Introducing a Clustering Technique into Recurrent Neural Networks for Solving Large-Scale Traveling Salesman Problems

Kunikazu Kobayashi
Faculty of Engineering, Yamaguchi University
2557 Tokiwadai, Ube, Yamaguchi 755-8611, Japan
e-mail: k@csse.yamaguchi-u.ac.jp

Abstract

It is difficult to solve large-scale traveling salesman problems (TSPs) using recurrent neural networks (RNNs). Because local solutions increase remarkably as the number of cities does and then computational cost increases dramatically. In this paper, introducing a clustering technique, it is shown that RNNs could apply to large-scale TSPs. At first, a large-scale TSP is divided into some small-scale TSPs, which can be easily solved by RNNs. If necessary this process is continued recursively. Then, such small-scale TSPs are solved by RNNs. Finally, a total tour of the large-scale TSP is derived from connecting the tours of small-scale TSPs. Through computer experiments, it is verified that the proposed algorithm is effective for solving large-scale TSPs.

1 Introduction

It is well known that traveling salesman problem (TSP) is one of the NP-hard problems. The problem is 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 of its various applications in engineering such as a wiring of LSI substrates, scheduling and so on.

Hopfield and Tank proposed a recurrent neural network (RNN), so-called Hopfield model, for solving combinatorial optimization problems [1]. In their model, it is guaranteed that the network gradually changes its state so as to decrease a cost function defined for the problem. They have applied it to TSP and have shown that an approximate solution is obtained. As the number of cities increases, however, the network tends to be trapped into local minima because they are dramatically increased. In general, this problem has been escaped using annealing techniques like the Boltzmann machine [2]. On the other hand, regarding computational cost, it becomes much heavier.

Nozawa added a negative self-feedback to a discrete-type Hopfield model [3]. Since this allows each neuron to have a chaotic state, the 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 is gradually decreased [4]. This realizes a chaotic simulated annealing. However, both models still have a problem on computational cost for large-scale TSPs. Then it is difficult to obtain a good approximate solution for the problems with more than 50 cities.

In the present paper, we focus on the reduction of computational cost for the large-scale TSP and propose an effective algorithm using a clustering technique. In our method, a large-scale TSP is divided into small-scale TSPs by clustering the cities according to their coordinates. Here, the small-scale TSP refers to one which can be easily solved by RNNs. Then, each small-scale TSP is solved by RNN. Finally, a total tour is given by connecting the tours of small TSPs.

2 Solving large-scale TSP using RNN

In this section, transiently chaotic neural network (TCNN) proposed by Chen and Aihara, which is one of the TSP solvers using RNNs, is described (2.1). Then, a new method for solving large-scale TSPs is proposed (2.2).

2.1 Transiently chaotic neural network

The dynamics of TCNN with N neurons is characterized by the following equations [4].

$$\begin{aligned} x_i(t) &= \frac{1}{1+\exp(-y_i(t)/\epsilon)}, & (1)\\ y_i(t+1) &= ky_i(t) + \alpha\left\{\sum_{j=1,\ j\neq i}^{N} w_{ij}x_j(t) + I_i\right\} - z_i(t)\{x_i(t) - I_0\}, & (2)\\ z_i(t+1) &= (1-\beta)z_i(t), & (3)\end{aligned}$$

where x_i and y_i are an output and internal state of neuron i, respectively, ϵ is a slope parameter of sigmoid function, k is a decay parameter, α is a positive scaling parameter for inputs, w_{ij} is a connection weight between neuron i and j, I_i is an external input to neuron i, z_i is a self-feedback connection weight, I_0 is a positive constant, and β is a decay parameter of z_i.

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 to steady state. An optimization process of TCNN is regarded as a chaotic simulated annealing (CSA) [4].

In TCNN, an energy function and connection weights for solving TSP is same as Hopfield model [1]. Chen and Aihara have shown that global solutions for 10- and 48-city problems were obtained using TCNN [4]. However, they focus 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 the problems with more than 50 cities.

2.2 Dividing large-scale TSP

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

The proposed algorithm is described as follows.

1. Divide a large-scale TSP into some small-scale ones according to their coordinates. In the present paper, we used K-mean method 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 last cities, respectively. In the later experiment, this is realized that we set the external input for such cities to a higher value than the other cities. As a result, the 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.

If the number of cities in a cluster is so large that TCNN cannot solve the TSP our algorithm is applied recursively.

Our algorithm will be illustrated using an example, which is 101-city problem, eil101.tsp, from TSPLIB [1]. At first, 101 cities are divided into 9 clusters, which have 20 to 30 cities as shown in Fig.1. In this figure, the same symbol denotes the cities in 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.

3 Computer experiment

The first experiment (Experiment 1) confirms that the proposed method is effective for reduction of computational cost. The second experiment (Experiment 2) shows that it is possible to reduce computational cost and obtain a good approximate solution compared with greedy method and genetic algorithm (GA).

[1]http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html

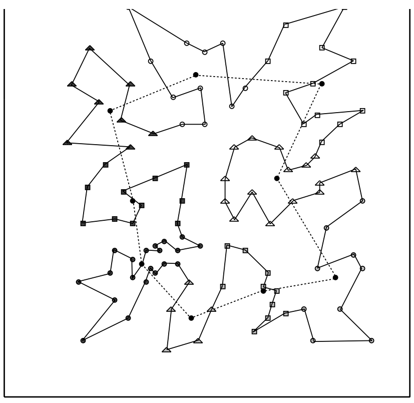

Figure 1: An example

3.1 Experiment 1

In this experiment, we compared our method with the original TCNN using the former 101-city problem, eil101.tsp. At first, we have to determine the number of clusters because of K-mean method. Table 1 shows the numerical result of both methods. In this table, CPU time of our method refers to the relative value when that of TCNN is assumed 100.

Table 1: Comparison of TCNN with and without clustering

TCNN without clustering		
No. of clusters	Error rate [%]	CPU time
-	15.6	100.0
TCNN with clustering		
6	10.0	4.31
7	7.8	4.00
8	7.2	2.62
9	5.7	1.78
10	8.4	1.49
11	8.4	1.49
12	7.5	1.49

As seen in this table, our method could dramatically reduce computational cost compared with TCNN without clustering. The error rate and computational cost depend on the number of clusters. The more the number of clusters is the lower computational cost is. Of course, the distribution of the cities and the number of cities in a cluster affect the quality of the solutions. Obviously, the optimal number of clusters for this problem is nine.

3.2 Experiment 2

In this experiment, our method was compared with two other TSP solvers, i.e. greeding method and genetic algorithm (GA) [5]. The three problems, pr76.tsp, eil101.tsp, and ch130.tsp in TSPLIB, were used for evaluation. The number of generations in GA was 1,000, 1,500, and 2,000 for pr76.tsp, eil101.tsp, and ch130.tsp, respectively.

Table 2 shows the numerical result of the experiments. In this table, GM, GA, and TC refer to greedy method, genetic algorithm, and our method, respectively. The result of GM represents the best solution in 10 trials and our method and GA are the average of 10 trials. Regarding CPU time, the time using GA is assumed 100.

Table 2: Comparison with other two methods, greedy method and GA

Method	Error rate [%]	CPU time	No. of clusters
76-city problem (pr76.tsp)			
GM	21.0	0.83	-
GA	3.0	100	-
TC	7.9	9.2	7
101-city problem (eil101.tsp)			
GM	14.6	1.72	-
GA	6.0	100	-
TC	5.7	5.17	9
130-city problem (ch130.tsp)			
GM	17.8	3.29	-
GA	9.1	100	-
TC	9.1	5.32	11

As seen in this table, the error rate is changed according to the distribution of cities. The greedy method is very fast because it uses only local information. But it cannot escape the local minima. Therefore, the quality of the solutions is worst. And the solution using GA is better but computational cost is high. On the other hand, the error rate of our method is almost same with that of GA but CPU time is lower than GA. Of course, the error rate depends on the number of clusters.

4 Applying to larger-scale TSP

The merit of the proposed method is that it can apply to larger-scale TSPs using recursive clustering.

Figure 2 shows an approximate solution for 1000-city problem, dsj1000.tsp, in TSPLIB using our method. In this case, the initial number of clusters is 20. Then the recursive clustering is applied. As a result, the final number of clusters is 63.

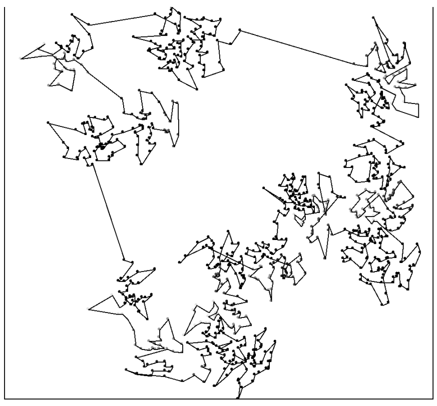

Figure 2: An approximate solution of 1000-city TSP (dsj1000.tsp)

5 Conclusions

In the present paper, we have proposed the TSP solver for large-scale problem using RNN. It is shown that our method could solve large-scale problems with more than 50 cities, which cannot be solved by other RNNs. Using the clustering technique, the computational cost is dramatically reduced.

The quality of solution and 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 optimal clustering and clustering using neural networks are future work. The application to other combinatorial optimization problems should be investigated.

References

[1] Hopfield J. J., Tank D. W. “neural” computation of decisions in optimization problems. Biological Cybernetics 1985; 52:141-152

[2] Ackley D. H., Hinton G. E., Sejnowski T. J. A learning algorithm for Boltzmann machines. Cognitive Science 1985; 9:147-169

[3] Nozawa H. A neural network model as a globally coupled and applications based on chaos. Chaos 1992; 2:377-386

[4] Chen L., Aihara K. Chaotic simulated annealing by a neural network model with transient chaos. Neural Networks 1995; 8:915-930

[5] Pai K. F. Genetic algorithm for the traveling salesman problem based on a heuristic crossover operation. Biological Cybernetics 1993; 69:539-546