2004 International Symposium on Nonlinear
Theory and its Applications (NOLTA2004)
Fukuoka, Japan, Nov. 29 - Dec. 3, 2004

An Approach to the TSP based on Growing Self-Organizing Maps

Takeshi Ehara †, Hiroki Sasamura ‡ and Toshimichi Saito†

†EECE Dept, Hosei University, Koganei, Tokyo, 184-8584 Japan
‡Pentax Co. Ltd., Itabashi, Tokyo, 174-8639, Japan,
Email: {ehara, sasamura}@nonlinear.k.hosei.ac.jp, tsaito@k.hosei.ac.jp

Abstract—This paper studies application of a simple self-organizing map to the traveling salesperson problem. The map has ring topology and the learning algorithm is characterized by two parameters. Applying city positions successively as input data, the map can grow flexibly and tour can be obtained. Basic experimental results suggest that our simple algorithm can find almost optimum solutions. Dependence of a parameter on the solutions is also discussed.

1. Introduction

Learning algorithms of self-organizing maps (ab. SOMs) are known as typical unsupervised learning algorithms which can extract features of input data automatically [1]. In order to increase flexibility and adaptability, self-organizing maps having growing cell structures (ab. GCS) have been studied [2]-[4]. The GCS can change the size and topology of the maps adaptively to the input data. Applications of the GCS include data visualizations, pattern classifications, vector quantizations and knowledge discovery [2]-[6]. If the GCS has tree or ring topology, it is applicable to image skeletonizations [7] [8] and the travelling salesperson problem (ab. TSP) [9] [10]. This paper studies an approach to the TSP based on the ring GCS (ab. RGCS). The TSP is a classical combinatorial optimization problem and is known to be NP-complete [11]. The task is to find the shortest possible tour through the set of M cities that passes through each city exactly once. In order to find the optimum solution, interesting algorithms have been studied: the Hopfield net approach [12], the elastic net approach [11], the RGCS approach [9] [10] and so on.

This paper studies a simple GCS-based algorithm to the TSP. The map has ring topology and can grow: the number of cells can increase. The algorithm is characterized by two parameters: learning rate and time interval to insert a cell. Applying city positions successively as input data, the map can grow flexibly and tour can be obtained. Although our algorithm is simpler than existing algorithms, we can find almost optimum solutions. The algorithm efficiency is confirmed by basic numerical experiments. Dependence of the insertion time interval on the solutions is also discussed.

2. The Ring GCS and algorithm

The map consists of cells and the number of cells is time variant. Let t be the discrete time and let $N(t)$ be the number of cells at time t. The i-th cell, $i \in \{1, \cdots, N(t)\}$, is connected with its closest neighbors $N_i = \{i-1, i+1\}$ where $i+1$ and $i-1$ are modulus $N(t)$: the RGCS has ring topology as shown in Fig. 1. At time t the position of the i-th cell is represented by its weight vector $w_i(t) \in \boldsymbol{R}^2$ and the i-th cell has the counter $C_i(t) \in \boldsymbol{Z}^+$ that controls the growing of the RGCS. $\boldsymbol{R}$ and $\boldsymbol{Z}^+$ denote reals and positive integers, respectively.

The objective data set consists of city positions

$$CP \equiv \{CP_1, \cdots, CP_M\}, \;\; CP_i \equiv (X_i, Y_i) \in \boldsymbol{R}^2$$

where M is the number of cities and CP_i is the i-th city position. Let $x(t)$ be the input at time t. One city position is selected from CP and is applies as an input to the RGCS.

In order to find the shortest possible tour, we present the following algorithm consisting of 8 steps.

STEP 1 (Initialization)
Let $t = 0$ and let $C_i(0)$=0.

STEP 2 (Input)
According to a uniform random number, we select one element from CP and apply it as an input $x(t)$.

STEP 3 (Winner)
We find the cell whose weight vector is the closest to

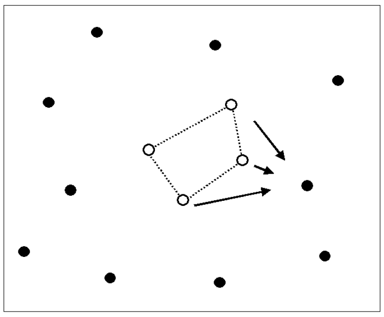

Figure 1: RGCS and its update. White circles denote cell positions and black circles denote city positions.

the input $x(t)$ and declare it as the winner c:

$$||x(t) - w_c(t)|| = \min_i ||x(t) - w_i(t)|| \tag{1}$$

where $||\cdot||$ denotes the Euclidean distance. If there exist plural closest cells we select one of them randomly.

STEP 4 (Update of weight vector and counter)
According to Equation (2) weight vectors of the winner and its neighbors are updated as illustrated in Fig. 1. The other weight vectors are preserved.

$$w_i(t+1) = \begin{cases} w_i(t) + \alpha(x(t) - w_i(t)) & \text{for} \quad i \in N_c \\ w_i(t) & \text{for} \quad i \notin N_c \end{cases} \tag{2}$$

where $N_c = \{c-1, c, c+1\}$ modulus $n(t)$. The learning rate α is the first parameter of this algorithm. According to Equation (3) counter of the winner is updated. The other counters are preserved.

$$C_i(t+1) = \begin{cases} C_i(t) + 1 & \text{for} \quad i = c \\ C_i(t) & \text{for} \quad i \neq c \end{cases} \tag{3}$$

STEP 5 (Insertion of a cell)
At every T_{int} times, we find the cell q that has the maximum counter value.

$$C_q(nT_{int}) > C_i(nT_{int}) \quad \text{for all} \quad i \neq q. \tag{4}$$

If there exist plural maximum counter values we select one of them randomly. This selection corresponds to inspection of the learning history. We then pick up closer neighbor cell f of the cell q.

$$f = \begin{cases} q-1 \bmod N(t) & \text{if } w_{q-1}(t) \text{ is closer} \\ q+1 \bmod N(t) & \text{if } w_{q+1}(t) \text{ is closer} \end{cases}$$

A novel cell r is inserted between q and f as shown in Fig. 2 and let $N(t+1) = N(t) + 1$. The weight vector of cell r is given by Equation (5)

$$w_r(t+1) = 0.5(w_q(t) + w_f(t)). \tag{5}$$

The insertion time interval T_{int} is the second parameter of this algorithm. Counter values of cells r and q are re-assigned by Equation (6)

$$C_q(t+1) = 0.5C_q(t), \;\; C_r(t+1) = 0.5C_q(t) \tag{6}$$

STEP 6 (Comparison of cities and cells)
If the number of cells exceeds the number of cities, $N(t+1) \geq M$, allocation of a cell to each city is possible and go to STEP 7. If $N(t+1) < M$ then $t = t+1$ and go to STEP 2.

STEP 7 (Allocation of cells)
For each city, we find the closest cell to the city as shown in Fig. 3. If there exist plural closest cells we select one of them randomly (the selection is only once for each cell). Since the RGCS has ring topology, a tour can be obtained after the allocation. Then go to STEP 8.

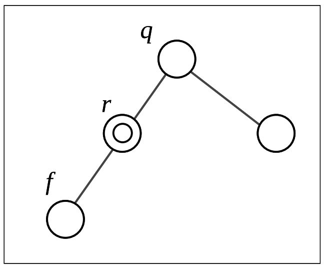

Figure 2: Insertion of a cell

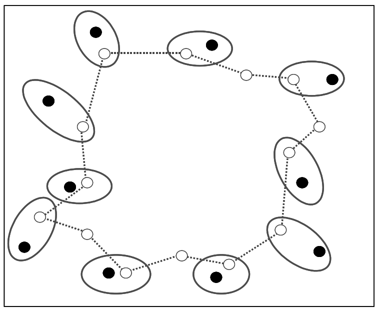

Figure 3: Finding a tour by allocation of cells. White circles denote cell positions and black circles denote city positions.

STEP 8
Let $t = t+1$. If $t < T_{max}$ then go to STEP 2, where T_{max} is the learning time limit. If $t = T_{max}$ then the learning is terminated.

3. Numerical Experiments

Noting the algorithm parameters are learning rate α and insertion time interval T_{int}, we apply the algorithm to a TSP with 52 cities at Web site

http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

At $t = 0$, we give initial conditions:

$$N(0) = 3, \; C_i(0) = 0$$

$$w_i(0) \in \text{ are 3 white circles in Fig. 4}$$

52 black circles in Fig. 4 correspond to the objective city positions and one of them are selected randomly as an input at time t. In this experiment, the learning rate is fixed $\alpha = 0.1$ and the insertion time interval is varied $T_{int} \in \{50, 100, 200\}$. As the learning goes on the map is growing as shown in Fig. 4. When the number of cells exceeds the number of cities, allocation of cells to all the cities is possible (STEP 8) and we can obtain a tour.

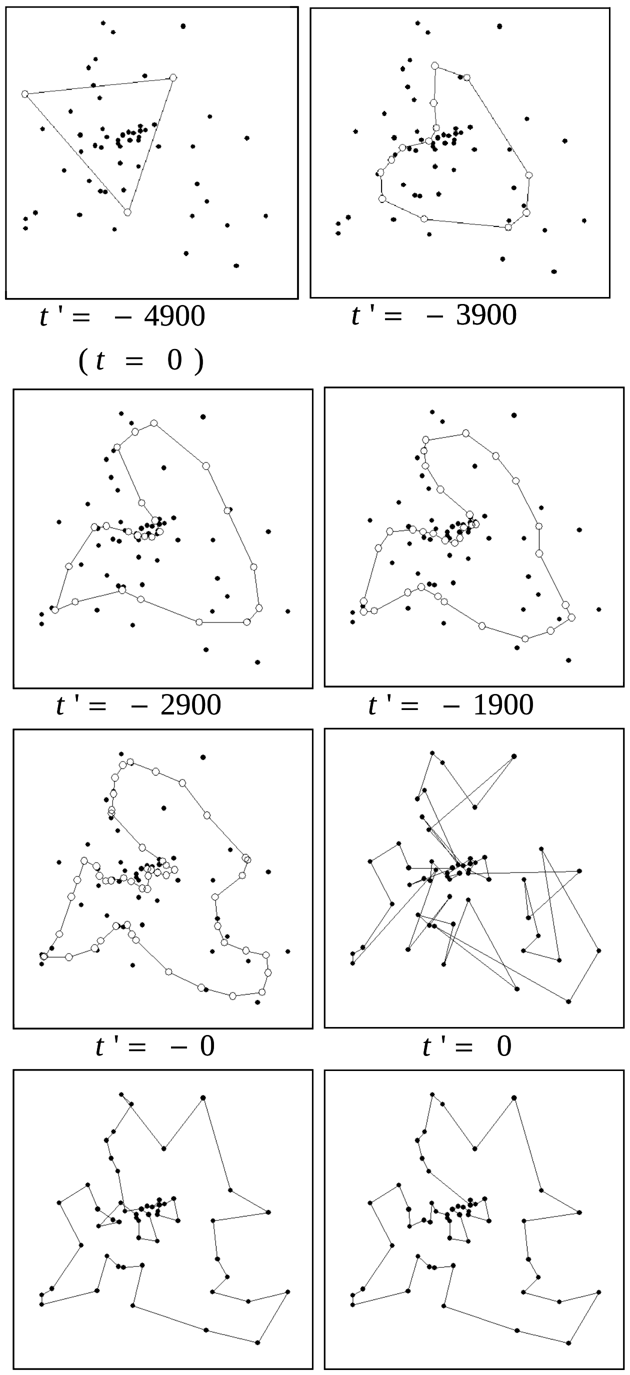

Figure 4: Learning process (T_{int} = 100, α = 0.1, $N(0)$ = 3) . white circles denote cell positions and black circles denote city positions. For $t' \geq 0$, cells are omitted and tour is shown.

Let T_1 be the time at when the number of cells attains the number of cities and let

$$t' = t - T_1. \tag{7}$$

t' (respectively, t) is the discrete time of the learning progress from T_1 (respectively, the beginning)to the termination For $t' < 0$ the tour does not exist. For $t' \geq 0$ the tour may be improved by continuing the learning. Fig. 5 shows several tour patterns at $t' = 2000$ for three values of T_{int} and a special case where $T_{int} = 100$ for $t' < 0$ and no insertion ($T_{int} = \infty$) for $t' \geq 0$. Each tour is characterized by the tour length L. These results suggest that the learning after T_1 can improve the tour provided T_{int} is selected suitably.

Fig. 6 shows the learning history for $t' \geq 0$. In this figure, the tour length is normalized by the known optimum tour length OPT. The doted curve denotes the special case without insertion: the number of cells is fixed for $t' \geq 0$. Usually tours are twisted at $t' = 0$ ($t = T_1$). Continuing inserting cells at T_{int} times after $t = T_1$, the twist can be removed and the tours can be improved. However, in the Special Case, the insertion is stopped after $t = T_1$ and improvement of the tours is hard. If the number of cells is small (respectively, large), each city has a few (respectively, many) options to capture a cell.

$$L' = \frac{L}{OPT} \tag{8}$$

We can see that the tour length closes rapidly to the optimum value ($L' = 1$) for $T_{int} = 50$. As T_{int} increases to 100 and then to 200, the initial approach to the optimum value delays but final value closes to the optimum value. For T_{int}= 50 each city has many options to capture a cell at relatively early step and the tour length can close to the optimum value. For T_{int}= 200 each city has a few options to capture a cell and improvement of the tour is hard at early steps. However, as the learning goes, the number of cells increases and the improvement is to be possible.

It should be noted that our method is simpler than the method in [13]. In [13], the learning rate α depends on distance between winner and other cells and the region of neighbor depend on the α. In our method, α is constant and region of neighbor is fixed.

4. Conclusions

We have studied a simple learning algorithm to the TSP. The algorithm is based on the GCS with ring topology and we can obtain almost optimum tour (solutions) provided the two parameters are selected suitably. Effects of the insertion time interval is also discussed. Future problems include analysis of the learning process, application to larger scale problems and automatic parameters setting.

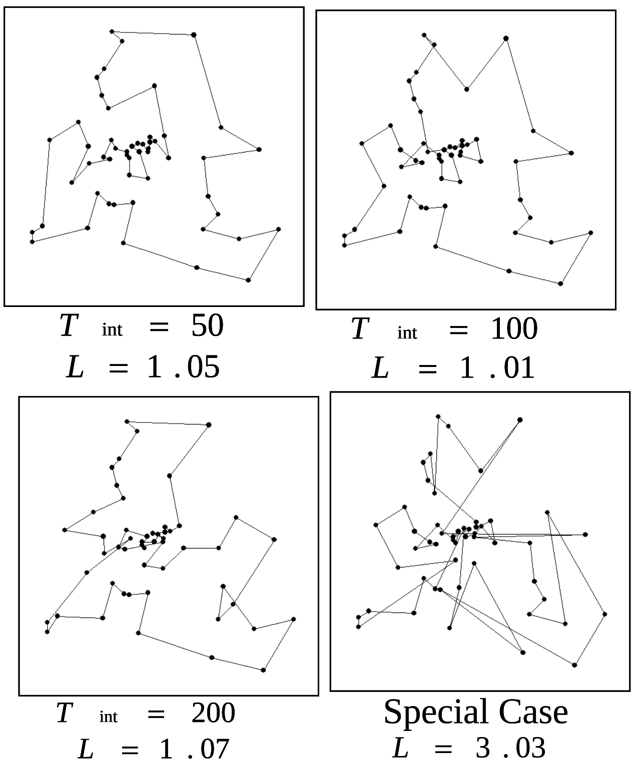

Figure 5: Tour at time $t' = 2000$ ($\alpha = 0.1$). Special Case : $T_{int} = 100$ for $t' \leq 0$ and no insertion ($T_{int} = \infty$) for $t' > 0$.

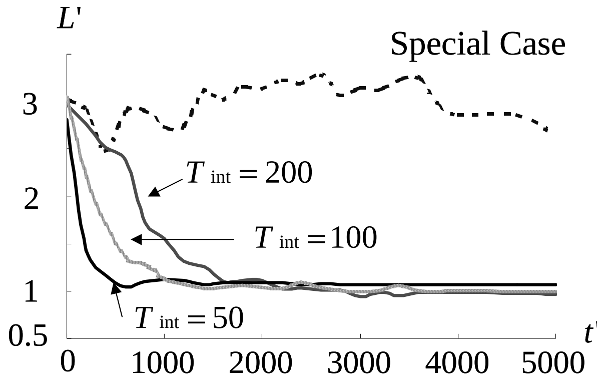

Figure 6: Tour length and insertion time interval T_{int}

References

[1] T. Kohonen, Self-organization and associative memory, 2nd Ed., Springer-Verlag, Berlin (1988)

[2] B. Fritzke, Growing cell structures - a self-organizing network for unsupervised and supervised learning, Neural Networks, 7, pp.1441-1460, 1994.

[3] B. Fritzke, Growing neural gas network learns topologies, Advances in Neural Information Processing Systems, 7, pp. 625-632, 1995

[4] S.Kawahara and T.Saito, An adaptive self-organizing algorithm with virtual connection, J. Advanced Comput. Intelli., 2, 6, pp. 203-207, 1998.

[5] D. Alahakoon, S. K. Halganmuge and B. Srinivasan, Dynamic self-organizing maps with controlled growth for knowledge discovery, IEEE Trans. Neural Networks, 11, 3, pp. 601-614, 2000.

[6] R. Ohta and T. Saito, A growing self-organizing algorithm for dynamic clustering, Proc. of IJCNN, pp.469-473, 2001.

[7] R. Singh, V. Cherkassky and N. Papanikolopoulos, Self-organizing maps for the skeletonization of sparse shapes, IEEE Trans. Neural Networks, 11, 1, pp. 241-248, 2000

[8] H. Sasamura and T. Saito, A Simple learning algorithm for growing self-organizing maps and its application to the skeletonization, Proc. of IJCNN, pp. 787-790, 2003

[9] B. Angeniol, G. de La C. Vaubois and J. Y. Le Texierr, Self-organizing feature maps and the traveling salesman problem, Neural Networks, 1, pp. 289-293, 1988.

[10] H. Sasamura, R. Ohta and T. Saito, A simple learning algorithm for growing ring SOM and its application to TSP, Proc. ICONIP, CD-ROM#1508, 2002.

[11] R. Durbin, R. Szeliski and A. Yuille, An analysis of the elastic net approach to the traveling salesman problem, Self-organization map formation, pp. 407-417, MIT press, ed. K. Obermayer and T. J. Sejnowski, 2001.

[12] J. J. Hopfield and D. W. Tank, Neural computation of decisions in optimization problems, Biol. Cybernet, 52, pp.141-152, 1985.

[13] K. Fujimura, H. Tokutaka and M. Ishikawa, Performance of improved SOM-TSP algorithm for traveling salesman problem of many cities, Trans. IEE Japan, 119-C, 7, pp.875-882 (in Japanese, 1999)