An Efficient Self-Organizing Map Designed by Genetic Algorithms for the Traveling Salesman Problem

Hui-Dong Jin, *Member, IEEE*, Kwong-Sak Leung, *Senior Member, IEEE*, Man-Leung Wong, *Member, IEEE*, and Zong-Ben Xu

***Abstract*—As a typical combinatorial optimization problem, the traveling salesman problem (TSP) has attracted extensive research interest. In this paper, we develop a self-organizing map (SOM) with a novel learning rule. It is called the integrated SOM (ISOM) since its learning rule integrates the three learning mechanisms in the SOM literature. Within a single learning step, the excited neuron is first dragged toward the input city, then pushed to the convex hull of the TSP, and finally drawn toward the middle point of its two neighboring neurons. A genetic algorithm is successfully specified to determine the elaborate coordination among the three learning mechanisms as well as the suitable parameter setting. The evolved ISOM (eISOM) is examined on three sets of TSPs to demonstrate its power and efficiency. The computation complexity of the eISOM is quadratic, which is comparable to other SOM-like neural networks. Moreover, the eISOM can generate more accurate solutions than several typical approaches for TSPs including the SOM developed by Budinich, the expanding SOM, the convex elastic net, and the FLEXMAP algorithm. Though its solution accuracy is not yet comparable to some sophisticated heuristics, the eISOM is one of the most accurate neural networks for the TSP.**

***Index Terms*—Convex hull, genetic algorithms, neural-evolutionary system, neural networks, self-organizing map, traveling salesman problem.**

I. Introduction

THE TRAVELING salesman problem (TSP) is one of the typical combinatorial optimization problems. It can be stated as a search for the shortest closed tour that visits each city once and only once. There are several real-life applications of the TSP such as, VLSI routing [31], hole punching [29], and wallpaper cutting [25]. On the other hand, it falls into a class of the NP-hard or NP-complete problems. Therefore, the research on the TSP is theoretically important. During the past decades, the TSP has attracted extensive research and has been repeatedly used as the basis of comparison for different optimization algorithms such as genetic algorithms (GAs) [11], [23], tabu search [20], automata networks [37], local search [17], ant colony system [7], and neural networks [2], [4]. These diverse approaches have demonstrated various degrees of strength and success. This paper focuses on an improved neural network that generates near optimal TSP solutions with quadratic computation complexity.

There are mainly two types of neural network approaches for the TSP: the Hopfield-type neural networks [14] and the Kohonen-type self-organizing map (SOM-like) neural networks [2]–[4], [21]. The underlying idea of the Hopfield-type networks is to find solutions by automatically searching for the equilibrium states of one dynamic system corresponding to the problem under consideration. The Hopfield-type networks can be successfully applied to solve small or some medium scale TSPs [1]. However, few promising solutions for general medium or large scale TSPs can be obtained. On the other hand, the SOM-like neural networks can handle large scale TSPs with low computation complexity. We will focus on the SOM-like neural networks in this paper.

The SOM-like neural networks, originally proposed by Kohonen, solve the TSP through unsupervised learning [21]. By simply inspecting the data values of the input cities for regularities and patterns, and then adjusting itself to fit the input data through cooperative adaptation of the synaptic weights, such a SOM brings about the localized response to the input data, and thus reflects the topological ordering of the input cities. This neighborhood preserving map then results in a tour of the TSP under consideration. From each city, the resultant tour tries to visit its nearest city. The shortest subtour can intuitively lead to a good tour for the TSP. Such a property learned by the SOM is referred to as *the local optimality* hereafter.

Due to their low computation complexity and promising performance, the SOM-like networks have attracted a large amount of research to explore and enhance the capability of handling the TSP [3], [5], [9], [10], [15]. The solution accuracy of the SOM-like networks are still not comparable to some other state-of-the-art heuristics for the TSP, including ant colony system [7], local search [18], [35], memetic algorithms [23] and simulated annealing [17]. It has been argued that the

Manuscript received Feb. 19, 2002; revised Aug. 14, 2002. The work of H. D. Jin and K. S. Leung was supported in part by RGC Grant CUHK 4212/01E of Hong Kong. The work of M. L. Wong was supported in part by RGC Grant LU 3012/01E of Hong Kong. The work of Z. B. Xu was supported in part by the Hi-Tech R&D (863) Programm (2001AA113182) and by the Nature Science Foundation Project (69975016). The overview of the idea of our approach has been published in the conference paper [16]. This paper was recommended by Associate Editor M. Dorigo.

H. D. Jin and K. S. Leung is with the Department of Computer Science and Engineering, the Chinese University of Hong Kong, Shatin, N.T., Hong Kong (e-mail: hdjin@cse.cuhk.edu.hk; ksleung@cse.cuhk.edu.hk).

M. L. Wong is with the Department of Information Systems, Lingnan University, Tuen Mun, Hong Kong (e-mail: mlwong@ln.edu.hk).

Z. B. Xu is affiliated with the Institute for Information and System Sciences, Faculty of Science, Xi'an Jiaotong University, Xi'an, 710049, China (e-mail: zbxu@mail.xjtu.edu.cn).

Digital Object Identifier 10.1109/TSMCB.2002.804367

TSP is perhaps not the best benchmark by which to judge the effectiveness of the neural networks for optimization [30]. However, we do not think that it is a right time to draw this conclusion. The heuristics usually have a much higher computation complexity than the SOM-like networks. Furthermore, improvements of neural networks for the TSP are being made [2], [4], [13], [26], [37]. Broadly speaking, there are three main streams to enhance the original SOM.

1) Introducing the variable structure network. Instead of the static structure, the output neurons may be dynamically deleted/inserted. Some typical examples are the SOM-like network with a dynamic structure [3], and the FLEXMAP algorithm with a growing structure [10].
2) Amending the competition criterion. Burke and Damany [5] have developed the guilty net by introducing a bias term into the original competition criterion for inhibiting the too-often-winning neurons. In the work of Favata and Walker [9], the competition criterion is based on the inner product, which is slightly different from the Euclidean distance when all weights are normalized.
3) Enhancing the learning rule. The learning rule in the elastic net, proposed by Durbin and Willshaw [8], is often used to enhance the SOMs [2], [15]. Recently, an expanding learning rule has been developed [26]. It is designed to reflect the global optimality, *the convex hull property* of the TSP, to some degrees. Using this rule, the expanding SOM (ESOM) may simultaneously achieve the global optimality and the local optimality. Thus the ESOM can generate shorter tours than several previous SOMs including the convex elastic net (CEN) [2], and the SOM developed by Budinich [4].

In this paper, we develop a SOM-like neural network called the Integrated SOM (ISOM). It uses a new learning rule to integrate the above learning mechanisms. This learning rule is based on the observation that all the previous learning rules can improve the performance of the SOMs from different viewpoints and therefore, can supplement one another. In a learning step, this new learning rule first follows the traditional SOM learning rule to drag the excited neuron toward the input city [21]. This helps the ISOM to learn the neighborhood preserving map. Secondly, the excited neuron is pushed to the convex hull of the TSP. The pushing force, specified according to the convex hull property, helps the ISOM to find tours with the global property. Finally, the excited neuron is drawn toward the middle point of its two neighboring neurons according to the learning mechanism in the elastic net. This mechanism aims to prevent tour intersections and to keep the length of the ring of the neurons as short as possible [8]. However, it is very difficult to design the ISOM that coordinates the three learning mechanisms effectively with the traditional trial and error approach. Thus, a genetic algorithm (GA) is customized to optimize their elaborate coordination. The evolved ISOM (eISOM) obtained by the GA is then tested on a wide spectrum of TSPs to demonstrate its superior performance.

The rest of the paper is organized as follows. We present the ISOM for the TSP and some possible realizations of its learning rule in the next section. We discuss the evolutionary design of an efficient ISOM in Section III. The implementation of a neural-evolutionary system that evolves a promising ISOM is given in Section IV, followed by the eISOM and its performance on three sets of TSPs. We conclude the paper in the last section.

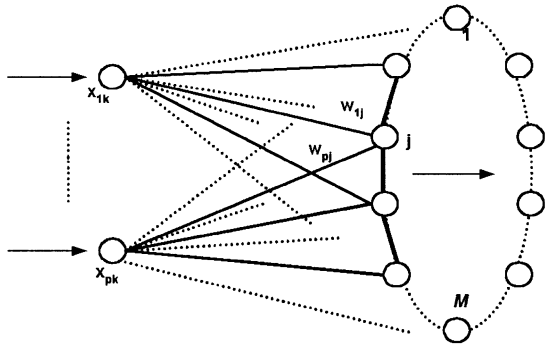


Fig. 1. Schematic SOM-like network for the TSP. M is the number of output neurons. p is the number of input neurons, say 2, for the 2-dimensional Euclidean TSP in this paper.

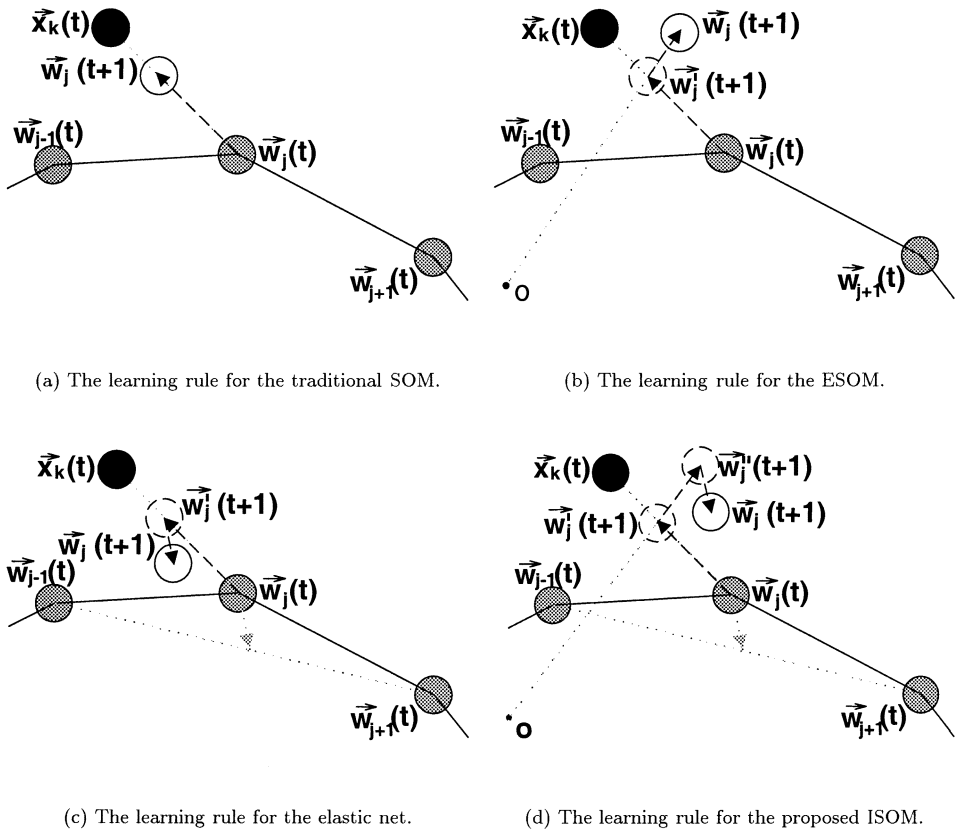
II. Integrated SOM for the TSP

Firstly, we give a brief description of the SOMs for the TSP and outline several typical techniques involved. These pave the way for our Integrated SOM (ISOM).

A. SOMs for the TSP

A SOM-like neural network is an unsupervised competitive learning scheme which simply inspects input data for regularities and patterns, and then organizes itself in such a way as to form a topologically ordered description. This ordered description leads to a solution of the problem under consideration. The SOM-like networks can be applied for many different purposes and in different ways, such as, cluster analysis [32] and data mining [22]. Through viewing a tour of a TSP as a particularly organized, topologically ordered path, SOM-like networks can also be successfully used to handle the TSP [2], [4], [9], and [31].

Fig. 1 shows a schematic view of a SOM-like network for the TSP. A ring of output neurons, denoted by $1, 2, \ldots, M$, is used to characterize a feature map, where M is the number of output neurons. The input neurons, receiving the data of the input city (say, coordinate values), are fully connected to every output neuron. The state of input neurons at time t is indicated by the vector $\vec{x}_k(t) = [x_{1k}(t), x_{2k}(t), \ldots, x_{pk}(t)]^T \in \Re^p$, where p is the number of input neurons. In this paper, we mainly consider the Euclidean TSP in a two-dimensional (2–D) space. That is, p is equal to 2. The synaptic weights between the jth output neuron and each of the input neurons form the vector $\vec{w}_j(t) = [w_{1j}(t), w_{2j}(t), \ldots, w_{pj}(t)]^T \in \Re^p$ $(1 \leq j \leq M)$. Therefore, these output neurons have two topological spaces. One lies on the ring of the output neurons to reflect a linear order of visiting the cities. The other one lies in the p-dimensional space where the coordinate of each output neuron is indicated by its synaptic weight vector. The underlying idea of the SOMs is to construct a topology-preserving map from the high-dimensional synaptic weight space onto the one-dimensional ring space and then form a tour.



(a) The learning rule for the traditional SOM.

(b) The learning rule for the ESOM.

(c) The learning rule for the elastic net.

(d) The learning rule for the proposed ISOM.

Fig. 2. Schematic view of four different learning rules. A black disc indicates an input city; a gray disc indicates a neuron; a solid line indicates the neuron ring; a circle indicates the new position of a neuron; a dashed circle indicates a neuron's interim position; a dashed arrow indicates a movement direction; a gray arrow indicates the elastic force which draws a neuron to the middle point of two neighboring neurons; and "o" indicates the origin, i.e., the center of all cities.

The overall procedure of applying a SOM-like network to a TSP can be divided into three steps, namely, the initialization step, the feature map formation step, and the solution mapping step. In the initialization step, the synaptic weights $\vec{w}_j(0)$ are initialized. Usually, they are initialized with random values. In the feature map formation step, the synaptic weight vectors of the network are modified by unsupervised learning to represent the topological properties of all cities. In the solution mapping step, a tour of the TSP is formed by examining the ordering of the associated output neurons on the ring. Since the solution mapping and the initialization steps are performed routinely, the feature map formation step is crucial to the whole procedure [22].

The feature map formation step aims to construct a perfect neighborhood preserving map in the output neurons. That means, the output neurons that are close on the ring space should be closely located on the synaptic weight space. It is accomplished by performing unsupervised learning on the data values of cities circularly. In other words, the coordinates of cities are fed into the input neurons iteratively in a random fashion. Then the output neurons compete with one another according to *a discriminant function,* for example, the Euclidean metric. After that, the excited neurons (the winning neuron, as well as its neighbors) update their synaptic weights according to a learning rule. The learning process continues until all cities are fed into the network for certain times.

Now we are going to discuss different learning rules which aim to learn the regularities and patterns of the TSP from the data of cities iteratively. The most commonly used learning rule is

$$\vec{w}_j(t+1) = \vec{w}_j(t) + \alpha_j(t)\,(\vec{x}_k(t) - \vec{w}_j(t)) \tag{1}$$

where $\alpha_j(t)$ is *a learning rate* of the network, ranging from 0 to 1 [21]. Intuitively, the learning rule means that the excited neuron $\vec{w}_j(t)$ will approach the input $\vec{x}_k(t)$ with the movement quantity proportional to the distance between the input city and the excited neuron. Fig. 2(a) illustrates this learning mechanism. Here the black disc indicates the input city, a gray disc indicates an output neuron, the solid line connecting the neurons indicates the neuron ring, and the circle represents the new position of the excited neuron. Based on the above learning rule, a SOM-like network works well on the TSP because it constructs a neighborhood preserving map. Thus, from each city, the generated tour tries to visit its nearest neighbor as far as possible. These shortest subtours hopefully lead to an optimal tour, however, the local optimality does not always appear in the optimal tours of all TSPs.

In order to improve the performance, our recently proposed algorithm, the expanding SOM (ESOM), takes into account global optimality [26]. In contrast with the local optimality, the global optimality is valid to the optimal tours of all TSPs. *The convex hull property* is an example. The convex hull for a TSP is the largest convex polygon whose vertices are cities of the TSP. The convex hull property says that, for any optimal tour of the TSP, the cities located on the convex hull must be

visited in the same sequence as they appear on the convex hull. The ESOM can generate such tours that visit the cities on the convex hull in the same sequence as they appear on the convex hull. It achieves this by embodying the convex hull property in its learning rule

$$\begin{aligned}\vec{w}_j(t+1) &= c_j(t)\vec{w}'_j(t+1)\\ &= c_j(t)\left\{\vec{w}_j(t)+\alpha_j(t)\left(\vec{x}_k(t)-\vec{w}_j(t)\right)\right\}. \quad (2)\end{aligned}$$

where the interim neuron $\vec{w}'_j(t+1)$ indicates the position after the excited neuron $\vec{w}_j(t)$ moves toward the input city $\vec{x}_k(t)$ and $c_j(t)$ is *an expanding coefficient* which reflects the convex hull, given as

$$c_j(t) = \left[1-2\alpha_j(t)(1-\alpha_j(t))\kappa_j(t)\right]^{-1/2} \quad (3)$$

and

$$\begin{aligned}\kappa_j(t) = 1 &- \langle \vec{x}_k(t),\, \vec{w}_j(t)\rangle\\ &-\sqrt{\left(1-\|\vec{x}_k(t)\|^2\right)\left(1-\|\vec{w}_j(t)\|^2\right)}. \quad (4)\end{aligned}$$

The formula in (3) is valid since all cities and output neurons are restricted within the unit circle in the ESOM. The expanding coefficient $c_j(t)$, normally larger than 1, pushes the excited neuron away from the origin. This functionality is illustrated in Fig. 2(b). After reaching the interim neuron $\vec{w}'_j(t+1)$, the excited neuron $\vec{w}_j(t)$ is further pushed away from the origin "o" and reaches a new position $\vec{w}_j(t+1)$. With repeated input of cities, the neuron ring expands since the center of all cities laps over the origin. The expansion can also be viewed as pushing neurons to the convex hull of the TSP. According to (3), the expanding coefficient increases with the distance of the input city from the origin. In other words, the closer the convex hull to the city, the more expanding force the excited neuron gets. Together with the neighborhood preserving property, this expansion helps the ESOM to generate such tours that visit the cities on the convex hull in the same sequence as the cities appear. Thus, the ESOM achieves both the convex hull property and the local optimality [26]. It will be seen in Section IV that the ESOM generates significantly shorter tours than the classical SOMs.

Another renowned learning rule which is adopted by the elastic net [8] is

$$\begin{aligned}\vec{w}_j(t+1) = \vec{w}_j(t) &+ \alpha_j(t)\left[\vec{x}_k(t)-\vec{w}_j(t)\right]\\ &+\frac{\beta_j(t)}{2}\left[\vec{w}_{j-1}(t)+\vec{w}_{j+1}(t)-2\vec{w}_j(t)\right] \quad (5)\end{aligned}$$

where $\beta_j(t)$ is another learning rate parameter. The last term in the right-hand side of (5) attracts the excited neuron to the middle point of its two neighboring neurons on the ring, as illustrated in Fig. 2(c). It reflects the elastic force constraint, as indicated by the gray arrow in the figure, and reduces the length of the resultant ring of neurons as far as possible [8]. Moreover, it has been empirically confirmed that this learning rule can inhibit intersections in the resultant tours [15].

B. Integrated SOM for the TSP

The above three learning rules have been used successfully in handling TSPs. Their underlying ideas emphasize different aspects of the TSP. Thus, we propose to integrate these ideas together and develop a novel integrated SOM (ISOM) to take advantage of these three learning mechanisms. For example, the new ISOM can employ the expanding mechanism to achieve the convex hull property, and explore the possible efficient interaction between the input city and the excited neuron. It can also use the elastic force constraint to inhibit intersections during learning. We present the ISOM below.

ISOM for the TSP:

1) Transform the coordinates $[x'_{1i},\, x'_{2i}]^T$ $(i=1,\ldots,n)$ of all cities such that they lie within a circle centered at the origin with radius R (≤ 1). Here n is the number of the cities. Hereafter, $[x_{1i},\, x_{2i}]^T$ denotes the new coordinate of $\vec{x}_i$.
2) Set $t=0$, $p=2$, and the initial weight vectors $\vec{w}_j(0)$ $(j=1,\ldots,n,$ thus $M=n)$ with random values within the circle above.
3) Select a city at random, say $\vec{x}_k(t)=[x_{1k}(t),\, x_{2k}(t)]^T$, and feed it to the input neurons.
4) Find the winning output neuron, say $m(t)$, nearest to $\vec{x}_k(t)$ according to the Euclidean metric

$$\begin{aligned}m(t) &= \arg\min_j d\left(\vec{x}_k(t),\, \vec{w}_j(t)\right)\\ &= \arg\min_j \left\|\vec{x}_k(t)-\vec{w}_j(t)\right\|^2. \quad (6)\end{aligned}$$

5) Train neuron $m(t)$ and its neighbors within *the effective width* $\sigma(t)$ by using the following:

$$\begin{aligned}\vec{w}_j(t+1) = c_j(t)\times &\left\{\vec{w}_j(t)+\alpha_j(t)\left[\vec{x}_k(t)-\vec{w}_j(t)\right]\right\}\\ &+\frac{\beta_j(t)}{2}\left[\vec{w}_{j-1}(t)+\vec{w}_{j+1}(t)-2\vec{w}_j(t)\right] \quad (7)\end{aligned}$$

where $j=m(t),\, m(t)\pm 1,\ldots,m(t)\pm\sigma(t)$. $c_j(t)$ is *the expanding coefficient* which will be discussed later. *The learning rates* $\alpha_j(t)$ and $\beta_j(t)$ are specified by

$$\alpha_j(t) = \eta_1(t)\times h_{j,\,m(t)} \quad (8)$$

$$\beta_j(t) = \eta_2(t)\times h_{j,\,m(t)} \quad (9)$$

and

$$h_{j,\,m(t)} = \begin{cases} 1-\dfrac{d_{j,\,m(t)}}{\sigma(t)+1}, & d_{j,\,m(t)} \le \sigma(t)\\ 0, & \text{otherwise.}\end{cases} \quad (10)$$

Here $\eta_1(t)$ and $\eta_2(t)$ are *the learning parameters* of the network, $h_{j,\,m(t)}$ is a *neighborhood function*, and $d_{j,\,m(t)}=\mathrm{MOD}(|j-m(t)|,\, n)$ is the distance between the neurons $m(t)$ and j on the ring.
6) Update the effective width $\sigma(t)$, and the learning parameters $\eta_1(t)$ and $\eta_2(t)$ with predetermined decreasing schemes. If a predetermined number of loops have not been executed, go to Step 3 with $t:=t+1$.
7) Calculate the activity value of each city $\vec{x}_k$ according to

$$\begin{aligned}a\left(\vec{x}_k\right) = m_k - \tfrac{3}{26}\Bigg\{&d\left(\vec{x}_k,\, \vec{w}_{m_k}\right)\\ &+\frac{2\left[d\left(\vec{x}_k,\, \vec{w}_{m_k+1}\right)-d\left(\vec{x}_k,\, \vec{w}_{m_k-1}\right)\right]}{3}\Bigg\} \quad (11)\end{aligned}$$

where m_k is the winning neuron associated with $\vec{x}_k$.
8) Order the cities by their activity values, and then form a tour of the TSP.

Steps 7 and 8 implement the solution mapping procedure of a SOM-like network for the TSP. They aim to yield a tour from the topologically ordered neurons. Eq. (11) maps each city to a floating point number which solves the confusion that multiple cities are mapped onto the same neuron. Furthermore, the mapping method exploits the information of the winning neuron and its nearest neurons. If more than one city excites the same neuron, the city that is closer to the preceding neuron on the ring and farther away from the subsequent neuron will be visited earlier in the tour. It can make the length of the resultant tour as short as possible.

Step 1 is a linear transformation. It moves the center of all cities to the origin, and restricts all cities within a circle with radius R. This transformation does not influence the solution space of the TSP. This step mainly facilitates the implementation of the expanding coefficient $c_j(t)$ and makes it possible to reflect the convex hull based only on the input city and the excited neuron. After this linear transformation, the distance between the city and the origin, namely the norm of the input vector, is proportional to the distance between the original city and the center of all cities. Thus, the norm of the input city can be used to reflect the location of the city with respect to all cities. If the norm is larger, the city is more likely to locate on the convex hull. We can then formulate the expanding coefficient $c_j(t)$ to reflect the convex hull property in such a way that $c_j(t)$ increases with the norms of input city and the excited neuron. Furthermore, since the input city and the excited neuron are within the unit circle and they are close to each other, their inner product is close to their norms. That means, the inner product may also be used to reflect the relative locations of the input city and the excited neuron. Thus, using the norms and the inner products, we can design some reasonable expanding coefficients to reflect the convex hull.

The learning rule in (7) is the key point of the proposed ISOM. It also distinguishes the ISOM from all previous SOMs. The learning rule is illustrated in Fig. 2(d). First, the excited neuron is dragged toward the input city. This adaptation is indicated by the terms enclosed in curved brackets. These terms are the same as those at the right-hand side of (1). In Fig. 2(d), the neuron reaches $\vec{w}'_j(t+1)$, which behaves like the adaptation in Fig. 2(a). Secondly, the excited neuron is pushed away from the origin, as specified by the expanding coefficient $c_j(t)(\geq 1)$. As shown in Fig. 2(d), the neuron moves from $\vec{w}'_j(t+1)$ to $\vec{w}''_j(t+1)$. This adaptation, similar to that in Fig. 2(b), may be viewed as pushing the neuron to the convex hull of the TSP since the convex hull surrounds the origin. Thus, it helps the ISOM to make tours visit the cities on the convex hull in the same sequence as these cities appear on the convex hull. Finally, the excited neuron is drawn by the elastic force as indicated by the last term in (7). In Fig. 2(d), the neuron moves from $\vec{w}''_j(t+1)$ to $\vec{w}_j(t+1)$. This adaptation is similar to the one in Fig. 2(c). It is clear that the rule embodies the three learning mechanisms.

C. Expanding Coefficient

We turn to formulate the expanding coefficient $c_j(t)$ in detail. To clarify the procedure, we express the expanding coefficient $c_j(t)$ with several parts that have respective functionalities. It is formulated as

$$c_j(t) = [1.0 + b_j(t) \times e_j(t)]^{a_4}. \tag{12}$$

The constant 1.0 ensures that the expanding coefficient is close to 1.0 so as to make our learning rule harmonize well with the one in (1). The constant a_4 unifies the expanding coefficient $c_j(t)$ with the one of the ESOM in (3). The term $b_j(t)$ is used to adjust the relative strength of the expanding force with respect to the learning rate $\alpha_j(t)$. In other words, it harmonizes the expanding force and the dragging force. To unify the expanding coefficient $c_j(t)$ in (12) with the one of the ESOM network in (3), $b_j(t)$ is formulated as

$$b_j(t) = a_1 \times \alpha_j(t)^{a_2} \times (1 - \alpha_j(t))^{a_3} \tag{13}$$

where parameters a_i $(i = 1, 2, 3)$ are positive numbers. The term $e_j(t)$ in (12) reflects the convex hull in terms of the vector properties of the input city and the excited neuron. As analyzed above, the norm of a city (or neuron) in the ISOM can reflect the location of the city (or neuron) with respect to other cities. The larger its norm is, the more likely the city (or neuron) locates on the convex hull. On the other hand, since both the city and the neuron are restricted within the circle with radius R, their inner product will approach their norms. Thus, the inner product can be used in the expanding coefficient to differentiate the roles of different cities, too. Together with the neighborhood preserving property, this expansion helps the ISOM to generate tours which visit the cities on the convex hull in the same sequence as the cities appear on the convex hull. In other words, the ISOM can reach the global optimality. Consequently, the ISOM may generate better tours. The norm of city (or neuron) and the inner product can be used to form a lot of implementations of the term $e_j(t)$. We list some implementations used in our experiments.

1) $e_j(t) = \|\vec{w}'_j(t+1)\|^2 - |\langle \vec{x}_k(t), \vec{w}_j(t)\rangle|$ is the difference between the distance of the interim neuron $\vec{w}'_j(t+1)$ from the origin and the absolute value of the inner product of the input city $\vec{x}_k(t)$ and the excited neuron $\vec{w}_j(t)$.
2) $e_j(t) = \|\vec{w}'_j(t+1)\|^2 + \|\vec{x}_k(t) - \vec{w}_j(t)\|^2$ is the sum of the distance of the interim neuron $\vec{w}'_j(t+1)$ from the origin and the distance of the city $\vec{x}_k(t)$ from the neuron $\vec{w}_j(t)$.
3) $e_j(t) = \|\vec{x}_k(t) - \vec{w}_j(t)\|^2 \times \|\vec{x}_k(t)\|^2$ is the product of the distance of the city $\vec{x}_k(t)$ from the origin and the distance of the city $\vec{x}_k(t)$ from the neuron $\vec{w}_j(t)$.
4) $e_j(t) = \|\vec{w}_j(t)\|^2 - \langle \vec{x}_k(t), \vec{w}_j(t)\rangle$ is the difference between the distance of the neuron $\vec{w}_j(t)$ from the origin and the inner product of the city $\vec{x}_k(t)$ and the neuron $\vec{w}_j(t)$.
5) $e_j(t) = \|\vec{x}_k(t)\|^2 - \langle \vec{w}_j(t), \vec{x}_k(t)\rangle$ is the difference between the distance of the city $\vec{x}_k(t)$ from the origin and the inner product of the city $\vec{x}_k(t)$ and the neuron $\vec{w}_j(t)$.

Since the proposed ISOM integrates three learning mechanisms together, an efficient ISOM must have good coordination among the local optimality of the traditional SOM, the global optimality of the expanding coefficient, and the elastic force constraint. Moreover, a suitable implementation of the expanding coefficient and parameter settings should be determined. It seems very difficult to specify a good ISOM manually.

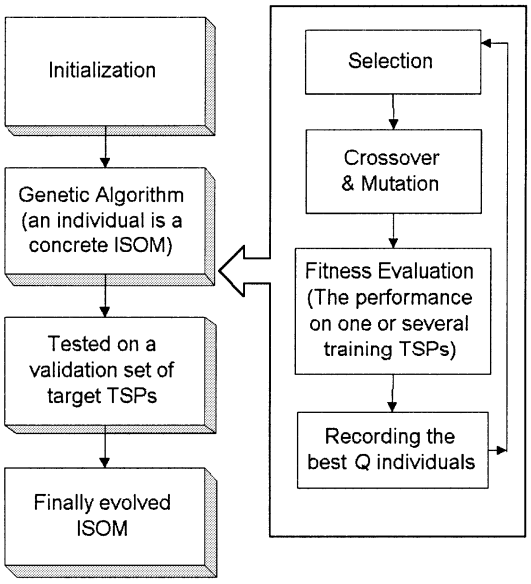


Fig. 3. Neural-evolutionary system that evolves learning schemes.

Thus, we employ a genetic algorithm in this paper to design an efficient ISOM.

III. Evolutionary Design of the ISOM

During the past two decades there has been growing interest in evolutionary algorithms, especially genetic algorithms (GAs). They are a family of global stochastic search algorithms based on Darwin's theory of evolution (survival of the fittest) [11], [36]. Normally, these algorithms maintain a population of chromosomes, and manipulate them by using several genetic operators. A schematic view of a GA is given in the right-hand side of Fig. 3. The most significant advantages of using evolutionary algorithms lie in the gain of flexibility and adaptability to the task on hand, in combination with robust performance and global search characteristics. Thus they have been employed to handle many inherently hard problems. As mentioned in [32], [36], and [38], the search space of all possible network structures (size and connectivity), and learning rules is infinitely large, undifferentiable, deceptive, and multimodal. Evolutionary algorithms provide promising and automatic alternatives to solve the difficult problem of designing neural networks. Moreover, the combination of learning and the evolutionary mechanisms can significantly improve the trainability, productivity, and problem-solving capability of *the neural-evolutionary systems.*

In a neural-evolutionary system, evolution and learning are the two fundamental adaptation mechanisms. Evolution can be introduced into neural networks on various levels. It can be used to evolve the termination criteria [32], weights [38], architecture [6], [24], [28], and learning rules [34]. It is hard to say which one is on a higher level [38].

Since the architecture of the SOMs for the TSP has been well-studied, we concentrate on evolving a learning scheme that consists of a learning rule, the parameter setting of the learning rule, and the learning parameter setting. Once the architecture of a network is known, a learning scheme can generate a concrete neural network algorithm. The algorithm can be used to handle several TSPs and the evolutionary algorithm will not be executed again. Our approach is different from the one that evolves the architecture and learns the synaptic weights together. In the latter approach, both evolution and learning are employed for each target problem.

For the problem of evolving a good learning scheme, the performance of a candidate learning scheme acts as its fitness. The performance is estimated by considering the speed, the accuracy, and the generalization capability of the learning scheme. In order to obtain an accurate estimation of the fitness value, a large number of different target problems should be solved by using the scheme. Obviously, this fitness evaluation procedure will take a long time to evaluate a scheme. Furthermore, the fitness evaluation procedure must be executed for different schemes during the evolution process. Thus, it will take extremely long to evolve a learning scheme if an accurate fitness evaluation procedure is used.

In order to handle this problem, the fitness value of a learning scheme may be estimated by applying the learning scheme to one or a few small-scale target problems. However, this approach will introduce the noisy fitness evaluation problem because the fitness value of a scheme relies on the selected target problems. To alleviate the noisy fitness evaluation problem, a learning scheme is examined on the target problems for several runs and our fitness function considers the average performance and the variance of performance among these runs.

Since we cannot ensure that the learning scheme with the best fitness value on a few small-scale target problems also performs well on all target problems, we introduce a validation set of target problems to verify the generalization capability of the learning scheme. It is expected that a learning scheme with good performance on the validation set of problems will also perform well on other problems.

A neural-evolutionary system that evolves a learning scheme is shown in Fig. 3. After initialization, a genetic algorithm is used to evolve good learning schemes. A concrete neural network algorithm is obtained from a learning scheme. The neural network algorithm is then used to solve a number of small-scale target problems in order to estimate the fitness value of the corresponding learning scheme. The best Q different schemes during the evolution process are stored. The evolution process iterates until the termination criterion is satisfied and the stored learning schemes are verified on a set of large-scale problems. Finally, the learning scheme with the best fitness on the validation problems is returned as an evolved ISOM.

IV. Implementation and Results

We have implemented the above neural-evolutionary system to evolve an efficient ISOM for the TSP. All algorithms are implemented in C++ and all experiments are performed on a Sun Ultrasparc 5/270 workstation.

A. Evolving the ISOM

We have used the canonical GA in the neural-evolutionary system. In our GA, every individual (chromosome) represents a

learning scheme. For each learning scheme, the parameters include the type of formula to calculate the expanding coefficient and the parameters a_i ($i = 1, \ldots, 4$). They also include other parameters in the ISOM, such as the radius R, the total learning loop L, the initial values, and the decreasing schemes of the effective learning width $\sigma(t)$, and the learning parameters $\eta_1(t)$ and $\eta_2(t)$. Since these 13 parameters include both integers and floating point numbers. For simplicity, each allele (gene) represents a parameter in an individual. Thus, given the ISOM discussed in Section II, an individual determines a concrete ISOM. These 13 alleles and their domains are listed in Table I. Our GA ensures that all alleles stay in their respective domains, thus invalid individuals will never be generated.

We use the relative difference between the generated tour length and the optimal tour length to measure *solution quality*, i.e., (length$_{generated}$−length$_{optimal}$)/length$_{optimal}$. The fitness function is designed to indicate the average solution quality as well as its consistency on several runs. It is formulated as

$$fitness = 3 - \text{ave}(quality) - \text{var}(quality) \qquad (14)$$

where $\text{ave}(quality)$ and $\text{var}(quality)$ are the average solution quality and its variance, respectively.

In the fitness function, 3 is a constant used to keep the fitness value positive. Thus, the roulette wheel selection method can directly be applied based on these fitness values for the GA maximization. If the optimal tour length is unknown, we use a theoretical lower bound in place of the optimal tour length. The theoretical lower bound says that the shortest tour length for a random TSP with n cities within the unit square is close to 0.765 $\times \sqrt{n}$ [12].

The roulette wheel selection method is used to select parents in each generation [11]. The mutation operator modifies the old allele value to a random value in the domain. Two crossover operators are applied alternatively. The first one is the widely used one-point crossover operator that exchanges the alleles after a randomly selected crossover point in the two parents [11]. The second considers each allele in turn and generates a random value that is close to the two corresponding allele values in the parents. For example, if the allele values of the parents, respectively, are z_1 and z_2, the allele value z_o of the offspring will be $z_2 + \lambda(z_1 - z_2)$, where λ is a random value in [0, 1]. This operator is commonly used in evolution strategies. If the value z_o is out of the domain, it will be changed to its closest valid value.

In our neural-evolutionary system, the crossover and the mutation probabilities are 0.99 and 0.01, respectively. The population size is 100, and the maximum number of generations is 6000. The fitness value is evaluated on three runs of two random TSPs with 30 and 50 cities, respectively. During the evolution process, the best 30 individuals are stored. The stored individuals are then evaluated on three runs of three random TSPs with 200, 800, 1800 cities, respectively. All random TSPs used in this paper can be found at http://www.cse.cuhk.edu.hk/~hdjin/som/.

The learning scheme evolved is listed in the last column of Table I. The executable program and the source codes of the evolved ISOM (eISOM) can also be downloaded at the above web site. The explanation of the evolved learning scheme is given as follows:

TABLE I
13 ALLELES OF AN INDIVIDUAL (CHROMOSOME) AND THEIR DOMAINS IN THE NEURAL-EVOLUTIONARY SYSTEM, AND THE PARAMETER SETTING IN THE EISOM. THE LEARNING PARAMETERS $\eta_1(t)$, $\eta_2(t)$, AND $\sigma(t)$ ARE DECREASED LINEARLY AFTER EACH LEARNING ITERATION. $\eta_1(t)$ REACHES ZERO AT THE LAST LOOP

Alleles in an individual	**Domains**	**eISOM**
Formula for $e_j(t)$	$\{1, 2, \cdots, 20\}$	1
Parameters for $e_j(t)$:		
a_1, a_2 and a_3	{0,0.25,$\cdots$, 5}	1,3,0.25
a_4	{0.2,0.4,$\cdots$, 5}	1.0
Radius R	(0.001,1.0)	0.61
Learning loop L	{50,65,$\cdots$, 200}	160
Learning parameter $\eta_1(0)$	(0.001,1.0)	0.95
Learning parameter $\eta_2(0)$	(0.001,1.0)	0.12
$\eta_2(t)$ decreasing mode (be 0 after p_1 percent iterations)	p_1: (0,100)	48
The effective width $\sigma(0) = a + b \cdot n$	a: {1,2,$\cdots$, 14}	10
	b: [0.001,0.6]	0.01
$\sigma(t)$ decreasing mode (be 1 after p_2 percent iterations)	p_2: (0,100)	62

- *The expanding coefficient* $c_j(t)$ is

$$c_j(t) = 1 + \alpha_j(t)^3(1 - \alpha_j(t))^{0.25}\left\{\sum_{i=1}^{2}[\alpha_j(t)x_{ik}(t) + (1 - \alpha_j(t))w_{ij}(t)]^2 - \left|\sum_{i=1}^{2} x_{ik}(t)w_{ij}(t)\right|\right\}. \qquad (15)$$

That means, $e_j(t)$ in (12) is calculated using the first formula listed.

- *Radius* R is 0.61.
- *Learning loop* L is set to be 160. Thus, each city is circularly fed into the ISOM 160 times. Namely, there are totally $160 \times n$ learning iterations.
- *The learning parameter* $\eta_1(t)$ is initialized to 0.95 and is decreased linearly for each learning loop until it reaches zero at the last loop.
- *The learning parameter* $\eta_2(t)$ is initialized to 0.12 and is decreased linearly to 0 in the first 48% learning iterations.
- *The effective width* $\sigma(t)$ is initialized to $10 + 0.01n$, and is decreased linearly to 1 in the first 62% iterations. It keeps 1 in the remaining 38% iterations.

The expanding coefficient $c_j(t)$ in (15) only consists of the learning rate $\alpha_j(t)$, the input city $\vec{x}_k(t)$ and the excited neuron $\vec{w}_j(t)$. The expanding term $e_j(t) = \|\vec{w}'_j(t+1)\|^2 - |\langle \vec{w}_j(t), \vec{x}_k(t)\rangle|$ is used to reflect the convex hull. Roughly, the closer to the convex hull the

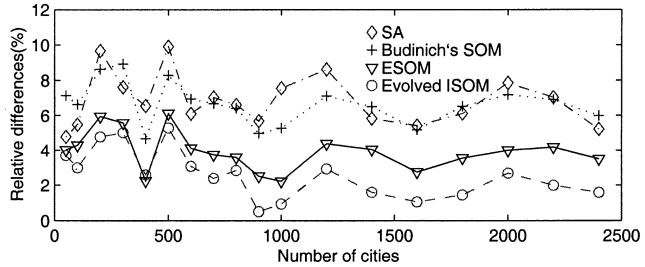
input city (or the excited neuron) is, the larger the expanding term is. That means, the cities on the convex hull have more influence during learning. So, together with the neighborhood preserving property, the expanding coefficient can help the ISOM to generate shorter tours. On the other hand, the expanding coefficient $c_j(t)$ equals 1.0, either when the excited neuron is identical to the input city, or when the learning rate $\alpha_j(t)$ reaches 0. Thus, when the learning rate $\alpha_j(t)$ approaches zero, the expanding coefficient has no influence on learning. In other words, the eISOM has similar asymptotic behavior to the SOM because the learning rates $\alpha_j(t)$, and $\beta_j(t)$ finally approach zero. Therefore, the evolved expanding coefficient is reasonable.

B. Performance of the eISOM

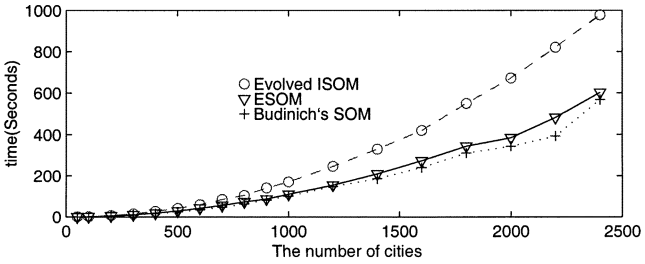
The computation complexity of the eISOM is $O(n^2)$, which is the same as the ESOM [26] and the SOM developed by Budinich [4]. It is worth noting that almost all nonneural network methods, such as simulated annealing (SA) [19], GAs [23], [33] and ant colony systems [7], have much higher complexity. Thus, they take longer time to find tours comparable to those generated by the SOMs.

To evaluate the performance of the eISOM, we have compared it with several typical algorithms, including the SA approach [4], the SOM developed by Budinich [4], the ESOM [26], the convex elastic net (CEN) [2], and the FLEXMAP algorithm [10] on three sets of TSPs. We have chosen these SOM-like algorithms since they are state-of-the-art and have similar computation complexity with the eISOM. For the SA approach, the annealing factor 0.95 is used as in [4]. The exchange method, known as *2-opt* inspired by Lin and Kernigham [27], is adopted. The SA approach allows $20 \times n$ trials at each temperature level. It usually generates better tours than the heuristic *2-opt* [4]. It is worth pointing out that the performance of the SA approach depends on the exchange method used [17], [35]. It can generate shorter tours than our implementation in this paper if *3-opt*, or Lin–Kernigham [27] heuristics are used. However, much more execution time is required. The Budinich's SOM, an effective implementation of the traditional SOM, maps each city onto a linear order without ambiguity. We set all parameters according to Budinich [4], which can generate tours comparable to the SA approach. The ESOM network uses the learning rule in (2), and we implement it using the parameter setting in [26]. The CEN algorithm has to form the convex hull of the given cities explicitly. It then takes the initial tour on the convex hull and trains the network in a similar way as the elastic net, yielding shorter tours of TSPs than the elastic net [2]. We have not implemented CEN due to the lack of its implementation details. The experiment results of CEN below are quoted from [2]. The FLEXMAP algorithm inserts a new neuron in the ring of the neurons every several learning loops. We take its performance results directly from the paper [10].

The first set of experiments were conducted on a set of 18 TSPs with 50 to 2400 cities. These TSPs are all generated randomly within the unit square. Fig. 4(a) shows the experiment results of the eISOM, the ESOM, the Budinich's SOM, and the SA approach. The solution quality is represented in terms of the relative difference between the average tour length and the theoretical lower bound. The results are based on 10 runs.



(a) Comparison of the solution quality of the evolved ISOM, the ESOM, the Budinich's SOM, and the SA approach.



(b) The average execution time comparison among the three SOMs.

Fig. 4. Performance on 18 random TSPs.

From Fig. 4(a), it can be observed that the tours generated by the eISOM are much nearer to the theoretical bounds than those by the SA approach and Budinich's SOM. Except for the tours of the TSP with 400 cities, those generated by the eISOM are shorter than those generated by the ESOM on average. The solution quality of the eISOM varies slightly with the sizes of the TSPs. For example, its tours for the TSP with 2400 cities is about 1.59% longer than the theoretical bound. The ESOM network generates tours 3.50% longer than the theoretical bound, the Budinich's SOM generates tours 5.99% longer than the theoretical bound, and the SA approach generates tours 5.24% longer than the theoretical bound. The eISOM performs substantially better than its three counterparts. Fig. 5 depicts the typical tours generated by these four algorithms. It is interesting to point out that the tour generated by the ESOM in Fig. 5(c) visits the cities on the convex hull in sequence as specified by the convex hull property of an optimal tour of the TSP. The tour generated by the eISOM also has such property, moreover, the tour has no intersections, as illustrated in Fig. 5(d). However, the three tours, generated by its counterparts, all intersect themselves.

We have averaged all the experiment results on these 18 TSPs. The average relative difference is 2.63% for the eISOM. In other words, the tours obtained are 2.63% longer than the theoretic lower bounds on average. The average relative differences are 3.93% for the ESOM, 6.65% for the Budinich's SOM, and 6.83% for the SA approach respectively. Consequently, the ISOM makes 1.30% improvement over the ESOM, and makes around 4% improvement over the Budinich's SOM and the SA approach.

The execution time of the three SOMs is illustrated in Fig. 4(b). The execution time increases similarly with the sizes

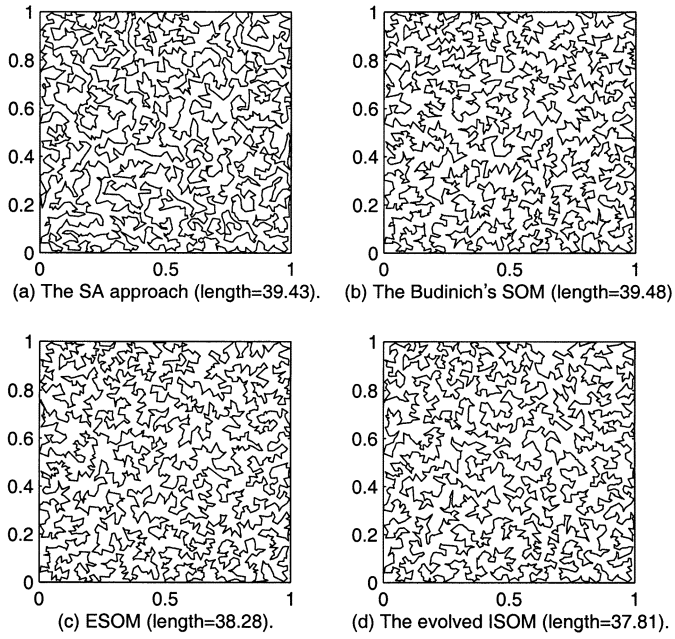


Fig. 5. Four typical tours of the random TSP with 2400 cities obtained by (a) the SA approach. (b) Budinich's SOM. (c) ESOM. (d) Evolved ISOM.

of TSPs. The execution time of the eISOM is about 1.6 times that of the ESOM and the Budinich's SOM for each TSP. This is mainly due to the fact that the eISOM has more learning loops to execute. It should be emphasized that three SOMs are much faster than the SA approach. For example, the SA approach spends about 5400 s to generate a solution of the TSP with 2400 cities. However, the eISOM spends about 1000 s, and the other networks spends about 600 s.

The second set of experiments were mainly designed to compare the eISOM with the convex elastic net (CEN) of Al-Mulhem and Al-Maghrabi [2]. We also present the experiment results of the SA approach, the Budinich's SOM, the ESOM, and their enhanced versions. An enhanced version is that a network is improved by the *NII* heuristic. The *NII* heuristic, used in [2], improves tours by using a rearrangement heuristic derived from *2-opt*. Since the experiment results of the CEN algorithm quoted from [2] have been enhanced by the *NII* heuristic, the other SOMs have also been enhanced to make a fair comparison. We tested the algorithms on five TSP benchmarks examined by the CEN algorithm. The five TSPs can be taken from the TSPLIB, collected by Reinelt [29].

Table II lists the experiment results of the original and enhanced SOMs. The results are based on 10 independent runs and are presented in terms of the relative differences between the average tour lengths and the optimal tour lengths. It can be observed from Table II.[1] that the eISOM always yields better solutions than the SA approach, the Budinich's SOM, and the ESOM. On average, the eISOM makes 0.77% improvement over the ESOM, and makes about 3.5% improvement over the Budinich's SOM and the SA approach. These results accord with the first set of experiments. The enhanced eISOM obtains shorter tours than other enhanced neural networks for all problems except the GR96 problem. On average, the enhanced eISOM generates tours 1.20% longer than the optima. The enhanced CEN algorithm generates tours 2.95% longer than the optima. The enhanced ESOM, and the Budinich's SOM generate tours 1.50% and 2.07% longer than the optima respectively. The enhanced eISOM performs better than three other algorithms. Since the computation complexity of CEN is also $O(n^2)$, we conclude that the enhanced eISOM substantially outperforms the enhanced CEN algorithm.

The third set of experiments were performed to compare the eISOM with the FLEXMAP algorithm [10]. The experiment results are listed in Table III.[2] All 10 TSPs can be found in the TSPLIB [29]. The listed results for each TSP are the relative differences between the best tour length and the corresponding optimum. They are based on 20 runs for each problem. The results of the FLEXMAP algorithm is quoted from [10]. An enhanced version means that the algorithm is improved by the local improvement heuristic used by the FLEXMAP algorithm [10]. This heuristic computes all 24 permutations of every subtour with four cities and employs the shortest permutation in order to get a better tour. The experiment results for four enhanced algorithms are listed in the last four columns of Table III.

It can be observed from Table III that the enhanced eISOM generates shorter tours than the FLEXMAP algorithm, the enhanced ESOM, and the Budinich's SOM for all TSPs except the EIL51, the EIL101, and the PCB442 problems. The average relative difference for the enhanced eISOM is 2.72%. The average relative differences are 4.37%, 3.17%, and 4.41% for the enhanced versions of the FLEXMAP algorithm, the ESOM, and the Budinich's SOM respectively. The enhanced eISOM makes 1.65% improvement over the FLEXMAP algorithm. Furthermore, the eISOM performs very well even without being enhanced by the local improvement heuristic. Observed from Table III, the average relative difference for the eISOM is close to that for the enhanced ESOM network. It is smaller than that for the FLEXMAP algorithm. The eISOM makes 1.13% improvement over the FLEXMAP algorithm. The improvement is promising because the tours obtained by the FLEXMAP algorithm are very near to the optima.

In summary, for these three comprehensive sets of experiments, the eISOM can generate about 3% shorter tours than the SA approach with respect to the optima using less execution time. For the average relative differences, it makes at least 1% improvement over the other four SOMs on a wide spectrum of TSPs. To the best of our knowledge, it is one of the most accurate SOMs for the TSP. With the same quadratic computation complexity, the eISOM substantially outperforms the Budinich's SOM, the ESOM, and the CEN. The latter three SOMs are the special cases of the ISOM. They are designed manually while the eISOM are designed by the GA. This point also indicates that the GA helps the eISOM to reach a good coordination among three learning mechanisms.

V. Conclusion

In this paper, we have developed the integrated self-organizing map (ISOM), a new self-organizing map (SOM) for the

[1]The optimal tour of GR96 cited by [2] is as long as 55 209.

[2]The optimal tour length of HT30 is 415 according to [9] and our experiments. And the optimal tour of ATT532 cited by [10] is as long as 27 686.

TABLE II
EXPERIMENT RESULTS OF THE SA APPROACH, THE BUDINICH'S SOM, THE ESOM, AND THE EISOM, AND THE ENHANCED CEN ALGORITHM WHEN APPLIED TO 5 TSP BENCHMARKS. THE BOLD-FACED TEXT INDICATES THE BEST ONE AMONG 4 DIFFERENT ALGORITHMS FOR A TSP. THE ENHANCED ALGORITHMS ARE IMPROVED BY THE *NII* HEURISTIC

TSP name	number of Cities	Optimum	Solution quality of 4 algorithms (%)				Solution quality of enhanced SOMs (%)			
			SA	Budinich	ESOM	eISOM	CEN	Budinich	ESOM	eISOM
GR96	96	51231	4.12	2.09	1.03	**0.81**	4.39	**0.46**	**0.46**	0.53
GRID100	100	100	2.07	2.17	**0.83**	**0.83**	**0.80**	1.63	**0.80**	**0.80**
KROA100	100	21282	5.94	3.68	1.01	**0.57**	1.60	0.93	0.81	**0.54**
GR137	137	69853	8.45	8.61	4.27	**3.16**	3.29	4.51	2.52	**2.18**
LIN318	318	44169	7.56	8.19	4.11	**2.05**	4.67	2.81	2.89	**1.96**
Average			5.63	4.95	2.25	**1.48**	2.95	2.07	1.50	**1.20**

TABLE III
EXPERIMENT RESULTS OF THE SA APPROACH, THE FLEXMAP ALGORITHM, THE BUDINICH'S SOM, THE ESOM, AND THE EISOM WHEN APPLIED TO THE THIRD SET OF TSPS. THE BOLD-FACED TEXT INDICATES THE BEST SOLUTION AMONG 4 DIFFERENT ALGORITHMS FOR A TSP. THE ENHANCED ALGORITHMS ARE IMPROVED BY THE LOCAL IMPROVEMENT HEURISTIC

TSP name	Number of cities	Optimum	Solution quality of 4 algorithms (%)				Solution quality of 4 enhanced SOMs (%)			
			SA	Budinich	ESOM	eISOM	FLEXMAP	Budinich	ESOM	eISOM
HT30	30	415	**0**	1.51	**0**	**0**	2.17	**0**	**0**	**0**
EIL51	51	426	2.33	3.10	**2.10**	2.56	1.88	2.48	**0.93**	1.97
EIL101	101	629	5.74	5.24	**3.43**	3.59	**2.07**	4.31	2.72	2.92
KROA150	150	26524	4.31	4.36	2.04	**1.83**	2.90	2.23	1.69	**1.26**
KROA200	200	29368	5.61	6.13	2.91	**1.64**	3.22	2.67	1.96	**1.21**
LK318	318	42029	7.56	8.19	4.11	**2.05**	4.55	5.34	3.48	**1.93**
PCB442	442	50779	9.15	8.43	7.43	**6.11**	5.31	6.88	**5.11**	5.67
ATT532	532	87550	5.38	5.67	4.95	**3.35**	5.81	4.76	3.54	**2.39**
TK1002	1002	259045	7.32	8.75	6.37	**4.82**	6.99	7.44	5.07	**4.01**
TK2393	2392	378032	8.18	10.26	8.49	**6.44**	8.76	8.03	7.21	**5.83**
Average			5.56	6.16	4.18	**3.24**	4.37	4.41	3.17	**2.72**

TSP. Its learning rule has embodied three effective learning mechanisms of different SOM-like neural networks. It simultaneously takes account of the local optimality of the traditional SOM, the global optimality of the expanding SOM (ESOM), and the elastic force constraint in the elastic net. This learning rule enables the ISOM to generate near optimal solutions.

Since an efficient ISOM must have good coordination among the three learning mechanisms and use a suitable implementation of the expanding coefficient and parameter setting, it is very difficult to design a good ISOM manually. We have used a genetic algorithm (GA) to evolve an efficient ISOM automatically. The evolved ISOM (eISOM) has been examined on a wide spectrum of TSPs. Compared with the simulated annealing approach, it can generate tours about 3% shorter using less execution time. It has made at least 1% improvement over the SOM developed by Budinich, the ESOM, the convex elastic net, and the FLEXMAP algorithm. Though its solution accuracy is not yet comparable to some sophisticated heuristics, the eISOM is one of the most accurate SOMs for the TSP with the quadratic computation complexity. This point also indicates the GA has successfully found a good coordination of the three learning mechanisms of the ISOM.

This research not only supports that GAs can be used to solve complicated problems effectively, but also substantiates the idea of the ISOM to integrate strengths of different learning mechanisms. We expect to use this methodology to handle other problems such as cluster analysis. We also expect to enhance the performance of the SOM-like neural networks by embodying another global property in the learning rules.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their careful reading of this paper and for their valuable and constructive comments, which have helped to improve the quality of the paper.

REFERENCES

[1] S. Abe, "Convergence acceleration of the Hopfield neural network by optimization integration step sizes," *IEEE Trans. Syst., Man, Cybern. B*, vol. 26, pp. 194–201, 1996.
[2] H. Al-Mulhem and T. Al-Maghrabi, "Efficient convex-elastic net algorithm to solve the Euclidean traveling salesman problem," *IEEE Trans. Syst., Man, Cybern. B*, vol. 28, pp. 618–620, Aug. 1998.
[3] B. Angèniol, G. D. L. C. Vaubois, and J. Y. L. Texier, "Self-organizing feature maps and the travelling salesman problem," *Neural Netw.*, vol. 4, no. 1, pp. 289–293, 1988.
[4] M. Budinich, "A self-organizing neural network for the traveling salesman problem that is competitive with simulated annealing," *Neural Comput.*, vol. 8, pp. 416–424, 1996.

[5] L. I. Burke and P. Damany, "The guilty net for the traveling salesman problem," *Comput. Opt.*, vol. 19, pp. 255–265, 1992.
[6] M. Chang, H. Yu, and J. Heh, "Evolutionary self-organizing map," in *Proc. 1998 IEEE Int. Joint Conf. Neural Networks*, vol. 1, 1998, pp. 680–685.
[7] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 53–66, Apr. 1997.
[8] R. Durbin and D. Willshaw, "An analogue approach to the traveling salesman problem," *Nature (London)*, vol. 326, pp. 689–691, Apr. 1987.
[9] F. Favata and R. Walker, "A study of the applicationof Kohonen-type neural networks to the traveling salesman problem," *Biol. Cybern.*, vol. 64, pp. 463–468, 1991.
[10] B. Fritzke and P. Wilke, "FLEXMAP—A neural network with linear time and space complexity for the traveling salesman problem," in *Proc. IJCNN-90 Int. Joint Conf. Neural Networks*, 1991, pp. 929–934.
[11] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[12] B. Golden and W. Stewart *et al.*, "Empirical analysis of heuristics," in *The Travelling Salesman Problem*, E. Lawler *et al.*, Eds. New York: Wiley, 1985.
[13] F. Guerrero, S. Lozano, D. Canca, J. M. Garcia, and K. A. Smith, "A new self-organizing neural network for solving the travelling salesman problem," in *Eng. Syst. Design: Neural Networks, Fuzzy Logic, Evolutionary Programming, Data Mining, ComplexSystems*, C. Dagli *et al.*, Eds., 2001, vol. 11, pp. 865–870.
[14] J. J. Hopfield and D. W. Tank, "'Neural' computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.
[15] D. S. Hwang and M. S. Han, "Two phase SOFM," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 2, 1994, pp. 742–745.
[16] H. D. Jin, K. S. Leung, and M. L. Wong, "An integrated self-organizing map for the traveling salesman problem," in *Advances in Neural Networks and Applications*, N. Mastorakis, Ed, Singapore: World Scientific, Feb. 2001, pp. 235–240.
[17] D. S. Johnson and L. A. McGeoch, "The travelling salesman problem: A case study," in *Local Search in Combinatorial Optimization*, E. Aarts and J. Karel, Eds. New York: Wiley, 1997, pp. 215–310.
[18] K. Katayama and H. Narihisa *et al.*, "Iterated local search approach using genetic Transformation to the traveling salesman problem," in *Proc. Genetic Evolutionary Computation Conf.*, vol. 1, W. Banzhaf *et al.*, Eds., 1999, pp. 321–328.
[19] S. G. Kirkpatrick, Jr., C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.
[20] J. Knox, "Tabu search performance on the symmetric traveling salesman problem," *Comput. Oper. Res.*, vol. 21, pp. 867–876, 1994.
[21] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biol. Cybern.*, vol. 43, no. 2, pp. 59–69, 1982.
[22] T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, V. Paatero, and A. Saarela, "Organization of a massive document collection," *IEEE Trans. Neural Networks*, vol. 11, pp. 574–585, May 2000.
[23] N. Krasnogor and J. Smith *et al.*, "A memetic algorithm with self-adaptive local search: TSP as a case study," in *Proc. Genetic Evolutionary Computation Conf.*, D. Whitley *et al.*, Eds., 2000, pp. 987–994.
[24] S. Kwong, C. W. Chau, K. F. Man, and K. S. Tang, "Optimization of HMM topology and its model parameters by genetic algorithms," *Pattern Recognit.*, vol. 34, no. 2, pp. 509–522, Feb. 2001.
[25] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *Eur. J. Oper. Res.*, vol. 59, pp. 345–358, 1992.
[26] K. S. Leung, H. D. Jin, and Z. B. Xu, "An expanding self-organizing neural network for the traveling salesman problem," *Neural Netw.*, Aug. 2002, submitted for publication.
[27] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling salesman problem," *Oper. Res.*, vol. 21, pp. 498–516, 1973.
[28] A. S. Nissinen and H. Hyotyniemi, "Evolutionary training of behavior-based self-organizing map," in *Proc. 1998 IEEE Int. Joint Conf. Neural Networks*, vol. 1, 1998, pp. 660–665.
[29] G. Reinelt, "TSPLIB—A traveling salesman problem library," *ORSA J. Computing*, vol. 3, no. 4, pp. 376–384, 1991.
[30] K. A. Smith, "An argument for abandoning the travelling salesman problem as a neural network benchmark," *IEEE Trans. Neural Networks*, vol. 7, pp. 1542–1544, 1996.
[31] K. A. Smith, "Neural networks for combinatorial optimization: A review of more than a decade of research," *INFORMS J. Comput.*, vol. 11, no. 1, pp. 15–34, 1999.
[32] M. C. Su and H. T. Chang, "Genetic-algorithms-based approach to self-organizing feature map and its application in cluster analysis," in *Int. Joint Conf. Neural Networks Proc. IEEE World Congr. Computational Intelligence*, vol. 1, 1998, pp. 735–740.
[33] A. Y. C. Tang and K. S. Leung, "A modified edge recombination operator for the travelling salesman problem," *Lecture Notes Computer Science*, vol. 866, pp. 180–188, 1994.
[34] S. Viswanathan, I. Ersoy, F. Bunyak, and C. Dagli, "Evolving neural networks applied to predator-evader problem," in *Int. Joint Conf. Neural Networks, IJCNN'99*, vol. 4, 1999, pp. 2394–2397.
[35] C. Voudouris and E. Tsang, "Guided local search and its application to the traveling salesman problem," *Eur. J. Oper. Res.*, vol. 113, no. 2, pp. 469–499, 1999.
[36] G. Weiß, "Neural networks and evolutionary computation. Part I: Hybrid approaches in artificial intelligence," in *Proc. First IEEE Conf. Evolutionary Computation*, vol. 1, 1994, pp. 268–272.
[37] Z. B. Xu, H. D. Jin, K. S. Leung, L. Leung, and C. K. Wong, "An automata network for performing combinatorial optimization," *Neurocomputing*, vol. 47, pp. 59–83, Aug. 2002.
[38] X. Yao and Y. Liu, "Toward designing artificial neural networks by evolution," *Appl. Math. Comput.*, vol. 91, no. 1, pp. 83–90, Apr. 1998.

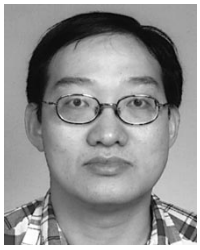
Hui-Dong Jin (S'02–M'03) received the B.Sc. degree in applied mathematics in 1995, and the M.Sc. degree in applied mathematics from the Institute of Information and System Sciences in 1998, both from Xi'an Jiaotong University, China. He is currently pursuing the Ph.D. degree in computer science and engineering at the Chinese University of Hong Kong, Hong Kong.

His research interests include data mining, evolutionary computation, and neural networks. He is a Student Member of ACM and ISGEC.

Kwong-Sak Leung (M'77–SM'89) received the B.Sc. degree and Ph.D. degree in engineering, 1977 and 1980, respectively, from the University of London, Queen Mary College.

He worked as a Senior Engineer on contract R&D at ERA Technology and later joined the Central Electricity Generating Board to work on nuclear power station simulators in England. He joined the Computer Science and Engineering Department at the Chinese University of Hong Kong in 1985, where he is currently Professor and Chairman of the Department. His research interests are in soft computing, including evolutionary computation, neural computation, probabilistic search, information fusion, and data mining, fuzzy data and knowledge engineering. He has published over 170 papers and two books in fuzzy logic and evolutionary computation.

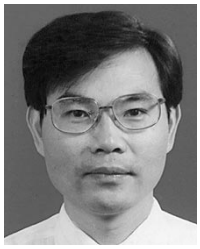
Dr. Leung has been chair and member of many program and organizing committees of international conferences. He is in the Editorial Board of Fuzzy Sets and Systems and an Associate Editor of International Journal of Intelligent Automation and Soft Computing. He is a chartered engineer, a member of lEE and ACM and a fellow of HKCS and HKIE.



Man-Leung Wong (M'96) received the B.Sc., M.Phil., and Ph.D. degrees in computer science from the Chinese University of Hong Kong in 1988, 1990, and 1995, respectively.

He is an Assistant Professor at the Department of Information Systems of Lingnan University, Tuen Mun, Hong Kong. Before joining the university, he worked as an Assistant Professor at the Department of Systems Engineering and Engineering Management, the Chinese University of Hong Kong, and the Department of Computing Science, Hong Kong Baptist University. He worked as a Research Engineer at the Hypercom Asia Ltd. in 1997. His research interests are evolutionary computation, data mining, machine learning, electronic commerce, knowledge acquisition, fuzzy logic, and approximate reasoning.

Dr. Wong is a member of the ACM.



Zong-Ben Xu received the M.S. degree in mathematics in 1981, and the Ph.D. degree in applied mathematics in 1987 from Xi'an Jiaotong University, China.

In 1988, he was a Postdoctoral Researcher in the Department of Mathematics, the University of Strathclyde, UK. He worked as a research fellow in the Information Engineering Department from February 1992 to March 1994, the Center for Environment Studies from April 1995 to August 1995, and the Mechanical Engineering and Automation Department from September 1996 to October 1996, at the Chinese University of Hong Kong. From January 1995 to April 1995, he was a research fellow in the Department of Computing at the Hong Kong Polytechnic University. He has been with the faculty of Science and Research Center for Applied Mathematics at Xi'an Jiaotong University since 1982, where he was promoted to Associate Professor in 1987 and Full Professor in 1991, and now serves as an authorized Ph.D. Supervisor in mathematics, Dean of the Faculty of Science, and Director of the Institute for Information and System Sciences. He has published two monographs and more than 80 academic papers on nonlinear functional analysis, numerical analysis, optimization techniques, neural networks, and genetic algorithms, most of which are in international journals. His current research interests include neural networks, evolutionary computation, and multiple objective decision making theory.

Dr. Xu holds the title "Owner of Chinese Ph.D. Degree Having Outstanding Achievements" awarded by the Chinese State Education Commission and the Academic Degree Commission of the Chinese Council in 1991. He is a member of the New York Academy of Sciences and International Mathematicians Union (IMU).