An new self-organizing maps strategy for solving the traveling salesman problem

Yanping Bai a,*, Wendong Zhang a, Zhen Jin b

a Key Lab of Instrument Science and Dynamic Measurement of Ministry of Education, North University of China, No. 3, Xueyuan Road, TaiYuan, ShanXi 030051, China
b Department of Applied Mathematics, North University of China, No. 3 Xueyuan Road, TaiYuan, ShanXi 030051, China

Accepted 15 August 2005

Abstract

This paper presents an approach to the well-known traveling salesman problem (TSP) using self-organizing maps (SOM). There are many types of SOM algorithms to solve the TSP found in the literature, whereas the purpose of this paper is to look for the incorporation of an efficient initialization methods and the definition of a parameters adaptation law to achieve better results and a faster convergence. Aspects of parameters adaptation, selecting the number of nodes of neurons, index of winner neurons and effect of the initial ordering of the cities, as well as the initial synaptic weights of the modified SOM algorithm are discussed. The complexity of the modified SOM algorithm is analyzed. The simulated results show an average deviation of 2.32% from the optimal tour length for a set of 12 TSP instances.

© 2005 Published by Elsevier Ltd.

1. Introduction

The traveling sales man problem (TSP) is a typical combinatorial optimization problem. It can be understood as a search for the shortest closed tour that visits each city once and only once. The decision-problem form of TSP is a NP-complete [1], thence the great interest in efficient heuristics to solve it. There are many of the heuristics that utilize the paradigm of neural computation or related notions recently [2]. The first approach to the TSP via neural networks was the work of Hopfield and Tank in 1985 [3], which was based on the minimization of an energy function and the local minima should correspond to a good solution of a TSP instance. However, this approach does not assure the feasibility, that is, not all the minima of the energy function represent feasible solution to the TSP.

The SOM algorithm, originally introduced by Kohonen [4], is an unsupervised learning algorithm, where the learning algorithm established a topological relationship among input data. This is the main characteristic explored by this work and by many others found in the literature that approached the TSP via modifications of the SOM algorithm. There are many types of SOM algorithms to solve the TSP [5–10], the purpose of this paper is to look for the incorporation of an efficient initialization method, and the definition of a parameters adaptation law to achieve better results and a faster convergence.

* Corresponding author.
E-mail addresses: baiyp@nuc.edu.cn (Y. Bai), wdzhang@nuc.edu.cn (W. Zhang).

0960-0779/$ - see front matter © 2005 Published by Elsevier Ltd.
doi:10.1016/j.chaos.2005.08.114
This paper is organized as follows. The SOM algorithm is briefly described in Section 2. Parameters adaptation of the learning rate and the neighborhood function variance are presented in Section 3. The initialization methods are discussed in Section 4. Computational experiments and comparisons with other methods are presented in Section 5. The algorithm complexity is discussed in Section 6. Conclusions are presented in Section 7.

2. Self-organizing maps and algorithm

There are many different types of self-organizing maps; however, they all share a common characteristic, i.e. the ability to assess the input patterns presented to the networks, organize itself to learn, on its own, based on similarities among the collective set of inputs, and categorize them into groups of similar patterns. In general, self-organizing learning (unsupervised learning) involving the frequent modification of the network’s synaptic weights in response to a set of input patterns. The weight modifications are carried out in accordance with a set of learning rules. After repeated applications of the patterns to the network, a configuration emerges that is of some significance.

Kohonen self-organizing map belong to a special class of neural networks denominated Competitive Neural Networks, where each neuron competes with the others to get activated. The result of this competition is the activation of only one output neuron at a given moment. The purpose of Kohonen self-organizing map is to capture the topology and probability distribution of input data [11]. This network generally involved an architecture consisting of uni- or bi-dimensional array of neurons. The original uni-dimensional SOM topology is similar to a bar and is presented in Fig. 1. A competitive training algorithm is used to train the neural network. In this training mode, not only the winning neuron is allowed to learn, but some neurons within a predefined radius from the winning neuron are also allowed to learn with a decreasing learning rate as the cardinality distance from the winning neuron increases. During the training procedure, synaptic weights of neurons are gradually changed in order to preserve the topological information of the input data when it is introduced to the neural networks.

To apply this approach to the TSP, a two-layer network, which consists of a two-dimensional input unit and \( m \) output units, is used. The evolution of the network may be geometrically imagined as stretching of a ring toward the coordinates of the cities. Fig. 2 represents the ring structure proposed. The input data is the coordinates of cities and the weights of nodes are the coordinates of the points on the ring. The input data (a set of \( n \) cities) are presented to the

Fig. 1. Uni-dimensional SOM structure.

Fig. 2. Toroidal SOM structure.
network in a random order and a competition based on Euclidian distance will be held among nodes. The winner node is the node \( J \) with the minimum distance to the presenting city. \( J = \text{Argmin}_j \{ \| x_i - y_j \| \} \), where \( x_i \) is the coordinate of city \( i \), \( y_j \) is the coordinate of nodes \( j \) and \( \| \cdot \|_2 \) is the Euclidean distance.

Furthermore, in each iteration, the winner node and its neighbors move toward the presenting city \( i \) using the neighborhood function:

\[
f(\sigma, d) = e^{-d^2/\sigma^2},
\]

according to the following updated function:

\[
y_j^{\text{new}} = y_j^{\text{old}} + \alpha f(\sigma, d) (x_i - y_j^{\text{old}}),
\]

where \( \alpha \) and \( \sigma \) are learning rate and neighborhood function variance, respectively. And \( d = \text{min}\{ \| j - J \|, m - \| j - J \| \} \) is the cardinal distance measured along the ring between nodes \( j \) and \( J \), where \( \| \cdot \| \) represents absolute value and \( m \) is the number of the neurons. The neighborhood function determines the influence of a city on the neighbor nodes of the winner. We noted that the update of neighbor nodes in response to an input creates a force that preserves nodes close to each other and creates a mechanism for shortening the constructed tour.

3. Parameters adaptation

The SOM algorithm has two adaptive parameters, the learning rate \( \alpha \) and the neighborhood function variance \( \sigma \). Kohonen [4] proposed an exponential evolution of this parameters in order to achieve a better convergence of the SOM algorithm, of the form:

\[
\alpha_k = \alpha_0 \times \exp\left(-\frac{k}{\tau_1}\right),
\]

\[
\sigma_k = \sigma_0 \times \exp\left(-\frac{k}{\tau_2}\right),
\]

where \( k = 0, 1, 2, \ldots \), is the number of iterations and \( \tau_1 \) and \( \tau_2 \) are the exponential time constants. “However, Eqs. (3) and (4) are heuristics and other methods can be adopted. It is must be remembered that the parameters adaptation is crucial for the convergence of the algorithm” [12].

During experiments considering the SOM applied to the TSP there were identified new parameter adaptation heuristics, which have led to a faster convergence. The adaptation laws proposed for these parameters are presented as follows:

\[
\alpha_k = \frac{1}{\sqrt{k}},
\]

\[
\sigma_k = \sigma_{k-1} \times (1 - 0.01 \times k), \quad \sigma_0 = 10.
\]

Eq. (5) determines the evolution of the learning rate. There is no need to define an initial value to this parameter since it depends only on the number of iterations.

Eq. (6) determines the evolution of the neighborhood function variance. This parameter requires an appropriate initialization, where \( \sigma_0 = 10 \) has been adopted.

4. Algorithm initialization

The initialization method is very important for the algorithm convergence and has great influence on the processing time to achieve a reasonable topological map configuration for TSP instances.

4.1. Selecting the number of nodes of neurons and neighbor length

Since selecting the number of nodes that is equal to the number of cities makes the process of node separation more delicate and results in sensitiveness of algorithm to the initial configuration, the authors recommended selecting the number of nodes as twice of the number of cities \( (m = 2n) \).
The neighborhood function, which determines the influence of the presented city on the lateral nodes of the winner, is another important factor. For the winner and its adjacent nodes, this attraction force is quite high and it deteriorates as the proximity node to the winner decreases. Considering this fact, we recommended that the effective neighbor length, which refers to the number of nodes that should be updated when a new winner is selected, is limited to 40% of the nodes. And the neighbor length also decreases gradually as time, which leads to a lower processing time.

4.2. Index of winner neurons and effect of the initial ordering of the cities

In order to prevent nodes from being selected as the winner for more than one city in each complete cycle of iterations, an inhibitory index is defined for each node, which puts the winner node aside from the competition, providing more opportunity for other nodes.

In the modified SOM algorithm, as well as the original algorithm, each city should be chosen only once during one complete iteration. Starting calculation with different sets of initial order will lead to different solution. The authors recommended to randomly redefining the order of the cities after a complete cycle of iterations to increase the robustness of the algorithm and reduce its dependence on initial configuration and the order of input data.

4.3. The initialization method of synaptic weights of neurons

Our choice of the initial curves is virtually unlimited. The question is, therefore, what kind of curves is “best” to use for a TSP with a given number of cites, based on a well-defined performance such as the minimum execution time or minimum tour length. There does not exist any analytical “answer” to the above posed question, but an attempt is made here to investigate the effect of starting the algorithm with different initial curves, using numerical experiments. Certainly, one cannot expect to investigate all possible curves, since such an “all out” investigation will be expensive. For this reason, we have limited our search to some cases.

The first method implemented was the random initialization of neurons over the input data space (TSP “cites”) [12]. The main drawback of this approach is that the neurons are initially scrambled, which requires a greater processing time to achieve a topological configuration that assures the neurons neighborhood.

The second implemented method was the random initialization of neurons on a circular ring whose center and the $n$ cites’ centroid coincide. However, we found it was little effect on tour length.

The third implemented method was the initialization of neurons with a tour. To construct a tour, we used the nearest neighbor algorithm [13]. Based on numerical experiments we found that initialization of the algorithm with a tour has little effect on execution time and tour length.

The fourth implemented method was the random initialization of neurons on a rhombic frame located in the right of the $n$ cites’ centroid. When starting calculation with different initial ordering of the cities, the rhombic frame acts as a ring that converges to the cities as the learning process evolution, keeping neurons neighborhood and yielding a faster convergence and better result. The initial condition, intermediate iterations and the final result (obtained after 30 iterations) of the modified algorithm for a TSP instance are illustrated on Fig. 3, where “*” represented the cities and the neurons are represented by “Æ” on the connected rings.

5. Computational experiments

The modified growing ring SOM approach for TSP (MGSOM) is briefly described as follows:

1. Initialization: Let $n$ is the number of cites. The algorithm inputs are the cartesian coordinate of the set of $n$ cites. Let $m(t)$ is the number of nodes on the rhombic frame located in the right of the $n$ cites’ centroid ($m(0) = n$, initially). Let the initial value of gain parameter, $\sigma_0 = 10$. And let $t = 1$ and $t_{\text{min}}$ be the value of the desired minimum number of iterations.

2. Randomizing: Starting calculation with different initial ordering of the cities and label cities 1, $\ldots$, $n$. Let $i$ be the index of the city in presentation. Set $i = 1$ and reset the inhabitation status of all nodes to false.

   Inhibit[$j$] = false for $j = 1, \ldots, m$.

3. Parameters adaptation: The learning rate $\alpha_k$ and the neighborhood function variance $\sigma_k$ are calculated using Eqs. (5) and (6), respectively.
4. Competition: Through a competitive procedure, select the closed node to city \( i \) based on Euclidean distance. This competition is held among only nodes, which have not been selected as a winner in this iteration. Therefore, the winner node \( J \) may be selected according to the following statement:

\[
J = \text{Argmin}_{j} \| x_i - y_j \| \text{ for } \{ j | \text{inhibit}[j] = \text{false} \}.
\]

Then, change the inhabitation status of the winner node: Inhibit \([ J ] = \text{true}\).

5. Adaptation: According to Eqs. (1) and (2), synaptic vectors of the winner and its neighbors are updated and other synaptic vectors are preserved at time \( t \). The neighbor length of the node \( J \) is limited to 40% of \( m(t) \). \( C_j(t) \) is a signal counter for inspection of the learning history. The signal counter of the winner is updated and other signal counters preserve their values at time \( t \).

\[
C_j(t + 1) = \begin{cases} 
C_j(t) + 1, & \text{if } i = J \\
C_j(t), & \text{otherwise}.
\end{cases}
\] (7)

6. Insertion a node: Inserts a novel node at every \( T_{\text{int}} \) learning times hereby the map can grow up to \( m(t) = 2n \). At \( t = kT_{\text{int}} \), \( k \) is a positive integer, we determine a node \( p \) whose signal counter has the maximum value.

\[
C_p(t) \geq C_j(t), \text{ for all } j.
\] (8)

If there exist plural maximum counter value we randomly select one of the nodes. This is an inspection of the learning history. We take a neighbor \( f = p - 1 \), where \( f \) is modulus \( m(t) \). A novel node \( r \) is inserted between \( p \) and \( f \). The synaptic vector of \( r \) is initialized as follow:

\[
y_r = 0.5(y_p + y_f).
\] (9)

Counter values of \( p \) and \( r \) are re-assigned as follows:

\[
C_p(t + 1) = 0.5C_p(t), \quad C_r(t + 1) = 0.5C_r(t).
\] (10)
The insertion interval $T_{\text{int}}$ is a control parameter of this algorithm. After the insertion, the number of nodes increases: let $n(t+1) = n(t) + 1$.

7. Set $i = i + 1$. If $i < n + 1$, go to step 4. Otherwise, set $t = t + 1$ and go to the next step.
8. Convergence test: Return to step 2 while $t \leq t_{\text{min}}$.

In order to obtain better quality of solution, the simulations by MGSOM are run as follows:

**Step 1:** Run 10 simulations with $t_{\text{min}} = 20$ iterations by using the MGSOM;

**Step 2:** Select the best solution from the solution of ten simulations runs in step 1.

**Step 3:** Run a simulation with $t_{\text{min}} = 10$ iterations of training that has the weight matrix associated with the best solution that selected in step 2 as its initial weight matrix.

In order to evaluate the obtained results, some results presented in Frederico et al. [12] were taken as a reference for comparisons, in which the author compares a proposed algorithm denominated SETSP (SOM efficiently applied in the TSP) with other four SOM based heuristics (KNIES-TSP-Kohonen Network Incorporating Explicit Statistics, local-KL-global-KG [5], PKN-Pure Kohonen Network [8,9], GN-Guilty Net [14], and AVL—The procedure of Ange'niol, Vaubois and Le Texier [10]). The sets of data used for the experiment are TSP instances available on the TSPLIB [15]. The distance between two cities is calculated using the Euclidian norm. Table 1 shows the TSP instances used.

Results obtained from the application of the modified algorithm MGSOM to some TSP instance and comparisons with the results presented in Frederico et al. [12] are presented in Tables 2 and 3. The presented values represent the suboptimum length achieved for each TSP instance.

### Table 1
TSP instances used for computational experiments

<table>
<thead>
<tr>
<th>Instances</th>
<th>No. of cities</th>
<th>Optimum length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bier127</td>
<td>127</td>
<td>118,282</td>
</tr>
<tr>
<td>Eil51</td>
<td>51</td>
<td>426</td>
</tr>
<tr>
<td>Eil76</td>
<td>76</td>
<td>538</td>
</tr>
<tr>
<td>kroA200</td>
<td>200</td>
<td>29,368</td>
</tr>
<tr>
<td>Lin105</td>
<td>105</td>
<td>14,379</td>
</tr>
<tr>
<td>pcb442</td>
<td>442</td>
<td>50,778</td>
</tr>
<tr>
<td>Pr107</td>
<td>107</td>
<td>44,303</td>
</tr>
<tr>
<td>Pr6</td>
<td>6</td>
<td>96,772</td>
</tr>
<tr>
<td>Pr152</td>
<td>152</td>
<td>73,682</td>
</tr>
<tr>
<td>Rat195</td>
<td>195</td>
<td>2323</td>
</tr>
<tr>
<td>ed100</td>
<td>100</td>
<td>7910</td>
</tr>
<tr>
<td>st70</td>
<td>70</td>
<td>675</td>
</tr>
</tbody>
</table>

### Table 2
Results from PKN, GN, AVL and MGSOM applied to TSP instance

<table>
<thead>
<tr>
<th>Instances</th>
<th>PKN</th>
<th>GN</th>
<th>AVL</th>
<th>MGSOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bier127</td>
<td>122211.7</td>
<td>155163.2</td>
<td>122673.9</td>
<td>119,580</td>
</tr>
<tr>
<td>Eil51</td>
<td>443.9</td>
<td>470.7</td>
<td>443.5</td>
<td>431.9569</td>
</tr>
<tr>
<td>Eil76</td>
<td>571.2</td>
<td>614.3</td>
<td>571.3</td>
<td>556.2061</td>
</tr>
<tr>
<td>kroA200</td>
<td>30927.8</td>
<td>39370.2</td>
<td>30994.9</td>
<td>29,947</td>
</tr>
<tr>
<td>Lin105</td>
<td>15374.2</td>
<td>15469.5</td>
<td>15311.7</td>
<td>14,383</td>
</tr>
<tr>
<td>pcb442</td>
<td></td>
<td></td>
<td>59649.8</td>
<td>55,133</td>
</tr>
<tr>
<td>Pr107</td>
<td>44504.3</td>
<td>80481.3</td>
<td>45096.4</td>
<td>44,379</td>
</tr>
<tr>
<td>Pr136</td>
<td>103878.0</td>
<td>5887.7</td>
<td>103442.3</td>
<td>98,856</td>
</tr>
<tr>
<td>Pr152</td>
<td>74804.2</td>
<td>105230.1</td>
<td>74641.0</td>
<td>74,228</td>
</tr>
<tr>
<td>Rat195</td>
<td></td>
<td></td>
<td>2681.2</td>
<td>2462</td>
</tr>
<tr>
<td>rd100</td>
<td>87.9</td>
<td>8731.2</td>
<td>8265.8</td>
<td>8002.7</td>
</tr>
<tr>
<td>st70</td>
<td>692.8</td>
<td>755.7</td>
<td>693.3</td>
<td>682.9886</td>
</tr>
</tbody>
</table>
Table 4 presented the percentage deviations from each TSP instance optimum length (specified in Table 1) for each algorithm analyzed. The average percentage deviation is also presented. By analyzing our numerical results, we notice that the quality of solutions of MGSOM is the average better than those algorithm presented in Frederico et al. [12]. The experiments cannot be declared representative, due to the small set of instances tested. However, it is can be observed the MGSOM superiority for the TSP instance tested. It should be remembered that better results would be obtained using MGSOM by adjusting the number of neurons.

6. Algorithm complexity

The complexity of the improved algorithm (for each value of $t$) may be evaluated as follows: for each city there are $0.3m$ (average neighbor length) operations for calculation of neighborhood function. Therefore, the time complexity function for each iterations may be given by $T(m,n) = n(0.3m)$. Furthermore, the complexity reduction is guided by the neighborhood function variance. Analyzing the variance evolution, it can be seen that the algorithm processing time decreases fast (due to the selective update). This feature is mainly attractive when processing large data sets.

7. Conclusions

Some adaptations were proposed to the SOM algorithm in order to provide an efficient method to solve the TSP. Computer programs developed in MATLAB for the heuristic were used to solve twelve test problems using a standing
desktop computer. Like the existing heuristic, the modified heuristic possesses many of these advantages of a good heuristic for the TSP solution. These advantages are: (1) easy implementation, (2) fast computation, (3) robust applicability, and (4) production of good solutions.

The contributions were the incorporation of an efficient initialization methods and the definition of a parameters adaptation law to achieve better results and a faster convergence. The results of a comparison between the existing heuristic SETSP and the modified heuristic MGSOM also show that the modified heuristic yields better results to the test problems. The MGSOM is well suited for larger instances of the TSP since it has a fast convergence and low complexity.

Acknowledgments

The authors are thankful that the research is supported by the Nature Science Foundations of Nation (10471040) and Shanxi Province (20051006) in China.

References