An Effective Solution to Large-scale Traveling Salesman Problems Using Chaotic Neural Networks

T. Nagashima[†], K. Kobayashi[‡*], and M. Obayashi[‡]

[†] Division of Computer Science and Systems Engineering
Graduate School of Science and Engineering, Yamaguchi University

[‡] Department of Computer Science and Systems Engineering
Faculty of Engineering, Yamaguchi University

[†‡] address: 2–16–1 Tokiwadai, Ube, Yamaguchi 755–8611, Japan
[*] phone: +81-836-85-9519 [*] fax: +81-836-85-9518
[*] e-mail: k@nn.csse.yamaguchi-u.ac.jp

Abstract

This paper presents an effective TSP (Traveling Salesman Problem) solver for large-scale problems using neural networks. Firstly, in the proposed method, an intractable large-scale TSP is divided into some tractable small-scale problems (clusters) using a clustering technique. Secondly, a visiting order of clusters is determined using chaotic neural networks (CNNs). Thirdly, the small-scale problems are solved using CNNs. Simultaneously, two pairs of connecting cities between adjacent clusters are also determined using CNNs. Through computer simulations, it is confirmed that the proposed method is effective to solve the large-scale problems.

1 Introduction

It is well known that TSP (traveling salesman problem) [1] is one of the combinatorial optimization problems and is categorized as NP-hard problem. The TSP is a problem to find a closed tour which visits each city once, returns to the starting city and has a shortest total path length. A lot of efforts have been devoted to solve TSP because its solutions are widely applicable to engineering and other field.

Hopfield and Tank proposed a recurrent neural network (RNN), so-called Hopfield model, for solving combinatorial optimization problems [2]. In their asynchronous model, it is guaranteed that a network gradually changes its state so as to decrease a cost function defined for the problem. They have applied it to TSP and shown that an approximate solution is obtained to some small-scale TSPs. As the number of cities increases, however, the network tends to be trapped into local minima because they are dramatically increased. In general, such a problem could be escaped using annealing techniques like Boltzmann machine [3]. But, regarding computational cost, it becomes much heavier.

Nozawa introduced a negative self-feedback connection to a discrete-type Hopfield model [4]. Since this allows each neuron to have chaotic states, a network converges to a global minimum not trapping into local minima. Furthermore, Chen and Aihara improved convergence property in Nozawa model, where the value of the self-feedback connection is gradually decreased [5]. This realizes a chaotic simulated annealing. However, both models still have a problem on computational cost for large-scale TSPs. Furthermore, it is difficult to obtain a good approximate solution for TSPs with more than few tens of cities.

In the present paper, we propose an effective algorithm for solving large-scale TSPs using a clustering technique. The basic concept of our method is that an intractable large-scale TSP is divided into some tractable small-scale TSPs. Here, the small-scale TSP refers to one which can be easily solved by RNNs. Kobayashi presented a method introducing a clustering technique for solving large-scale problems [6]. However, this method does not always give us good approximate solutions because it does not consider interactions between adjacent clusters. In the present paper, we focus on such interactions to obtain good approximate solutions for large-scale TSPs.

2 How to Solve Large-scale TSP Using CNN

In this section, transiently chaotic neural network (TCNN) proposed by Chen and Aihara [5], which is one of TSP solvers using CNNs, is described (Section 2.1). Then, formulation for TSP is introduced (Section 2.2). After that, the divide-and-conquer approach [6] for solving large-scale TSPs is presented (Section 2.3). Finally, a new effective method for solving large-scale

TSPs is proposed (Section 2.4).

2.1 Transiently Chaotic Neural Network

The TCNN is a type of RNNs and proposed for solving combinatorial optimization problems. The overview of TCNN is shown in figure 1. In this figure, the connections denoted by symbol ∘ are excitatory and •'s are inhibitory.

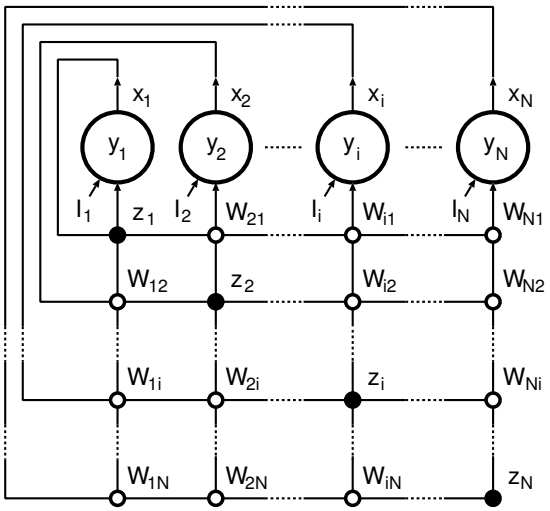

Figure 1: The structure of TCNN

The dynamics of TCNN with N neurons is characterized by the following equations [5]:

$$x_i(t) = \frac{1}{1+\exp(-y_i(t)/\epsilon)}, \tag{1}$$

$$y_i(t+1) = k y_i(t) + \alpha\left\{\sum_{j=1,\ j\neq i}^{N} W_{ij}x_j(t) + I_i\right\} - z_i(t)\{x_i(t) - I_0\}, \tag{2}$$

$$z_i(t+1) = (1-\beta)z_i(t), \tag{3}$$

where

x_i	output of neuron i ($0 \le x_i \le 1$)
y_i	internal state of neuron i
ϵ	slope parameter of sigmoid function ($\epsilon > 0$)
k	decay parameter of y_i ($0 \le k \le 1$)
α	positive scaling parameter for inputs
W_{ij}	connection weight between neuron i and j
I_i	external input to neuron i
z_i	self-feedback connection weight ($z_i > 0$)
I_0	positive constant
β	decay parameter of z_i ($0 \le \beta \le 1$)

The TCNN is characterized as follows. At the initial state, z_i is so large that a network state is chaotic and the network searches a global solution. Then, z_i is gradually decreased with time. The network is also gradually changed from chaotic state to steady state. An optimization process of TCNN is regarded as a chaotic simulated annealing (CSA) [5].

2.2 Formulation for TSP

In TCNN, the formulation for TSP is identical to Hopfield model [2]. For N-city TSP, the neurons are arranged as shown in figure 2. In this figure, each row specifies city and each column is visiting order. If neuron (i,k) is activated, city i is visited in the kth order.

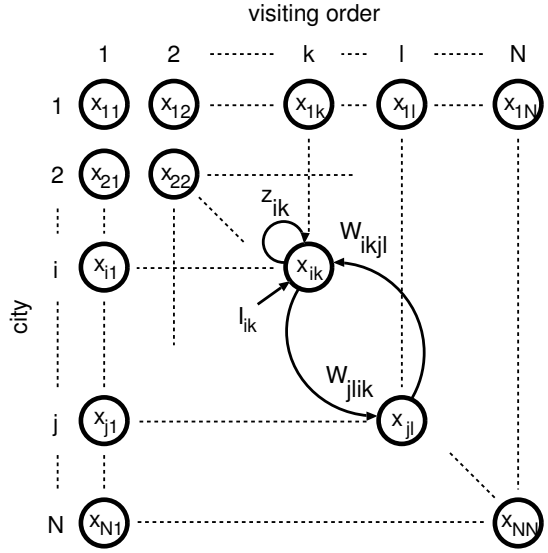

Figure 2: The arrangement of neurons for TSP

The energy function is defined as follows:

$$E = \frac{1}{2}\left(AE_1 + BE_2\right), \tag{4}$$

where A and B are constants and E_1 and E_2 are constraint term and distance term in energy function, respectively. E_1 and E_2 are defined as:

$$E_1 = \sum_{i=1}^{N}\left(\sum_{k=1}^{N} x_{ik} - 1\right)^2 + \sum_{k=1}^{N}\left(\sum_{i=1}^{N} x_{ik} - 1\right)^2, \tag{5}$$

$$E_2 = \sum_{i=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N} d_{ij}x_{ik}\left(x_{jk-1} + x_{jk+1}\right). \tag{6}$$

By comparing energy function E of Eq.(4) and the following general energy function,

$$E_{general} = -\frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N}\sum_{l=1}^{N} W_{ikjl}x_{ik}x_{jl} - \sum_{i=1}^{N}\sum_{k=1}^{N} I_{ik}x_{ik}, \tag{7}$$

connection weight W_{ikjl} and external input I_{ik} are derived as follows:

$$W_{ikjl} = -A\left\{\delta_{ij}(1-\delta_{kl}) + \delta_{kl}(1-\delta_{ik})\right\}$$

$$-Bd_{ij}(\delta_{lk+1} + \delta_{lk-1}), \quad (8)$$

$$I_{ik} = A, \quad (9)$$

where δ_{ij} is Kronecker's delta, i.e.

$$\delta_{ij} = \begin{cases} 1 & (i = k) \\ 0 & (otherwise) \end{cases} .$$

In the present paper, output x_{ik} is binarized as follows:

$$x_{ik}^D = \begin{cases} 1 & (x_{ik} > \overline{x}) \\ 0 & (otherwise) \end{cases} ,$$

where $\overline{x}$ refers to average of x_{ik}, i.e.

$$\overline{x} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{k=1}^{N} x_{ik}. \quad (10)$$

Chen and Aihara have shown that global solutions for 10- and 48-city problems were obtained using TCNN [5]. However, they focus only on the error rate, which is a measure of quality of the approximate solution, but not on computational cost. For example, when the number of cities becomes twice the number of neurons and computational cost per neuron also increases twice. As a result, total computational cost becomes four times. In fact, it is difficult to obtain a good approximate solution using TCNN for TSPs with more than 50 cities.

2.3 Divide-and-conquer Approach to Large-scale TSPs

To overcome the problems explained in the previous section, computational cost is reduced using a clustering technique. The basic idea is that an intractable large-scale TSP is divided into some tractable small-scale ones and then each small-scale TSP is solved using TCNN.

The following algorithm was proposed in [6].

1. Divide a large-scale TSP into some small-scale ones according to their coordinates. The K-mean method is used as a clustering technique because it is simple and requires no heavy computation.

2. Determine a visiting order among the clusters. Finding the visiting order is regarded as a TSP, called reference TSP. The coordinates of cities in the reference TSP are assumed as mean vectors in each cluster. Then solve the reference TSP.

3. Select two cities in each cluster, which have a shortest distance between the adjacent clusters and correspond to the starting and the last cities, respectively. this is realized that we set the external input for such cities to a higher value than the other cities. As a result, two neurons corresponding to such cities tend to fire.

4. Solve each small-scale TSP using TCNN.

5. Connect every two cities selected in step 3 to obtain a total tour of the large-scale TSP.

The algorithm will be illustrated using an example, which is 101-city problem, eil101.tsp, from TSPLIB [1]. The optimal tour is shown in figure 3. The coordinates of cities are normalized in the plane $[0, 1] \times [0, 1]$.

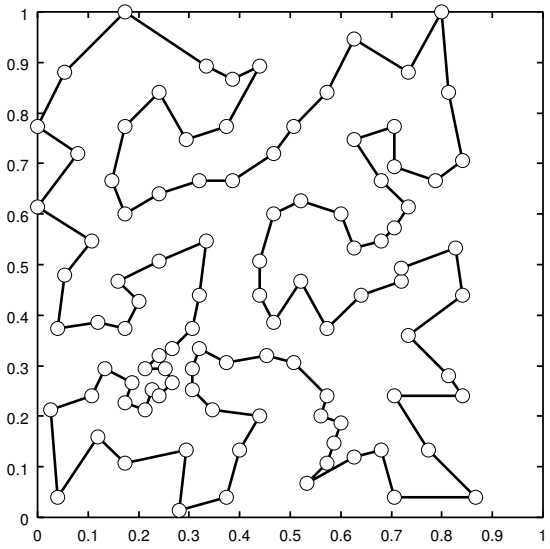

Figure 3: The optimal tour for 101-city problem (eil101.tsp)

At first, 101 cities are divided into 9 clusters, which have 20 to 30 cities as shown in figure 4. In this figure, the same symbol denotes the cities belonging to the same cluster. Next, a reference TSP composed of 9 mean vectors is solved using TCNN. The tour is illustrated by the dashed line in the figure. Then, 9 small-scale TSPs are solved using TCNN, respectively. Finally, by connecting the 9 tours we can obtain a total tour of 101-city problem, which is illustrated by the solid line in the figure.

This method, however, has the following drawbacks. When determining connecting cities between adjacent clusters, only local information on distance between two cities in adjacent clusters is used. As a result, it is difficult to obtain the optimal tour.

2.4 An Effective Method Considering Global Information

As described in the previous section, global information is needed to obtain good approximate solutions. In this section, we propose a new effective method for large-scale TSPs.

Our algorithm is described as follows.

1. Divide a large-scale TSP into some small-scale ones according to their coordinates using K-mean clustering method.

[1] http://www.iwr.uni-heidelberg.de/iwr/comopt/software/TSPLIB95/

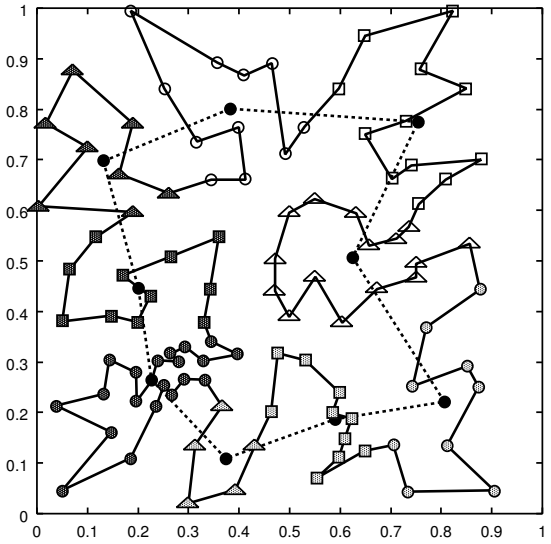

Figure 4: An approximate solution for 101-city problem using the divide-and-conquer approach

2. Determine a visiting order among the clusters by solving reference TSP.

3. Solve each small-scale TSP and simultaneously determine two connecting cities between adjacent clusters using TCNN.

Step 3 is concretely explained in the following. There are connections between adjacent clusters as shown in figure 5. If we focus on cluster g, there are connections between neurons in the first column of cluster g and in the last column of the previous cluster $g-1$. Let N^g denote the number of cities in cluster g. There are $N^{g-1} \times N^g$ connections between cluster g and $g-1$. Similarly, there are $N^g \times N^{g+1}$ connections between cluster g and the next cluster $g+1$.

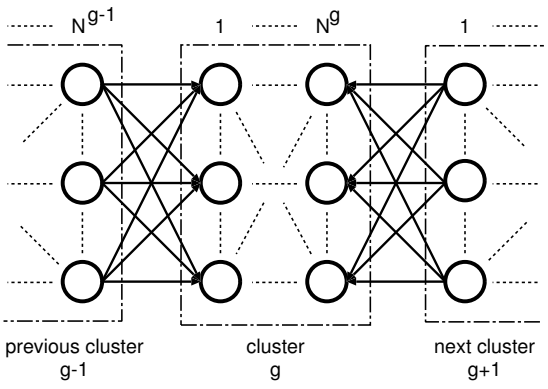

Figure 5: The arrangement of neurons for large-scale TSPs

Using this formation, the constraint term in energy function is described as:

$$E_1 = \sum_{i=1}^{N^g}\left(\sum_{k=1}^{N^g} x_{ik}^g - 1\right)^2 + \sum_{k=1}^{N^g}\left(\sum_{i=1}^{N^g} x_{ik}^g - 1\right)^2, \tag{11}$$

where x_{ik}^g refers to the output of neuron (i,k) in cluster g.

Then, the distance term in energy function consists of the following two terms, one represents total distance in the same cluster, E_{21}, and the other is that in adjacent clusters, E_{22}.

$$E_{21} = \sum_{i=1}^{N^g}\sum_{k=1}^{N^g}\sum_{j=1}^{N^g} d_{ij}^{gg} x_{ik}^g (x_{jk-1}^g + x_{jk+1}^g), \tag{12}$$

$$E_{22} = \sum_{i=1}^{N^g}\left(\sum_{j=1}^{N^{g-1}} d_{ij}^{gg-1} x_{i1}^g x_{jN^{g-1}}^{g-1} + \sum_{j=1}^{N^{g+1}} d_{ij}^{gg+1} x_{iN^g}^g x_{j1}^{g+1}\right), \tag{13}$$

where d_{ij}^{gg} refers to the distance between city i and j in the same cluster g and d_{ij}^{gg-1} is the distance between city i in cluster g and city j in cluster $g-1$.

Finally, distance term is defined as:

$$E_2 = CE_{21} + DE_{22}, \tag{14}$$

where C and D are constants.

As comparing energy function E and $E_{general}$ in Eq.(7), connection weights and external inputs are derived as:

$$W_{ikjl}^g = -A\left\{\delta_{ij}(1-\delta_{kl}) + \delta_{kl}(1-\delta_{ik})\right\} - BCd_{ij}^{gg}(\delta_{lk+1} + \delta_{lk-1}), \tag{15}$$

$$I_{ik}^g = A - \frac{1}{2}BD\left(\delta_{k1}\sum_{j=1}^{N^{g-1}} d_{ij}^{gg-1} x_{jN^{g-1}}^{g-1} + \delta_{kN^{g+1}}\sum_{j=1}^{N^{g+1}} d_{ij}^{gg+1} x_{j1}^{g+1}\right). \tag{16}$$

As an example, a solution using this method is shown in figure 6.

3 Computer Simulation

In the previous paper [6], we have shown that by introducing the clustering technique, computational cost is dramatically reduced and quality of the solution is also improved compared with three other methods, i.e. TCNN without a clustering method, greedy method and genetic algorithm.

In this section, the performance of two methods, our method proposed in the present paper, method #2, and method #1 [6] is evaluated.

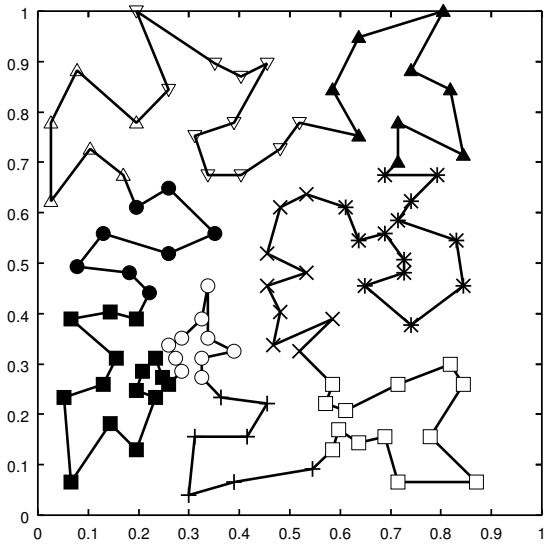

Figure 6: An approximate solution for 101-city problem using the proposed method

In simulations, the value of parameters is set as Table 1. Three TSPs, st70.tsp (figure 7), eil101.tsp (figure 3), and ch130.tsp (figure 8), from TSPLIB were used for evaluation.

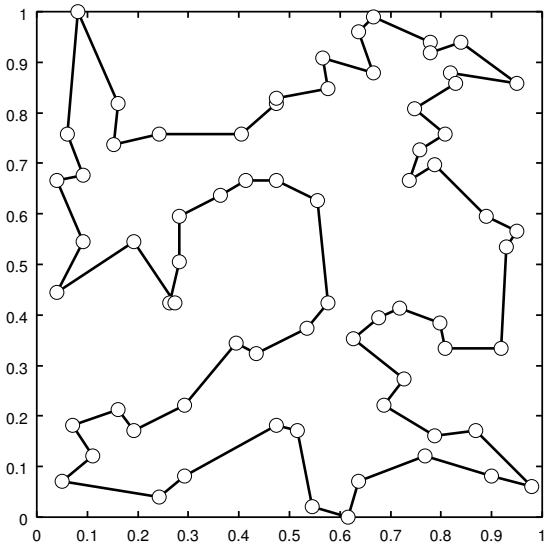

Figure 7: The optimal tour for 70-city problem (st70.tsp)

The simulation results are shown in Table 2. These results are averaged over 10 trials. As seen in this table, method #2 is superior to method #1 on tour length and error rate. But method #2 takes more steps to converge than method #1.

4 Summary and Conclusions

In the present paper, we have proposed the effective TSP solver for large-scale problem using CNN.

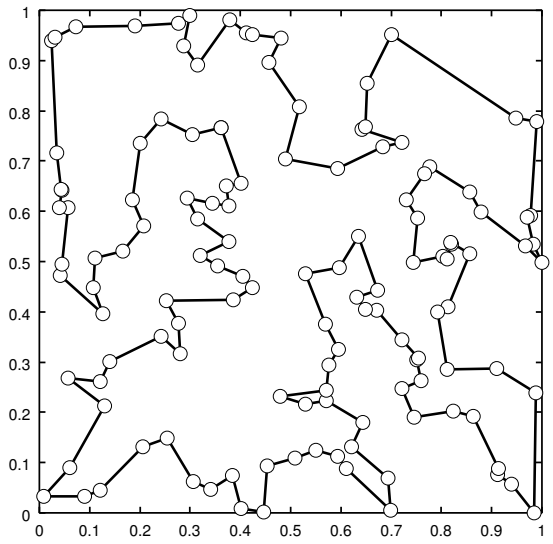

Figure 8: The optimal tour for 130-city problem (ch130.tsp)

Table 1: Parameters used in simulations

parameter	value
k	0.9
ϵ	0.004
I_0	0.65
α	0.015
β	0.003
$z_i(0)$	0.08
A	1.5
B	1.0
C	1.0
D	3.0

Table 2: Simulation results

TSP from TSPLIB	method #1		
	L	e(%)	n
st70.tsp	782	15.9	289
eil101.tsp	701	11.5	393
ch130.tsp	7334	20.0	619
TSP from TSPLIB	method #2		
	L	e(%)	n
st70.tsp	756	12.0	298
eil101.tsp	699	11.1	407
ch130.tsp	7137	16.8	638

L — tour length
e — error rate (%)
n — the number of steps to converge

Through computer simulations, it is shown that our method could solve large-scale problems with more than 50 cities, which cannot be solved by other CNNs. Using the clustering technique, the computational cost is dramatically reduced. By considering both global and local information, the quality of solution is improved.

The computational cost highly depends on the number of clusters and cities in a cluster. Moreover, the quality of the solution of reference TSP affects the final tour length. The development of the optimal clustering using neural networks is future work. The application to other combinatorial optimization problems should be investigated.

References

[1] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy-Kan, and D. B. Shmoys, editors. *The Traveling Salesman Problem — A Guided Tour of Combinatorial Optimization*. John Wiley & Sons, 1997.

[2] J. J. Hopfield and D. W. Tank. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

[3] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[4] H. Nozawa. A neural network model as a globally coupled and applications based on chaos. *Chaos*, 2(3):377–386, 1992.

[5] L. Chen and K. Aihara. Chaotic simulated annealing by a neural network model with transient chaos. *Neural Networks*, 8:915–930, 1995.

[6] K. Kobayashi. Introducing a clustering technique into recurrent neural networks for solving large-scale traveling salesman problems. In *Proceedings of the 8th International Conference on Artificial Neural Networks (ICANN98)*, volume 2, pages 935–940, 1998.