}{cccc} i_{11} & i_{21} & \ldots\ldots\ldots\ldots\ldots & i_{N1} \\ i_{12} & i_{22} & \ldots\ldots\ldots\ldots\ldots & i_{N2} \\ i_{13} & i_{23} & \ldots\ldots\ldots\ldots\ldots & i_{N3} \\ i_{14} & i_{24} & \ldots\ldots\ldots\ldots\ldots & i_{N4} \end{array}\right] \end{array}$$

where I_N is the column vector for the N^{th} city consisting of the sub vectors

i_{N1} = coded X co-ordinate (n by 1)
i_{N2} = coded Y co-ordinate (n by 1)
i_{N3} = Complement coded X co-ordinate (n by 1)
i_{N4} = Complement coded Y co-ordinate (n by 1)

The ART1 algorithm forms clusters of cities using the above binary input vector. The number of clusters depends on the vigilance parameter '**rho**'. We fixed rho at 0.8 since it gave us best results, but it can be changed by the user.

2.2 Intra Cluster paths

Kohonen's self organizing feature map has the topological characteristics that can be effectively used in solving the TSP. The key idea is to perform a mapping from cities to neurons which preserves the neighborhood relationships among the cities, ie cities which are geometrically close should be mapped onto neurons which are close to each other on the ring. For finding the intra cluster tours we have used a modification of the method outlined in [1].

The Intra Cluster Algorithm

We use a fixed number of neurons at any stage; equal to the number of input cities N. The winner neuron for each city is not duplicated, but excluded in the next competition. The main advantage of using a fixed number of neurons is that the convergence speed is better, especially for large scale instances. We have made a slight change to the above method since for the intra clusters we look for Hamiltonian paths and not Hamiltonian cycles.

We start with a ring of N neurons, each being characterized by its weights [W_{xi} , W_{yi}]. An epoch consists of presentation of all the N cities. City 'i' with co – ordinates [X_i , Y_i] is presented as input to the SOFM and the neurons compete to be the winner. The winner neuron 'j' is determined as the neuron on the ring closest to the city 'i'. For each neuron on the ring we compute its Euclidean distance 'D' from the input city and select the winner 'j' as

$$D_j = \min(D) \qquad (1)$$

We then move the winner neuron and its neighbors toward city 'i' .The distance that each neuron moves toward the city 'i' is determined by the Gaussian function H(σ, d) where

$$H(\sigma, d) = \exp (- d^2 / \sigma^2) \qquad (2)$$

The parameter 'σ' measures the effective width of the topological neighborhood ; it measures the degree to which the excited neurons in the vicinity of the winner neuron participate in the learning process. For solving the TSP [1] uses 'd' to represent the circular distance from winner neuron. The circular distance between neuron 'k' and winner neuron 'j' is

$$d = \min [|(j - k)| , |(j + N - k)| , |k + (N - j)|] \qquad (3)$$

This is because for the TSP we are looking to form Hamiltonian cycles and so we want to form a closed loop where neuron 1 and neuron N are neighbors. However, for our problem, we are looking to for best intra cluster tours, which are Hamiltonian paths. So we define the distance between neuron 'k' and the winner neuron 'j' as

$$d = \text{abs} (j - k) \qquad (4)$$

The update rule we use for the changing the weights of the neurons is

$$W_{xk} = W_{xk} + H(\sigma, d) (X_i - W_{xk}) \qquad (5)$$
$$W_{yk} = W_{yk} + H(\sigma, d) (Y_i - W_{yk}) \qquad (6)$$

Thus the winner neuron moves toward city i, and also induces its neighbors on the ring to do so, with a decreasing intensity along the ring. Also the winner is inhibited, by making its weights negative, so that the winner can no longer participate in any competition for this epoch. This ensures that in any epoch there is a unique neuron 'j' associated with a city 'i' though this association could change between epochs.

The size of the topological neighborhood function shrinks with time, which is achieved by making 'σ' decrease with every successive epoch. Thus we start with a large neighborhood which includes almost all the neurons centered around the winner and eventually reduce to a very

small value of only a couple of neighboring neurons around the winner.

2.3 Inter Cluster cycles

Solving the inter cluster problem for all the clusters gives us two cities for each cluster, the starting and ending cities C_{1k} and C_{2k}. To solve the problem of linking together clusters we use another modification of the algorithm outlined above. We perform a constrained mapping, wherein neurons representing cities of the same cluster are adjacent on the ring. For example if neuron 'i' is the winner for city 'C_{1k}' which belongs to cluster 'k' , then the winner neuron for the other city C_{2k} of the cluster 'k' must be one of the immediate neighbors of neuron 'i'.

So when we present city 'C_{1k}' belonging to cluster 'k' as input , we determine the winner for city 'C_{1k}' as well as for the other city in cluster 'k'; city C_{2k}. Thus if neuron j maps to city C_{1k} then force neuron j +1 or j-1 on the ring to map to the other city in the cluster C_{2k}.

2.4 Linking the clusters

The final stage involves linking together the inter cluster and intra cluster tours to give the complete solution. However, just linking the tours according to the inter cluster solution gives rise to numerous crossings. So to improve on our initial solution we run heuristics to remove crossings and swap links.

- *Remove crossings*

There are two kinds of crossings which occur in the final tours intra cluster crossing and inter cluster crossing.

1) Intra Cluster / Minor crossing

This type of crossing occurs within the intra cluster solution causing sub optimal tour. To remove this crossing we perform a brute force exhaustive search for the optimum path taking 8 cities at a time.

2) Inter Cluster crossing / Major crossings

When the clusters are linked together , crossings are bound to occur. To remove these crossings we look at the line joining the ending city of cluster 'i' and the starting city of cluster 'i+1'. We search in a small rectangular area around this line and look for adjacent neurons lying on the opposite sides of the line which indicates that a crossing has occurred. The key is to minimize the search area so as to reduce the time complexity .

- *Intelligent link swapping*

We look at two neurons which are very close in weights but are located at a large distance apart in the ring and try and swap some links around these two neurons. This heuristic typically removes some long edges.

3 Simulations and Results

We tested our approach on TSP instances with different kinds of distributions like random, clustered random and semi random distributions. To generate the bench mark instances we used the code at http://www.research.att.com/~dsj/chtsp/download.html. This website aims to create a reproducible picture of the state of the art in the area of TSP heuristics (their effectiveness, their robustness etc.), so that future algorithm designers can quickly tell on their own how their approaches compare with already existing TSP heuristics. To this end the organizers have identified a standard set of benchmark instances and generators (so that quality of tours can be compared on the same instances). The benchmark instances have random and clustered random distributions. We also tested our algorithm on instances from the TSPLIB site.

We ran our code, which was written in Matlab, on a Sun Ultra Sparc II 450 Mhz quad processor system with 1G RAM. We compared our results to optimal solutions when known. Otherwise, the basis for comparison is the chained Lin – Kernighan approach, which is one of the most popular methods of solving large scale TSPs. A version of the code for implementing the same is available at http://www.caam.rice.edu/~bico/lk.html

The table below summarizes our results. Problem instances with uniform random distributions in a 10^6 X 10^6 square are indicated as R –problem size. For example R1000 indicates a 1000 city uniformly distributed instance. Similarly clustered random instances (integer-coordinate points located in clusters that are uniformly distributed in the 10^6 X 10^6 square) are indicated as CR – problem size. For example CR1000 refers to a clustered 1000 city problem. For these instances we compared our solution to the Lin Kernighan solution (indicated by an *). The other instances are from the TSPLIB web site for which we compared our results to the optimal reported ones.

Table 1 : Summary of results

Problem	Percentage Excess	
	Our solution	Best neural net solution
R1000	9.6*	
CR1000	8.1*	
R2000	11.1*	
CR2000	9.9*	
R4000	12.7 *	
CR4000	8.1*	
R6000	12.8*	
CR6000	8.3*	
R8000	12.2*	
CR8000	10.1*	
R10000	12.9*	
R14000	12.3*	
Pr1002	9.4**	7.6** [12]
Dsj 1000	8.3**	
U 2319	3.3**	6.7**[12]
TK 2392	10.1**	5** [13]
Fnl 4461	8.8**	
Rl 11849	17.1**	17.4**[12]

R *Uniform random instance*
CR *clustered random instance*

* *Percentage excess over the Lin Kernighan tour*
** *Percentage excess over optimal tour length*

4 Conclusions

Neural approaches to solving the Traveling Salesman Problem (TSP) have used either the Hopfield network or modified forms of the Self Organizing Feature Maps (SOFM). While the Hopfield networks have not scaled well, algorithms based on SOFM have shown much promise. However a large gap exists between neural net solutions and the best known classical /heuristic algorithms for TSP, partly because neural net approaches need to incorporate techniques from classical methods. Here we borrow the principle of divide and conquer from approximation methods and combine it with neural nets to make them more effective, using ART to form clusters of cities and the SOFM for linking the cities within clusters, and the clusters. Our approach is able to obtain reasonable solutions for about 14000 city problems and shows a good promise for scaling. Nevertheless it is still far short of the best known heuristic methods which are able to solve instances of 25,000,000 cities. Still, our results have the best scaling performance of neural net TSP results reported to date, to the best of our knowledge.

References:

[1] Kim, S.-J.; Kim, J.-H.; Choi, *H.-M.* "An efficient algorithm for traveling salesman problems based on self- organizing feature maps." *Fuzzy Systems, Second IEEE International Conference on 1993. pp 1085* –1090.

[2] Doo Sung Hwang; Mun Sung Han, "Two phase SOFM", *Neural Networks, 1994. IEEE World Congress on Computational Intelligence* .pp:742–745 vol.2.

[3] *B.Angeniol , G.D.Vaubois and J.Y.L.Texier,* "Self – Organizing Feature Maps and the Traveling Salesman Problems." *Neural Networks Vol1*, 1988 ,pp 289 – 293.

[4] Fritzke, B.; Wilke, P, "FLEXMAP-a neural network for the traveling salesman problem with linear time and space complexity" . *Neural Networks, 1991. IEEE International Joint Conference on , 1991.* pp: 929 –934.

[5] N.Aras,B.J.Oommen,I.K.Altinel, " The Kohonen Network incorporating explicit statistics and its application to the Traveling salesman problem.", *Neural Networks 12* (1999) 1273 –1284.

[6] Foo, Y.P.S.; Szu, H. "Solving large-scale optimization problems by divide-and-conquer neural networks." *Neural Networks*, 1989. IJCNN., International Joint Conference on pp: 507 –511.

[7] B.Codenotti and L.Margara , "Efficient Clustering Techniques for the Geometric Traveling Salesman Problem.",TR–92–036 , June 1992 International Computer Science Institute , Berkeley CA

[8] Park, D.C.; Figueras, A.L.; Chen, C. "A hierarchical approach for solving large-scale Traveling salesman problem." *Neural Networks,* 1994. IEEE World Congress on Computational Intelligence. pp: 4613 –4618.

[9] Achim Bachem , Barthel Steckemetz and Michael Wottowa "An efficient parallel cluster –heurisitic for large Traveling Salesman problems." Technical report 94 –150 , Universitat zu Koln 1994.

[10] Noel, S.; Szu, H. "Multiple-resolution divide and conquer neural networks for large-scale TSP-like energy minimization problems." *Neural Networks,*1997., International Conference on , Volume: 2 , pp: 1278 -1283 vol.2.

[11] Teresa Serrano- Gotarredona , Bernabe Linares – Barranco and Andreas G.Andreou. " Adaptive Resonance Theory Microchips." Kluwer Academic Publishers.

[12] E. H. L. Aars and H. P. Stehouwer. "Neural networks and the travelling salesman problem". *Proc. ICANN'93, Int. Conf. on Artificial Neural Networks*, pages 950--955

[13] E.M.Cochrane and J.M.Cochrane "Exploring Competition and co –operation for solving the Euclidean Travelling Salesman Problem by using the Self – Organizing Map ." *Artificial Neural Networks , 7-10 September 1999. 180 – 185.*