专 题 汇 编

ISCAS' 神经网络

1

ON THE QUADRATIC EXTENSION OF THE CANONICAL PIECEWISE-LINEAR NETWORK

JI-NAN LIN AND ROLF UNBEHAUEN

UNIVERSITÄT ERLANGEN-NÜRNBERG
Lehrstuhl für Allgemeine und Theoretische Elektrotechnik
Cauerstrasse 7, D-8520 Erlangen, Federal Republic of Germany

Abstract – The network based on the canonical piecewise-linear representation (CPWL) is a new type of neural networks. Compared to the conventional mapping networks, e.g., the back-propagation network and the radial basis network, the CPWL network has advantages in both learning and implementation aspects. In this paper, an extension of the CPWL network called the network of canonical piecewise-quadratic representation with linear partitions (CPWQ-L) is proposed. The CPWQ-L network shares the advantages of the CPWL network in explicitly and compactly realizing a spline approximation of nonlinear functions, while it extends the approximation capability of the latter at a low cost in computation and implementation. Examples of computer simulations are presented to demonstrate the superiority of the CPWQ-L network in some applications.

I. INTRODUCTION

Various mapping networks based on different ways of function representation or approximation are developed. The representative ones are the back-propagation network [1] and the radial basis network [2]. Recently, a new type of networks has arisen which learn to realize a nonlinear mapping in form of the canonical piecewise-linear representation (CPWL) of functions [3,4]. The basic representation of the CPWL [5] can be writen as

$$y = f(\boldsymbol{x}) = \boldsymbol{b}^T\boldsymbol{x} + \sum_{i=1}^{\sigma} c_i \mid \boldsymbol{\alpha}_i^T\boldsymbol{x} \mid , \tag{1}$$

where $\boldsymbol{x} := [x_0\ x_1\ x_2\ \cdots\ x_N]^T$ with $x_0 \equiv 1$ and N being the dimension of the domain. $\boldsymbol{b}$ and $\boldsymbol{\alpha}_i$ are parameter vectors, and c_i are constant parameters. Compared to other representation methods of piecewise-linear (PWL) functions, this representation has superior properties. It takes a unique form with the minimal number of free coefficients for describing a large class of continuous PWL functions which is easy to be dealt with in both theoretical study and applications. This representation has an explicit meaning for the vectors $\boldsymbol{\alpha}_i$ indicate directly the σ linear partition boundaries of the domain of a PWL function. From the viewpoint of neural networks, (1) implies a two-layer feedforward network with the nonlinearity of the hidden units being the absolution $|.|$. It is not difficult to derive a back-propagation-like learning algorithm for this network. Compared to the conventional mapping networks, the network based on (1) (which will be called the CPWL network) has advantages in both learning and implementation aspects.

In this paper we propose a quadratic extension of the CPWL and some simplifications. On this basis, mapping networks are developed with corresponding learning algorithms. The advantages of these networks are related with some computer simulation examples, in comparison with the CPWL network.

II. A QUADRATIC EXTENSION OF THE CPWL

We extend (1) in the form

$$y = f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{B}\boldsymbol{x} + \sum_{i=1}^{\sigma}(\boldsymbol{c}_i^T\boldsymbol{x}) \mid \boldsymbol{\alpha}_i^T\boldsymbol{x} \mid \tag{2}$$

with $\boldsymbol{B} := \{B_{pq}\}$ being a parameter matrix. (2) represents a class of continuous piecewise-quadratic (PWQ) functions with linear partitions of the domain. Since it stems from the CPWL, we will call it "the canonical piecewise-quadratic representation with linear partitions" (denoted as CPWQ-L).

By substituting the first term on the right-hand side (called the base function) with a bilinear or a linear form, we get simplifications of (2), i.e.

$$y = f(\boldsymbol{x}) = (\boldsymbol{b}_1^T\boldsymbol{x})(\boldsymbol{b}_2^T\boldsymbol{x}) + \sum_{i=1}^{\sigma}(\boldsymbol{c}_i^T\boldsymbol{x}) \mid \boldsymbol{\alpha}_i^T\boldsymbol{x} \mid \tag{3}$$

or

$$y = f(\boldsymbol{x}) = \boldsymbol{b}^T\boldsymbol{x} + \sum_{i=1}^{\sigma}(\boldsymbol{c}_i^T\boldsymbol{x}) \mid \boldsymbol{\alpha}_i^T\boldsymbol{x} \mid . \tag{4}$$

We denote these two simplifications (3) and (4) by CPWQ-L(I) and CPWQ-L(II), respectively. Clearly, the CPWQ-L(I) and the CPWQ-L(II) have successively smaller representation capabilities of PWQ functions than the CPWQ-L. However, these three are all extensions of the CPWL and, hence, are able to modify it.

This work is supported by the Deutsche Forschungsgemeinschaft, Fed. Rep. of Germany.

0-7803-0593-0/92 $3.00 1992 IEEE

The CPWQ-L's share the advantages of the CPWL over the conventional spline representations of functions, i.e., they take an explicit and compact form. The additional number of free coefficients of them over the CPWL can be regarded as a cost for the extension, which, as will be seen, is worthy in many cases.

III. THE CPWQ-L NETWORKS

According to the CPWQ-L representations (2), (3) and (4), we can build up feedforward networks as shown in Fig. 1, where the subnetwork N_B implements the base function. This network requires the same basic components as the CPWL network, i.e., linear correlators and absolutors. Additionally, it require multipliers. The number of multipliers required by the CPWQ-L network is $N(N+1)/2+\sigma$, while, by the CPWQ-L(I), $\sigma+1$ and by the CPWQ-L(II), σ.

Similar to the derivation of the learning algorithm for the CPWL network [3,4], we can derive the learning algorithms for the CPWQ-L networks. By using the LMS algorithm to minimize the cost function of the mean squared errors (MSE)

$$J := \frac{1}{T}\sum_{t=1}^{T} e_t^2 := \frac{1}{T}\sum_{t=1}^{T}[d_t - y_t]^2 := \frac{1}{T}\sum_{t=1}^{T}[d_t - f(\boldsymbol{x}_t)]^2 , \quad (5)$$

where $\{(\boldsymbol{x}_t, d_t) \mid t = 1,2,...,T\}$ are the training samples, we have the learning procedures as follows:

$$\boldsymbol{\alpha}_i(k+1) = \boldsymbol{\alpha}_i(k) + \mu e(k) u_i(k) \mathrm{sgn}(v_i(k)) \boldsymbol{x}(k), \quad (6)$$

$$\boldsymbol{c}_i(k+1) = \boldsymbol{c}_i(k) + \mu e(k) \mid v_i(k) \mid \boldsymbol{x}(k), \quad (7)$$

with

$$u_i(k) := \boldsymbol{c}_i^T(k)\boldsymbol{x}(k)$$

$$v_i(k) := \boldsymbol{\alpha}_i^T(k)\boldsymbol{x}(k), \quad i = 1,2,...,\sigma,$$

and for the CPWQ-L network

$$B_{pq}(k+1) = B_{pq}(k) + \mu e(k) x_p(k) x_q(k), \quad (8\text{-}1)$$

for $p = 0,1,...,N, q = 0,1,...,p$, for the CPWQ-L(I) network

$$\boldsymbol{b}_1(k+1) = \boldsymbol{b}_1(k) + \mu e(k) u_{02}(k) \boldsymbol{x}(k), \quad (8\text{-}2\text{-}1)$$

$$\boldsymbol{b}_2(k+1) = \boldsymbol{b}_2(k) + \mu e(k) u_{01}(k) \boldsymbol{x}(k), \quad (8\text{-}2\text{-}2)$$

with

$$u_{01}(k) := \boldsymbol{b}_1^T(k)\boldsymbol{x}(k),$$

$$u_{02}(k) := \boldsymbol{b}_2^T(k)\boldsymbol{x}(k),$$

and for the CPWQ-L(II) network

$$\boldsymbol{b}(k+1) = \boldsymbol{b}(k) + \mu e(k) \boldsymbol{x}(k), \quad (8\text{-}3)$$

with $(\boldsymbol{x}(k), d(k)) \in \{(\boldsymbol{x}_t, d_t) \mid t = 1,2,...,T\}$, for $k = 1,2,...$.

We have only considered the single output case, but it is easy to extend to the multi-output case. A direct way is to take a mapping of $\mathbb{R}^N \longrightarrow \mathbb{R}^M$ as M functions of $\mathbb{R}^N \longrightarrow \mathbb{R}$ and to obtain a network of multi-output through parallelling M subnetworks of single output [6]. A simplification is also possible. For example, if we assume that there is always an optimal partition of the input space for all the functions of the different outputs, the array of $\mid \boldsymbol{\alpha}_i^T \boldsymbol{x} \mid$, $i = 1,2,...,\sigma$ can be common for the subnetworks of different outputs.

IV. SIMULATIONS

This section concerns the significance of the quadratic extension of the CPWL, i.e., the CPWQ-L's. Examples of our computer simulation results are presented to demonstrate the advantages of the CPWQ-L networks over the CPWL network. For convenience of illustration, the examples presented here have all to do with the 2-D input space and a single output. But similarity with higher input and output dimensions is obvious.

Ex. 1: Surface fitting Learning of a mapping network from some given training samples can be considered as surface fitting, from the viewpoint of function approximation. We know that the CPWL is suitable for representing a large class of PWL functions. However, a PWL representation is not always effective in approximating arbitrary functions. To satisfy a degree of accuracy in some applicatons of the CPWL network, fine and close partition of the input space is often required, which causes not only a high cost in computation and implementation, but also difficulties in learning. In these cases, improvement can be achieved by using the CPWQ-L networks. Here we present one of the computer simulation results, where the CPWL network and the CPWQ-L networks were used to fit continuous surfaces in terms of given groups of training samples. Fig. 2(a) shows the original surface in this simulation. 1000 training samples were drawn from this surface randomly. The CPWL network learned from these training samples. Fig. 2(b) shows a typical result of the CPWL network with $\sigma = 10$. 80 epoches of learning was required to reduce the residual MSE to 0.051. Then, the CPWQ-L networks were used to learn from the same group of training samples to fit the surface. Fig. 2(c) shows the typical results of a CPWQ-L(I) network with $\sigma = 5$. The corresponding residual MSE is 0.014, which was obtained after 40 epoches of learning.

Our simulations demonstrate that the CPWQ-L networks behave, in general, superiorly to the CPWL network in surface fitting, i.e., they can achieve a more smooth fitting with a smaller residual MSE for either fewer partition boundaries or fewer free coefficients than that of the latter. Since, with the increase of the number of the boundaries, the convergence of the learning procedure of the CPWL network be-

comes difficult and slow, it means that, for a same requirement of accuracy, the CPWQ-L networks learn more easily and quickly than the CPWL network. The simplifications of the CPWQ-L network through the CPWQ-L(I) and the CPWQ-L(II) may bring about some degradation, but it is not much, compared with the improvement over the CPWL network.

Ex. 2: Classification Learning to classifying the input space in terms of a given group of training patterns is an important application of mapping networks. Due to some limitations of the CPWL, the CPWL network is often not effective to work as a classifier. Fig. 3(a) shows a desired classification. The input space is partitioned by three lines into six subregions of two classes. The CPWL network learned this classification in terms of 1000 random samples from Fig. 3(a). A threshold device was used at the output of the network, in order to get the binary output values. The learning strategy was similar to that of the ADALINE given in [7]. A typical result with $\sigma = 3$ is shown in Fig. 3(b), which was obtained after 50 epoches of learning. The classification error is 28.9%. Clearly, there is no continuous PWL representation of three boundaries which, after the analog-to-binary conversion, is corresponding to Fig. 3(a). To approximate the classification more accurately, the number of boundaries need to be increased. Fig. 3(c) shows a typical learning result obtained by the CPWL network with $\sigma = 6$, where, after 90 epoches of learning, the classification error was reduced to 15.3%. Thanks to the curved property of the CPWQ-L, the CPWQ-L networks can approximate the classification more effectively. With $\sigma = 3$ the CPWQ-L(II) achieved after 30 epoches of learning the result shown in Fig. 3(d) where the classification error is 5.3%.

Ex. 3: Global learning It is known that, when the partition of the input space is fixed, the learning of the CPWL network becomes global, i.e., it converges uniquely to a global solution. This property is attractive in cases where the boundaries are pre-provided. For example, a regular (e.g. grid) partition of the domain is familiar for spline-approximation of functions, which can be effective in some cases. It is also expected that the global learning property of the CPWL network can provide a way to get an appraisal of an unknown function without the trouble of local solutions. However, with a regular partition of the domain a continuous PWL representation degenerates simply to a sum of univariate subfunctions, which limits its capability in approximating an arbitrary function. This limitation, however, is overcome by the CPWQ-L's. With fixed boundaries the learning procedures of the CPWQ-L networks are also global. The example presented here is to illustrate the problem of the CPWL network with a pre-provided regular partition of the input space and the advantages of a CPWQ-L network in this cases. The original function is shown in Fig. 4(a). The CPWL network learned to estimate the form of this surface in terms of 1000 random samples. It had 10 boundaries which were fixed as a uniform grid in the input space. The resulting surface of the estimation is shown in Fig. 4(b), with the residual MSE 0.025. In this case, adding the boundaries could not improve the estimation. The CPWQ-L(II) network was also employed to estimate the surface in terms of the same group of training samples. With four boundaries fixed as a uniform grid in the input space, it achieved the result shown in Fig. 4(c), where the residual MSE is 0.0023.

V. CONCLUSION

In this paper, the CPWL representation is extended to a quadratic version denoted by CPWQ-L, accompanied by some simplifications. The CPWQ-L networks and the corresponding learning algorithms are developed. The CPWQ-L networks share the advantages of the CPWL network in explicitly and compactly realizing a spline approximation of nonlinear functions, while they extend the approximation capability of the latter at a low cost in computation and implementation. Examples of computer simulations of learning to approximate functions are presented, which demonstrate that the CPWQ-L networks can generally achieve a more satisfactory result with either fewer partition boundaries or fewer free coefficients than the CPWL network. For a given requirement of accuracy, the CPWQ-L networks learn more quickly than the latter. Meanwhile, they can overcome some drawbacks of the latter in applications, e.g., classification or learning to approximate with a preprovided partition.

Some theoretical aspects concerning the quadratic extension of the CPWL are under current study. For instance, as with the CPWL, we can derive the existence conditions for the representation of CPWQ-L.

REFERENCES

[1] D. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in D. Rumelhart and J. L. McClelland (Eds.), *Parallel Distributed Processing*, vol. 1, pp. 318-362. MIT Press, Cambridge, MA, 1986.

[2] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems* 2, pp. 321-355, 1988.

[3] J.-N. Lin and R. Unbehauen, "Adaptive nonlinear digital filter with canonical piecewise-linear structure," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 347-353, Mar. 1990.

[4] R. Batruni,"A multilayer neural network with piecewise-linear structure and back-propagation learning," *IEEE Trans. Neural Networks*, vol. 2, pp. 395-403, May 1991.

[5] L. O. Chua and A. C. Deng, "Canonical piecewise-linear representation," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 101-111, Mar. 1988.

[6] V. Y. Kreinovich, "Arbitrary nonlinearity is sufficient to represent all functions by neural networks: A theorem," *Neural Networks*, vol. 4, pp. 381-383, 1991.

[7] B. Widrow, R. G. Winter, and R. A. Baxter, "Layeres neural nets for pattern recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 1109-1118, July 1988.

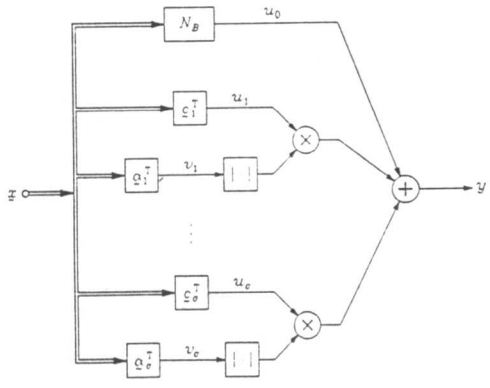

Fig. 1 Diagram of the CPWQ-L networks

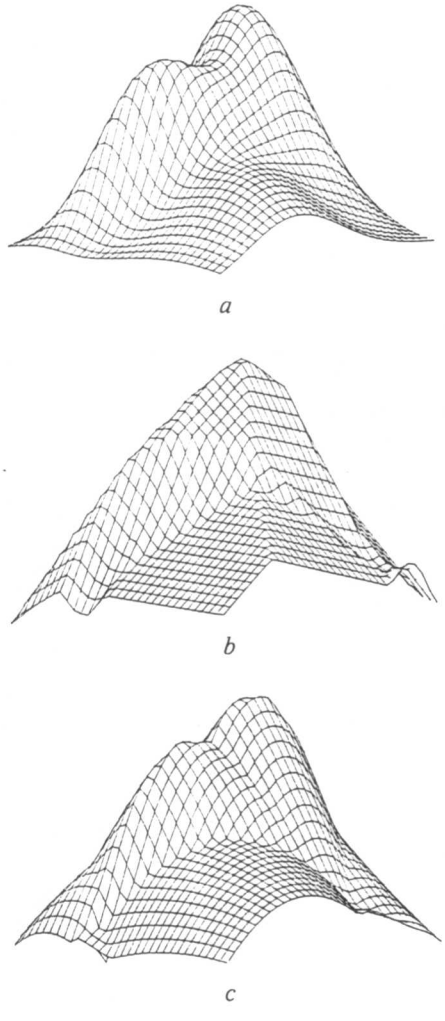

Fig. 2 (a) The original function for Ex. 1; (b) A learning result of the CPWL network with $\sigma = 10$ (epoches: 80, residual MSE: 0.051); (c) A learning result of the CPWQ-L(I) network with $\sigma = 5$ (epoches: 40, residual MSE: 0.014)

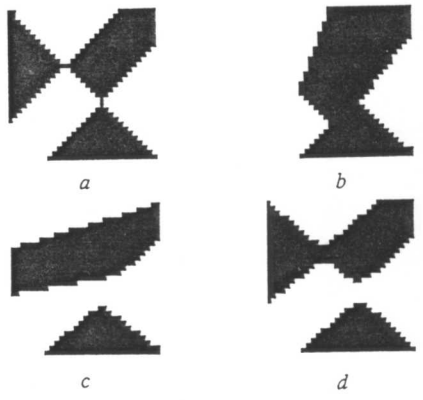

Fig. 3 (a) The original classification for Ex. 2; (b) A learning result of the CPWL network with $\sigma = 3$ (epoches: 50, classification error: 28.9%); (c) A learning result of the CPWL network with $\sigma = 6$ (epoches: 90, classification error: 15.3%); (d) A learning result of the CPWQ-L(II) network with $\sigma = 3$ (epoches: 30, classification error: 5.3%)

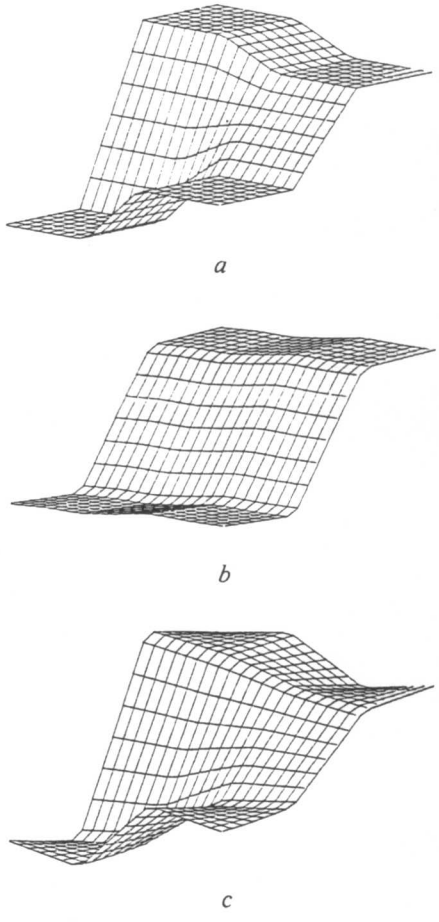

Fig. 4 (a) The original function for Ex. 3; (b) The learning result of the CPWL network (fixed partition of uniform grid) with $\sigma = 10$ (epoches: 50, residual MSE: 0.025); (c) The learning result of the CPWQ-L(II) network (fixed partition of uniform grid) with $\sigma = 4$ (epoches: 20, residual MSE: 0.0023)

DOUBLE SCROLL AND CELLULAR NEURAL NETWORKS

Fan Zou and Josef A. Nossek
Institute for Network Theory and Circuit Design
Technical University of Munich, FRG

Abstract – This paper reports a chaotic attractor with an autonomous three-cell Cellular Neural Network (CNN). It is shown that the attractor has a structure very similar to the *double scroll* attractor. Through some equivalent transformations this circuit in three major subspaces of its state space is shown to belong to *Chua's circuit family*, although originating from a completely different field.

1 Introduction

It has been long recognized that both real and mathematical models of neural systems can give rise to complex nonperiodic ("chaotic") dynamics. It is also recognized that brains are nonlinear networks composed of chaotic subsystems [4]. Therefore, it is important to investigate dynamical behaviors of minimalsized networks, which exhibit chaos. Unfortunately such chaotic attractors observed in continuous analog models of neural networks are rarely reported.

Recently, chaos has been observed in a two-cell nonreciprocal CNN with sinusoidal excitation [3]. For autonomous analog circuits three cells are needed to generate complex dynamic behavior. In this paper such a chaotic attractor is shown with a three-cell CNN. This attractor has a surprising similarity to the *double scroll* attractor [2]. In fact, with some approximations and equivalent transformations this three-cell CNN can be interpreted as a circuit, which in its major operating subspaces belongs to Chua's circuit family.

First the system equations are given and some related aspects are discussed in Section 2 briefly. Some simulated results from the proposed system are also given in this section. In Section 3 relations of this circuit to Chua's circuit family will be discussed.

2 System Description

Consider the CNN of Fig. 1(a), where the function $f(\cdot)$ is graphically shown by Fig.1(b). The dynamics of the system can be described by the set of ordinary differential equations:

$$\begin{aligned} \dot{x}_1 + x_1 &= p_1 f(x_1) - s f(x_2) - s f(x_3) \\ \dot{x}_2 + x_2 &= -s f(x_1) + p_2 f(x_2) - r f(x_3) \\ \dot{x}_3 + x_3 &= -s f(x_1) + r f(x_2) - p_3 f(x_3) \end{aligned} \qquad (1)$$

with the output function

$$f(x_i(t)) = \frac{1}{2}(|x_i(t)+1| - |x_i(t)-1|) \qquad (i = 1,2,3) \quad (2)$$

where $p_1 > 1$, $p_2 > 1$, $p_3 \geq 1$, $r > 0$, $s > 0$.

One may notice that a normalized description is used for convenience, and the inputs v_{uij}'s and bias currents I_{ij}'s are all omitted.

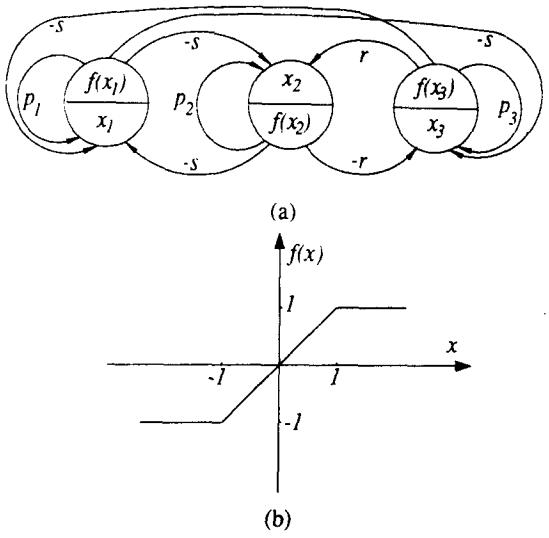

Figure 1. (a) The autonomous three-cell CNN ; (b) The output function

Fig. 2 shows the chaotic attractor observed by solving (1) with the following parameter set

$$p_1 = 1.25, \quad p_2 = 1.1, \quad p_3 = 1, \quad s = 3.2, \quad r = 4.4 \qquad (3)$$

and initial condition

$$\boldsymbol{x}(0) = [\ 0.1\ \ 0.1\ \ 0.1\ ]^T\ .$$

It is obvious that Fig. 2 shows an amazing similarity to the double scroll attractor. System (1), (2) along with the parameter set (3) has only three equilibria, which are all unstable. Therefore, the unstable solution of the system is not surprising. It will be shown that the eigenspaces of the equilibria have the same structure as the double scroll, so that the strong similarity is not a accidental phenomenon.

0-7803-0593-0/92 $3.00 1992 IEEE

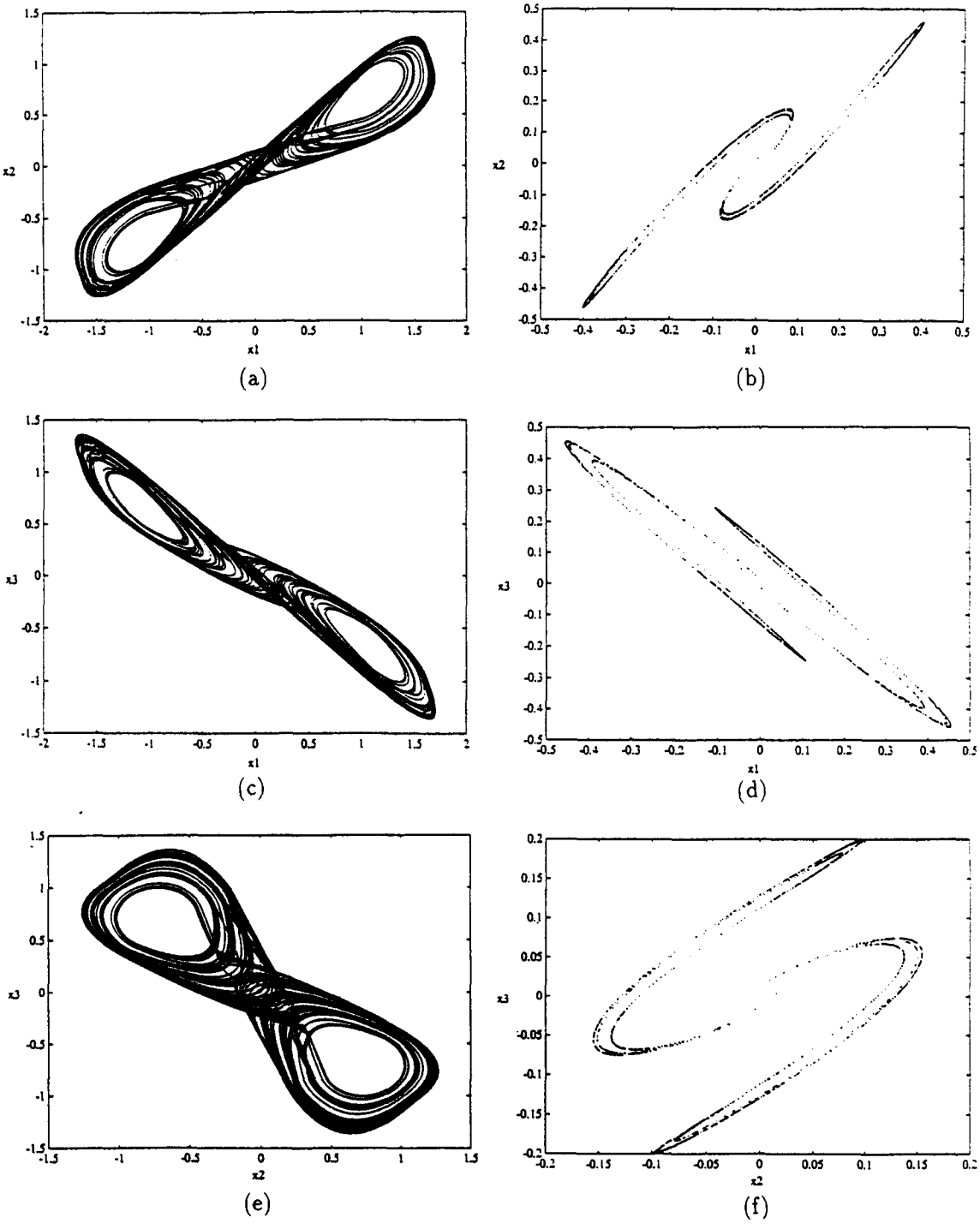

Fig. 2 The chaotic attractor. (a) Projection onto the (x_1, x_2)-plane. (b) Cross section of the (x_1, x_2)-plane with $x_3 = 0$. (c) Projection onto the (x_1, x_3)-plane. (d) Cross section of the (x_1, x_3)-plane with $x_2 = 0$. (e) Projection onto the (x_2, x_3)-plane. (f) Cross section of the (x_2, x_3)-plane with $x_1 = 0$.

Let us define the following three subsets of $\mathbb{R}^3$:

$$\begin{aligned} D_{+1} &= \{(x_1, x_2, x_3) \mid x_1 \geq 1,\ |x_2| \cdot |x_3| < 1\} \\ D_0 &= \{(x_1, x_2, x_3) \mid\ |x_1| \cdot |x_2| \cdot |x_3| < 1\} \\ D_{-1} &= \{(x_1, x_2, x_3) \mid x_1 \leq 1,\ |x_2| \cdot |x_3| < 1\}\ . \end{aligned} \tag{4}$$

With the parameters in (3), the equilibria are given by:

$$\begin{aligned} \boldsymbol{P}_{+1} &= (1.1971, 0.7273, -0.7107) \in D_{+1} \\ \boldsymbol{P}_0 &= (0, 0, 0) \in D_0 \\ \boldsymbol{P}_{-1} &= (-1.1971, -0.7273, 0.7107) \in D_{-1} \end{aligned} \tag{5}$$

The eigenvalues of Jacobi-matrix at these equilibria, and hence in the whole subspaces respectively, are calculated as:

$$\begin{aligned} \gamma_p &= -1.0:\ \sigma_p \pm j\omega_p = 0.05 \pm j4.3997 \text{ in } D_{+1} \text{ and } D_{-1} \\ \gamma_0 &= 1.935\ \ \sigma_0 \pm j\omega_0 = -0.7925 \pm j1.1593 \text{ in } D_0 \end{aligned}$$

Let $E^s(\boldsymbol{P}_\pm)$ be the eigenspace corresponding to the real eigenvalue γ_p at $\boldsymbol{P}_\pm$ and let $E^u(\boldsymbol{P}_\pm)$ be the eigenspace corresponding to the complex eigenvalues $\sigma_p \pm j\omega_p$ at $\boldsymbol{P}_\pm$. Following the same way, we define $E^u(\boldsymbol{P}_0)$ and $E^s(\boldsymbol{P}_0)$ corresponding to γ_0 and $\sigma_0 \pm j\omega_0$, respectively. Then,

$$\begin{aligned} \dim E^s(\boldsymbol{P}_\pm) &= \dim E^u(\boldsymbol{P}_0) = 1 \\ \dim E^u(\boldsymbol{P}_\pm) &= \dim E^s(\boldsymbol{P}_0) = 2 \end{aligned}$$

Now we see that the eigenspaces of the equilibria have almost identical structures with the double scroll, this gives some reasons, why both attractors share nearly the same appearance and form.

3 Relations to Chua's Circuit

Let us imagine that the output variables of cell 2 and 3 are not saturated, i.e. $f(x_2) = x_2$ and $f(x_3) = x_3$, (notice that this is the case in the subspaces D_{+1}, D_0, and D_{-1}). Then, because of the opposite-sign weights $+r$ and $-r$, which can be interpreted as a ideal gyrator, these two cells construct a oscillatory circuit. This transformation is shown in Fig. 3.

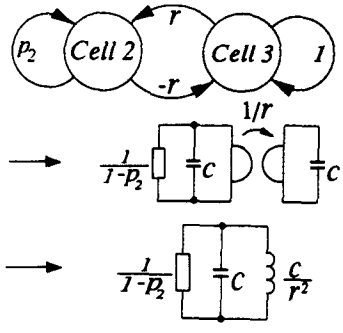

Figure 3. Circuit equivalent transformations of cell 2 and cell 3

On the other hand, cell 1 can be approximately substituted by a circuit with a linear capacitor and a nonlinear resistor (Fig. 4(a)). The nonlinear function of the resistance is given in Fig. 4(b). Now it is very obvious that the whole circuit shown in Fig. 5 really belongs to Chua's circuit family.

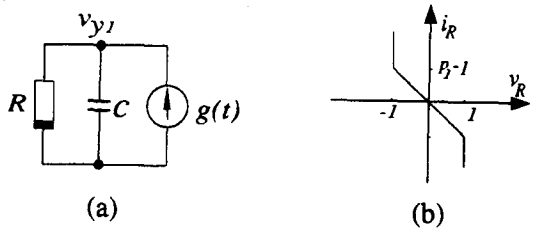

Figure 4. Approximation of the cell 1

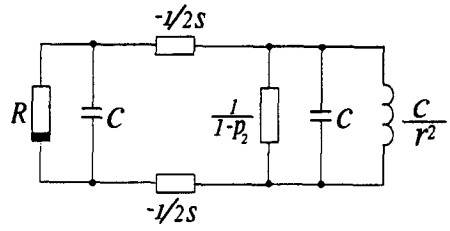

Figure 5. Approximation of the whole circuit

As one can see from Fig. 2, the output variables of cell 2 and 3 do not exceed the linear region ($|x_i| < 1$) for most operating time, so the approximation above gives a reasonable circuit-theoretic interpretation of the chaotic behavior. But it would be misleading to identify this attractor with the double scroll, because x_2 and x_3 do reach the saturation region ($|x_i| \geq 1$) for a not negligible time. It seems, that it is this saturation function that makes the chaotic behavior possible here. If $p_3 = 1$, $s = 3.2$ and $r = 4.4$ are fixed, the following relationship can be obtained: for a large p_2 (and consequently large p_1) one has a bad approximation and vice versa. An example to show this is given in Fig. 6.

4 Conclusion

An autonomous chaotic attractor with a three-cell CNN has been found, which has very similar structures with the double scroll.

References

[1] L. O. Chua and L. Yang, "Cellular Neural Networks: Theory", IEEE Trans. CAS, vol. 35, No.10, pp1257-1272, Oct.1988.

[2] T. Matsumoto, L. O. Chua and M. Komuro, "The Double Scroll", IEEE Trans. CAS, vol. 32, No.8, pp798-818, Aug. 1985

[3] F. Zou and J. A. Nossek, "A Chaotic Attractor with Cellular Neural Networks", IEEE Trans. on CAS, vol. 38, July, 1991

[4] K. Aihara,"Neural Networks and Chaos", Proceeding of IEEE ISCAS, 1991, pp 1367-1368

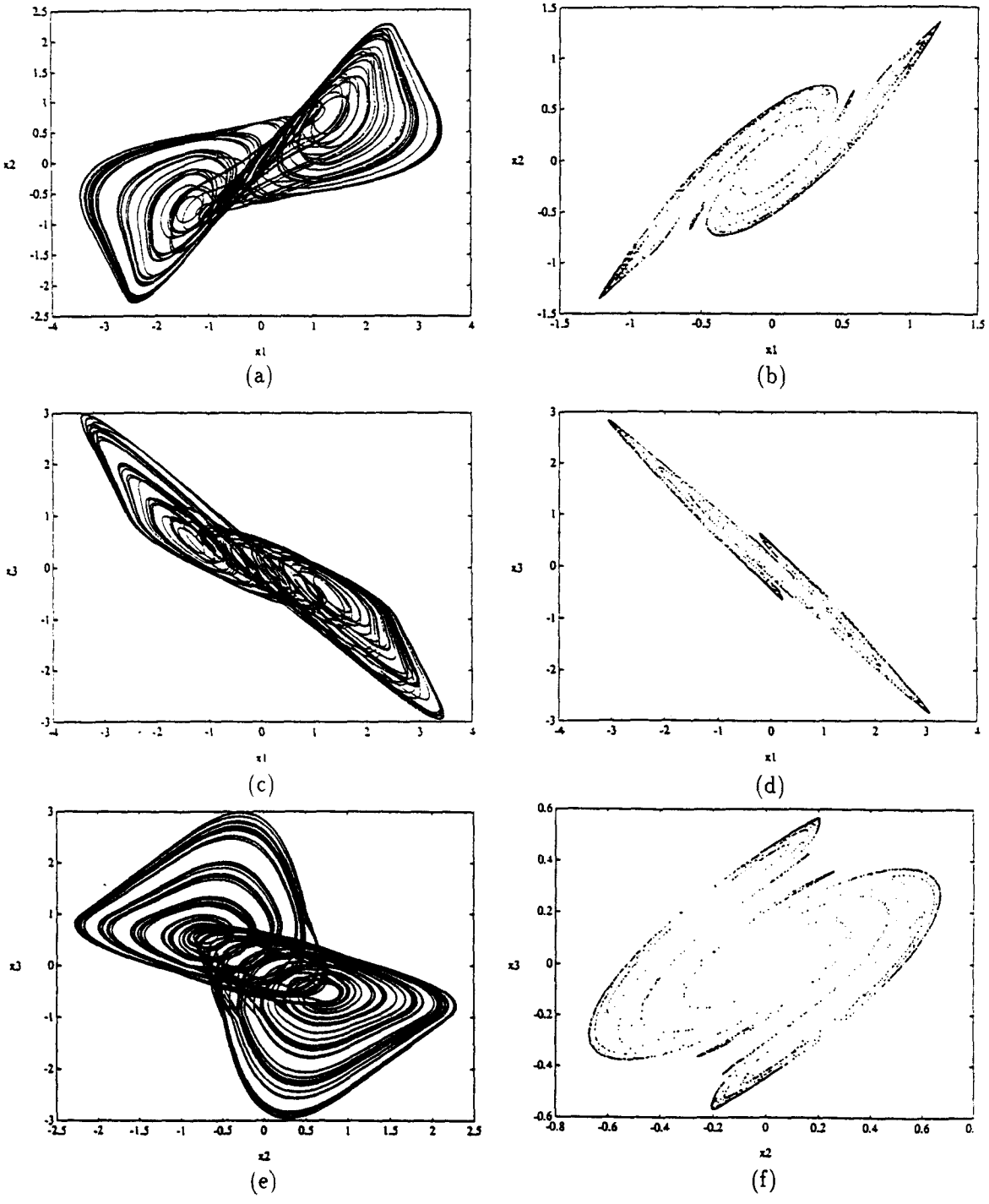

Fig. 6 The chaotic attractor with the parameters $p_1 = 1.99$, $p_2 = 2$, $p_3 = 1$, $s = 3.2$, $r = 4.4$; and initial states $\boldsymbol{x}(0) = [\ 0.1\ 0.1\ 0.1\ ]^T$. (a) Projection onto the (x_1, x_2)-plane. (b) Cross section of the (x_1, x_2)-plane with $x_3 = 0$. (c) Projection onto the (x_1, x_3)-plane. (d) Cross section of the (x_1, x_3)-plane with $x_2 = 0$. (e) Projection onto the (x_2, x_3)-plane. (f) Cross section of the (x_2, x_3)-plane with $x_1 = 0$.

PARTITIONING ON BOLTZMANN MACHINES*

A. Koenig N.Wehn † M. Glesner

Institute for Microelectronic Systems
Darmstadt University of Technology
D-6100 Darmstadt, Karlstrasse 15
Germany

Abstract

Recently neural networks have been claimed to be well suited for solving a class of optimization problems such as the travelling salesman problem. The Boltzmann Machine is a representative of a special class of neural networks, viz. probabilistic neural networks. The Boltzmann Machine's algorithm is very similar to the simulated annealing algorithm, with the additional advantage of inherent massive parallelism. This feature motivated the investigation and application of Boltzmann Machines for a plethora of optimization problems. In this paper we present a study to determine if the Boltzmann Machine is applicable to the partitioning problem which is one of the key problems in VLSI-synthesis. Various models will be derived. Sequential and parallel implementations of these models will be presented and their results will be compared to existing heuristics.

1 Introduction

In the last years the field of neural networks experienced a tremendous lot of activities, due to the great technological advance in VLSI-design. These advances offer the possibility of the implementation of large neural networks by dedicated VLSI-hardware. Neural networks are attractive for combinatorial optimization problems, because optimization problems can conveniently be mapped onto those networks. The main property of neural nets, beside adaptivity, is their inherent parallelism, thus offering computational power far beyond conventional approaches. In the past the Hopfield model was emphasized by different researchers [2], [3], [4] and only little attention was payed to the Boltzmann Machine. In our opinion the Boltzmann Machine seems to be well suited for optimization since it is very similiar to simulated annealing which has been proven to be a very efficient optimization strategy in many related CAD-VLSI fields. However the disadvantage of simulated annealing is its computational complexity. In contrast parallel Boltzmann machines provide equal performance and additionally offer, due to their inherent parallelism, the possibility of speed up of orders of magnitude by means of dedicated hardware implementations. In this paper we investigate the true potential of Boltzmann machines for the graph partitioning problem which is one of the key problems in VLSI synthesis [6], [7].

*Supported by EEC (ESPRIT project 3281 "ASCIS")
†N. Wehn is also with Siemens AG Corporate R & D

2 The Boltzmann Machine

The Boltzmann Machine, introduced by Hinton & Sejnowski [5], is a neural network with a stochastic state transition mechanism. As in Hopfield's model, the units have binary-valued states, i.e. they are either "on" or "off", and the connections are bidirectional. The strength of a Boltzmann Machine's connection, i.e. the edge weight, can be considered as the desirability that the units incident with this connection are both "on". The units in a Boltzmann Machine try to reach a maximal consensus about their individual states, subject to the desirabilities expressed by their connection strenghts. To accomplish this aim units adjust their states by means of a probabilistic state transition mechanism, which is governed by the simulated annealing algorithm [1]. The Boltzmann Machine can be employed in the domains of classification, learning and combinatorial optimization. In this paper the Boltzmann Machine (BM) is applied for combinatorial optimization only. Basic terms and definitions can be found in [1], that has been the foundation of our work.

3 Bipartitioning

In this approach only bipartitioning is considered, i.e the nodes V of an instance graph are divided in two sets V_1, V_2 so that the sum of the edge weights incident with nodes of those two sets is optimal. Introducing, without loss of generality, some restrictions on the node and edge weigths, this can be formulated as :

Definition 1 : Let $G(V, E)$ be a graph with n vertices and let $w : V \rightarrow [1]$ and $c : E \rightarrow [0, 1]$ be the node and edge weighting function, respectively. Let a cost function $f(V_1, V_2)$ be defined as

$$f(V_1, V_2) = \frac{1}{2} \sum_{i=1}^{2} \sum_{e \in E_{ext,i}} c(e)$$

$$E_{ext,i} = \{e \in E | e \cap V_i \neq \emptyset, e \backslash V_i \neq \emptyset\} \quad (1)$$

Then a *Bipartition* of the set V in the sets V_1 and V_2 subject to an additional balance constraint is defined as

$$V_1 \cup V_2 = V; V_1 \cap V_2 = \emptyset \quad (2)$$

$$| |V_1| - |V_2| | \leq \alpha \cdot |V| \;\; ; \;\; \frac{1}{2} \leq \alpha \leq 1 \quad (3)$$

$$f(V_1, V_2) = \frac{1}{2} \sum_{i=1}^{2} \sum_{e \in E_{ext,i}} c(e) \;\; optimum \quad (4)$$

0-7803-0593-0/92 $3.00 1992 IEEE

Minimizing the sum of equ.4 is denoted as MIN CUT, maximizing as MAX CUT. Equ. 3 expresses the balance constraint imposed during optimization to exclude, e.g. the *trivial solution* $V_1 = \emptyset$ and $V_2 = V$, or $V_1 = V$ and $V_2 = \emptyset$. All other solutions not satisfying the balance constraint are also rejected. The cost function for the general CUT problem can be reformulated, if $w_{ij} = c(e_{ij})$ and x_i represents the 0-1 variable associated with v_i and $x_i = 0 \rightarrow v_i \in V_1$; $x_i = 1 \rightarrow v_i \in V_2$:

$$f(X) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij}((1 - x_i)x_j + x_i(1 - x_j)) \qquad (5)$$

4 Development of Boltzmann Machine Models

This section is dedicated to the development of several distinct BM implementations (denoted BM models) for the MIN CUT problem. There are two alternative approaches. In [1] a BM model has been tailored for the MAX CUT problem, on which the transformed problem could be mapped. Alternatively, a BM model could be specifically tailored for the MIN CUT problem.

4.1 Implementation by graph transformation

As mentioned above, the MAX CUT problem has been already successfully implemented on a BM. Let C_b=$\{\{u_i, u_i\} | i \in V\}$ and $C_w = \{\{u_i, u_j\} | \{i, j\} \in E\}$. Then the cost function (COF) of the BM is :

$$C(k) = \sum_{\{u_i,u_i\} \epsilon C_b} b_i k^2(u_i) + \sum_{\{u_i,u_j\} \epsilon C_w} v_{ij} k(u_i) k(u_j) \qquad (6)$$

Choosing

$$b_i = \sum_{j=1}^{n} w_{ij} \qquad v_{ij} = -2w_{ij} \qquad (7)$$

This results in

$$C(k) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij}(x_i + x_j) \sum_{i=1}^{n} \sum_{j=i+1}^{n} -2w_{ij} \qquad (8)$$

which is identical to the COF of equ.5 and so the consensus function (CF) is **feasible** and **order-preserving** [1].
This result can be used to implement the MIN CUT problem on a BM by means of a little artifice.
The idea is to find an algorithm that transformes the graph so that the MIN CUT solution of the original graph is identical to the MAX CUT solution of the transformed graph. This algorithm is defined as follows :

1. Find the greatest edge weight $w^* = \max_{i=1}^{n} w_i$
2. Increment this value, i.e. $w^* = w^* + 1$
3. Complete the graph with zero weighted edges, omitting loops, i.e.
   $E^* = E \bigcup \{\{u_i, u_j\} | (\{u_i, u_j\} \cap E) \neq \emptyset \; \forall \; i = 1..n, j = i + 1..n\}$
4. Transform edge weights :
   for i:= 1 to $|E^*|$ do $w_i := w^* - w_i$
5. Apply BM MAX CUT solution to the transformed graph. It has to be mentioned, that unfortunately this solution does not comply with the balance constraint. Only the trivial solution is excluded, because the trivial solution is no local maximum.

So the balance constraint of equ.3 is reduced to

$$| |V_1| - |V_2| | \leq |V| - 1 \qquad (9)$$

4.2 Direct implementation of the Min Cut

Development of a Two-Layer Model The direct implementation is hampered by an obstacle origining from the opposed directions of "growth" of the CF and the COF during optimization. The CF of the BM increases, whereas the COF should decrease. This problem can be mended if, instead of minimizing the Cut sum, the sum of edge weights outside of the Cut is maximized. So a new COF $f(X)^*$ is introduced

$$f(X)^* = \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij}(x_i x_j + (1 - x_i)(1 - x_j)) \qquad (10)$$

Furthermore the stringent balance constraint of equ.3 can be softened to a balance dependend penalty term in the COF so that a modified COF $f(X)^{**}$ results :

$$f(X)^{**} = f(X)^* + P \qquad (11)$$

The penalty term is required to have a parabolic characteristic, centered at full balance. Based on this modified COF a two layer BM model (2LM) has been devised (Fig. 1).
Each layer mirrors the problem graph structure and the activity of a unit corresponds with its membership in the set left or right of the Cut. Since a node cannot be member of

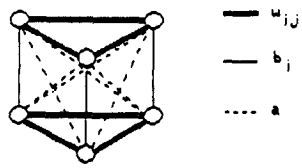

Figure 1: Two-Layer Model

both sets at once an additional fine term is introduced.
The COF has to be rewritten for the 2LM.

$$f(X)^{**}_{2LM} = \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_{i1} x_{j1} + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_{i2} x_{j2}$$

$$+ \sum_{i=1}^{n} \sum_{j=1}^{n} a x_{i1} x_{j2} \qquad (12)$$

The CF for the 2LM is identical to the COF plus the additional fine term.
Theorem 1 : Let $C_{2LM}(k)$ be the CF of a BM implementing a 2LM

$$C_{2LM}(k) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij}(x_{i1}x_{j1} + x_{i2}x_{j2})$$

$$+ \sum_{i=1}^{n} \sum_{j=1}^{n} a x_{i1}x_{j2} + \sum_{i=1}^{n} b_i x_{i1}x_{i2} \qquad (13)$$

and let the weights b_i and a be determined by the following equations

$$b_i < -(w_{ij}^* + (n-1)a) \qquad (14)$$

w_{ij}^* denotes the edge weight sum of all edges incident with node i, i.e. $w_{ij}^* = \{\sum_{j=1}^{n} w_{ij} | e_{ij} \cap V_i \neq \emptyset\}$,

$$a > \frac{4}{n^2} \sum w_{ij} \qquad (15)$$

then the CF of the 2LM is **feasible** and **order-preserving**. The proof of theorem can be found in [9]. In this proof it is shown, that any configuration k, with no fine term active, and with n "on" units is a local maximum of the CF and corresponds to a feasible solution of the COF and further, that the penalty term is maximal at full balance. Additionally, the motivation for the choice of the parameters b and a is presented.

Development of a One-Layer Model It is also possible to derive a one-layer-BM-model (1LM) that implements the balance constraint.

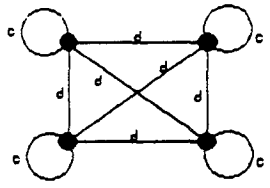

Figure 2: One-Layer Model

Theorem 2 : Let $C(k)$ be the CF for a BM just implementing the balance or penalty term

$$C(k) = \sum_{i=1}^{n} c \cdot x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} d \cdot x_i x_j \qquad (16)$$

Then, if c and d are chosen

$$d < 0 \;\; ; \; c = -\frac{d(n-1)}{2} \qquad (17)$$

the CF will be maximal for perfect balance. The proof of theorem can also be found in [9]. Now the straight forward approach would be to superimpose the COF of equ.10 and this balance term. But unfortunately edge weights of the COF and the balance term have opposed signs, which results in the unwanted elimination of edges. The MAX CUT approach of equ.8 does not have this drawback, so that a balanced MAX CUT can be implemented on the 1LM. In conjunction with the graph transformation algorithm a balanced MIN CUT implementation on the 1LM is possible. Additionally

$$d < -\frac{8}{n^2} \sum w_{ij} \qquad (18)$$

has to hold. This constraint is imposed to ensure that the worst balanced solution has a greater consensus than the best unbalanced solution. The parameter d determines the slope of the penalty term, and thus the emphasis on the two dual optimization aims of *cut sum minimization* and *balance*, respectively. The 2LM model has a complexity of $2n$ nodes and $1.5n^2 + 0.5n$ edges, the 1LM has $n$nodes and $0.5(n^2 + n)$ edges.

5 Simulation and Evaluation of BM Models

As problem instances three different graphs were randomly generated. Node number and connectivity is mirrored in the graph's name, edge number is 613, 2475 and 5588 for N50D50, N100D50 and N150D50, respectively. A connectivity of 50% was chosen, because for sparse graphs a heuristic procedure would perform better and for dense graphs, close to full connectivity and edge weights equal to one, partitioning becomes trivial. The graphs were partitioned with both BM models and, as a reference, with the Fiduccia-Mattheyses heuristic (FMH) [8].

The time steps displayed in all tables are relative, i.e. the number of trials of the BM during optimization has been counted. The best results obtained by simulations of the two models for the N50D50, the N100D50 and the N150D50 graph are displayed in Table 1.

Obviously, the 1LM outperforms the 2LM in both cases.

Mod.	COF	σ	Time	σ	Cut	σ	Best
1LM	2462.2	1	29750	1123	257.3	1	254
2LM	924.8	8	43050	756	298.4	12	271
FMH					257.0		257
1LM	12079.7	6	65750	3027	1087.8	6	1077
2LM	3959.0	14	85600	916	1231.8	16	1201
FMH					1119.0		1119
1LM	22169.0	10	104250	5335	2490.2	10	2476
FMH					2496.0		2496

Table 1: Results of N50D50 graph

So for the N150D50 graph the 2LM is no longer taken in consideration. The 1LM provides in any case results equal to or even superior than the FMH. It can be stated as the conclusion of this part, that a sequential BM implementing the MIN CUT problem by means of a 1LM can compete easily with known heuristics. In the following sections the topic of parallel BM will be addressed.

6 Parallel Boltzmann Machines

The advantage of neural networks is their inherent parallelism. Up to now only sequential BM have been considered. Now the possibilities of parallelisation of the BM will be examined. Theoretical background and proofs of convergence will be beyond of the scope of this paper but can be found in [1]. Focus of this work is on synchronously parallel BMs with unlimited parallelism. To efficiently implement parallelism it is necessary to localize all calculations to reduce global communication and thereby the complexity of a design. The following calculations have been localized :

- **Consensus difference $\Delta C_u(k)$, Cooling schedule, Markov chain length and Stop criterion**[1]
- **Acceptance propability** $A_k(u,c)$ The **AP** can be locally calculated based on the $\Delta C_u(k)$ of a unit u, but is then of course strongly influenced by the miscalculations occuring during the evaluation of $\Delta C_u(k)$.
- **Generation propability G(u) and the related selection mechanism** The **GP** of a BM applying unlimited parallelism can be chosen in the interval

$$\frac{1}{|U|} \leq G(u) \leq 1 \tag{19}$$

A global selection mechanism consists of just one random source, that selects one or a fixed amount of units. If the selection mechanism is changed and the global random source is replaced by $|V|$ independent random sources, one for each unit u, then the selection can take place without any global communication. This localisation of the selection mechanism introduces an interesting aspect. A global selection mechanism generates a state transition for an exactly fixed number of units, e.g. 60% for G(u) = 0.6 , whereas the local mechanism obeys the following rule :

$$P(X \leq m) = \sum_{k=0}^{m} P(X = k)$$
$$= \sum_{k=0}^{m} \binom{|U|}{k} G(u)^k (1 - G(u))^{m-k} \tag{20}$$

If $|U| = 50$, m = 30 and $G(u) = \frac{m}{|U|} = 0.6$ is chosen this will provide $P(X = 30) = 0.11456$ and $P(X < 30) = 0.43904$, $P(X > 30) = 0.4464$.

7 Simulations with Parallel BMs

A BM with the features described in the preceding section was implemented and simulations were carried out. The parameters K, L and α where chosen $K = 10, L = \frac{1}{4} \cdot |V|, \alpha = 0.95$. Fig. 3 illustrates results obtained for the N50D50 graph, using the 1LM. Evidently no convergence has been achieved. This result contradicts those reported in [1]. To verify the optimizers performance it has been tested with the MAX CUT problem, that has already been solved in [1]. The results of the optimizer for the MAX CUT problem and the graph from [1] were as expected and predicted. This proves the validity of our implementation of a BM applying unlimited parallelism.

8 Conclusion and Future Aspects

In this paper we have presented different models for the solution of the partitioning problem on a BM. Sequential and parallel BM optimizers were implemented and compared to existing heuristics. As it can be seen from the results the sequential implementation can compete with classical heuristics, but in this case there is no advantage in time complexity and behavior compared to simulated annealing. On the other hand the parallel implementation, which represents the advantage of the BM, is not applicable in this form for optimization problems. Convergence cannot be guaranteed and will not be achieved for dense graphs close to full connectivity, as is the case for the 1LM. Thus the BM implementing the described dynamics is not directly applicable for general problems. This contradicts statements found in the literature. Current research investigates the application of dynamics providing the desired convergence properties.

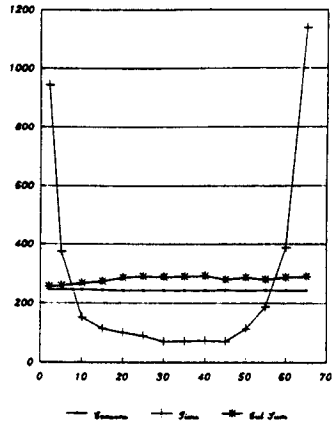

Figure 3: Parallel BM (1LM) and N50D50 graph

References

[1] E. Aarts, J. Korst, *Simulated Annealing and Boltzmann Machines*, John Wiley & Sons, Wiley-Interscience Series in Discrete Mathematics and Optimization, 1988

[2] M. L. Yu, *A Study of the Applicability of Hopfield Decision Neural nets to VLSI CAD*, 26. DAC, 1989, pp. 412-417

[3] R. Libeskind-Hadas, C. L. Liu, *Solutions of the Module Orientation and Rotation Problems by Neural Computation Networks*, 26. DAC, 1989, pp. 400-405

[4] J. S. Yieh, P. Mazunder, *A Neural Network for Circuit Partitioning*, 26. DAC, 1989, pp. 406-411

[5] G. E. Hinton, T. J. Sejnowski, D. H. Ackley, "Boltzmann Machines : Constraint Satisfaction Machines that Learn", *Tech. Rep. CMU-84-119* , Carnegie-Mellon-University, 1984

[6] W. E. Donath, *Logic Partitioning*, The Benjamin Cummings Publishing Company 1988, Physical Design Automation of VLSI Systems, pp. 65-80

[7] T. Lengauer, *The Combinatorial Complexity of Layout Problems*, The Benjamin/Cummings Publishing Company 1988, Physical Design Automation of VLSI Systems, pp. 461 - 488

[8] Fiduccia, Mattheyses, "A Linear time Heuristic for Improving Network Partitions", Proc. 19th Design Automation Conference, pp.175-181, ACM/IEEE 1982

[9] A. Koenig, "Ein Ansatz zur Loesung des Partitionierungsproblem auf der Basis der Boltzmann Maschine", *Tech. Report DAT DA KOEN 84*, Technische Hochschule Darmstadt, FG Mikroelektronische Systeme, 1990

An Improved Algorithm for Kohonen's Self-organizing Feature Maps

Chau-Yun Hsu and Hwai-En Wu

Department of Electrical Engineering Tatung Institute of Technology
Taipei Taiwan 10451 R.O.C.

Abstract--About artificial neural network, the well-known architecture and algorithm of self-organizing maps has the important property of topology-preserving mapping of various features of input signals and their abstractions with noise involving, However, in the formation (learning) of the mapping, the topological orders preserved in the map are not all "correct", e.g. for some topological orders there is no such feature relations in the input signals; also some "correct" topological orders corresponding to actual feature relations of the input signals are "lost" in the map. So it is more proper to call it a "piecewise-correct" map. This is due to two dominant factors: the initial weight problem and the input sequence order problem. Here we propose a modified algorithm which induces (1)insertion and deletion of cells (2)Coulomb effect of the learning factor to efficiently reduce those two dominant problems and successfully form the topologically correct map. Provided simulation results show the great improvement and excellent performance achieved by the proposed algorithm.

1. Introduction

The self-organizing feature maps were initially introduced by Kohonen [1][2] attempting to simulate the formation of topology-preserving neural projection found in various area of the brain. The well-known architecture and algorithm of self-organizing maps has the important property of automatic formation of topologically correct maps of various features of input signals and their abstractions. By using this important property, the self-organizing maps has been particularly successful in various pattern recognition tasks involving very noisy signals with acceptable computation complexity, such as speech recognition [3], robotic control [4], optimization problem [5][6], etc.

The network used in the self-organizing model consisting of interconnected adaptive PEs (cells) whose connection arrangement is generally one-dimensional or two-dimensional (rectangular, hexagonal, etc.), and the input signal (vector) is assumed to be fully connected in parallel to all the cells in the network. Every cell has a reference weight vector, and the output response is obtained from the cell where its weight vector get a best matching to the input vector, so this network performs a dimensionality reduction mapping of input vectors (generally in high dimensional space) onto a low-dimensional (usually one or two-dimensional) discrete lattice of PEs, or cells.

The learning algorithm of self-organizing maps is a kind of competitive learning which have all the cells in a neighborhood of the best matching cell (winner) reinforced as training proceeds. The reinforcement is accomplished by moving the weight vectors of these cells toward the input vector. It has been verified that with proper settings of neighborhood range and adjusting rate, the network will converge to a stable equilibrium to form the final mapping. Also, many simulation experiments show that in the learning process of the self-organizing maps, the location of the cell in a network where the response is maximum (best matching) becomes specific to a certain characteristic or feature in the set of input signals with the same (correct) topological orders which is presented in the relations of the input signal patterns, so the resulted mapping is able to preserve the topological relations.

There are some modifications have been suggested to self-organizing maps, but in stead of fitting it to some application, here we present the Kohonen's feature maps problem in a more general way. As in the self-organizing maps, those topological orders are not all "correct" or "complete" corresponding to the feature relations of input signals as discussed in section 2. As a remedy, we propose a combined algorithm in section 3 which induces the procedure of insertion and deletion of cell and applies Coulomb effect to the weight updation in order to give much more chance for each PE to characterize the distribution of the original event properly. In section 4 we provide some simulation results that show the performance of the modified algorithm on several simple examples. Finally, a brief conclusions are summarized in section 5.

2. Learning in the Self-organizing Maps

The learning procedure of a self-organizing map consists of two steps: (a) to find the best matching cell (winner) (b) to update the weights of the winner and its neighbors [7]. we summarize these two steps as follows:

a. Given a sequence of randomly chosen input samples X(k) from the input vector space, the matching measurement is generally the Euclidean distance between the weight vector of a cell and the present input vector

$$\|X(t)-W_k(t)\| = \min_i \|X(t)-W_i(t)\| \quad i=1,2,..,M \quad (1)$$

Where t is the discrete time index for input sequence, $W_i(t)$ is the weight vector of cell i, $W_k(t)$ is that of the best matching cell, and M is the number of cells in the network.

b. The weights of the winner and its neighbors are updated as follows

$$W_i(t+1) = W_i(t) + L_k(i)a(t)[\, X(t) - W_i(t) \,] \quad i=1,2,...,M \quad (2)$$

Where a(t) is the learning factor which is monotonically decreasing with time index t. $L_k(i)$ is the lateral interaction function which define the neighborhood of the winning cell k

0-7803-0593-0/92 $3.00 1992 IEEE

and the relative strength of adjustment of the winner and its neighbors. It requires that

$$L_k(i) \begin{cases} =1 & \text{for } i=k \\ <=1 & \text{for } i \in N_k \\ =0 & \text{otherwise} \end{cases}$$

In most cases, $L_k(i)=1$ for all $i \in N_k$, where N_k is a neighborhood of cell k which is symmetric defined around the cell k and shrink monotonically with time index t. This procedure makes the winner and its neighbors to learn the mean average and results a lateral interaction between cells in the network, to perform self-organization

As the learning proceeds the winner's location in the network becomes specific to a certain characteristic or feature in the set of input signals with the same topological orders in the original relations of input signals. Unfortunately, some times these topological orders on the map are not all "correct" or "complete" corresponding to the relations of input signals. For some topological orders on the map, there is no such feature relations in the input signals (incorrect), or some "correct" topological orders corresponding to actual feature relations of input signals are "lost" in the mapping (the map fail to preserve these topological orders - incomplete). For this reason, it is more proper to call it a "piecewise correct" map.

For the ease of understanding, we cite a simple example which takes a one-dimensional network to "learn" the topology-preserving mapping of an distribution of input signal in two-dimensional vector space. As we can see from Fig-1, there are both the "incorrect" topological relations and the "lost" topological relations corresponding to the original signals. For the display purpose, the input vector distribution and the weight vectors of cells are plotted together with the alphabet shape standing for uniform input distribution, dots for weight vectors and lines between dots for spatial orders on the map.

Further we found that, when the input distribution is a convex region, there is no "incorrect and lost" problem, but when the input distribution is a concave region especially when the distribution region is getting more spatially complex, the "incorrect" and "lost" situations are getting serious, even the feature map becomes poorly-organized because of the fragmentation due to too many "incorrect" relations.

This phenomenon can be explained as follows: Before the network learning begins, the initial weight vectors of cells are preset to distribute in the input vector space randomly or in a manner we prefer, anyway the initial topological orders of these weight vectors have already been determined in the beginning of learning . With "the winner takes all" learning rules, as the competition learning begins, the starting "positions" of the weight vectors in the input space will influence the competition results so as to determine the initial "moving directions" of weight vectors, therefore also limit the outcome of the feature maps. Once, Kohonen made the point that if the self-organizing map has any relevance to biology, it tells why it is so important for genetics to provide a basically correct topological lay-out for the nerve axon fascicles which later self-organize into precise topological mappings.[8] So the initial weight is a dominant factor to the development of the feature maps.

On the other hand, as the learning proceeds, the sampled input vectors are "fed" to the network one by one for learning as a sequence of time index. So, for fixed distribution samples, the order of the input sequence will also limit the competition, and the moving directions of weight vectors. Different order may even induces the totally different results.

The two factors mentioned above are combined to dominate the way which the network "fits" the input distribution. Kohonen has proved that it will certainly converge to fit the input distribution [2], but it was mentioned also by Hecht-Nielsen that in practice it is "lack of conformity" to the desired probability density function [9], and it is often out of luck to fit the distribution well. If we could preset the weight to have orders in some way similar to those of the input distribution, or if we could know the importance of the input samples in order to design a proper ordering sequence in advance, we would have much more chance to fit it well. But unfortunately these are impossible in practical, as we generally can not have any information about the characteristic of the distribution of input data. Whereas, the work we are dedicated can improve these two problem as describing in the following.

3. Modified Learning Algorithm

In point to the "piecewise-correct" mapping problem, the work here we intended to do is to apply some additional algorithm and architecture to the original ones of Kohonen's self-organizing maps in order to reduce the probabilities of "incorrect" and "lost" phenomenon in the resulted maps. For this purpose, we add two procedures to the learning rules of the self-organizing maps.

a. Insert and Delete

This algorithm was first introduced by B. Angeniol's group [5], but we make a little modifications here. In the competitive learning of self-organizing maps, we adopt a network (for simplified example, a linear array) with a dynamic size. Initially the network has only a few (generally one or two) cells. As the learning begins, we count the times it wins in the competition (be a winner) for every cell in each learning turn (with all input samples presented).

Insert: When the winning times of a cell excesses an upper-bound, one cell (or a few cells) of PE is inserted to its nearest neighborhood with the weight vector being set to all most the same of the original cell (except a little shift from the latter in the expanding direction of the network or just in a random direction).

Delete: When the winning times of any cell during a full learning turn is beyond a lower-bound, we delete it from the network.

As the learning begins, every time a cell respond to too many input samples, it makes a new copy of itself to "share" its "burden", so the network "grow" from a small size to a larger size by cell insertion, also some redundant cells (cells that are seldom win) are deleted and finally a stable network size is reached with those cells which win steadily. Since there are only a few cells in the network initially, and the weights of those sucessively inserted cells are determined by the former. So, the initial weight problem mentioned in previous section will be greatly reduced. On the other hand, those cells of finally resulted network will have about the same winning times (between two bounds). This means that each of them maps a roughly equiprobable region of input distribution. This has the same effect with conscience mechanism [10].

b. Coulomb Effect

Coulomb effect has been combined with neural network in some research [11]. According to the Coulomb law, the attractive force between two charges with opposite polarity is inverse proportional to the square of the distance between these two charges. Here we apply this law to the learning factor a(t). The modified learning factor would be :

$$a^*(t) = \frac{a(t)}{C(t)\,\|X(t) - W_k(t)\|^2},$$

Where C(t) is the Coulomb effect coefficient function which is monotonically decreasing with time index t.

To avoid infinite value resulted from division by zero, we further make a little modification :

$$a^*(t) = \frac{a(t)}{[\,C(t)\,\|X(t) - W_k(t)\|^2 + 1\,]} \quad . \tag{3}$$

So the weight updating rule in equ.(2) becomes

$$W_i(t+1) = W_i(t) + L_k(i)a^*(t)[\,X(t) - W_i(t)\,] \qquad i=1,2,\ldots,M \tag{4}$$

With the coulomb effect, the learning factor would be much stronger when the present input vector are getting closer to the winner's weight vector, so those weights of the input vectors which are near to the winner's weight are much favorable in the updation procedure, e.g. the cells of the network would learn the "nearer" input patterns faster than others. Since the nearer inputs have more influence on the winner and its neighbors, so the sequence order of the feeding input samples to the network becomes less important. Therefore the influence of the input sequence order mentioned in previous section is also greatly reduced.

The total effect of combining these two algorithms is actually to make the network "grow along the distribution" to form the correctly topology-preserving maps, so it is understandable that the network would get much more chance to "fit" the distribution region of input vectors and have better exemption from topologically "incorrect" connections. As a consequence, it becomes more adaptive to complex feature distribution. Using these two strategies, we could efficiently alleviate the initial-weight problem as well as the input-sequence-order problem. For the ease of presentation, some simulation experiments are also presented to verify the performance in the next section.

4. Simulation Results

In a series of computer simulations, performance of the proposed algorithm is examined for several input distributions of various complexity in this section. Also, a comparison of the results of the proposed algorithm with those of the original one is provided to illustrate the improvement.

For the ease of discussion, we adopt one-dimensional networks to learn the mapping from two-dimensional vector space. The input space is assumed to be a unit square, and the input samples are uniformly distributed in the alphabet shaped region within the unit square. In all the figure of results, the alphabet shaped region stands for the input distribution, the weight vectors' positions in the input space are marked by dots, and those lines between dots stand for the neighborhood relations of cells .

The lateral interaction function Lk(i) we use in all simulation are chosen to be a Gaussian function center on cell k. Where

$$L_k(i) = \begin{cases} \exp.\left[-\left(\dfrac{2*|i-k|^2}{R(t)}\right) \right] & \text{for } |i-k| <= R(t), \\ 0 & \text{otherwise ;} \end{cases}$$

where R(t) is the radius of the neighborhood region which is monotonically decrease with time index. Lk(i) is an approximation of the "Mexican hat" shaped function introduced by Kohonen for lateral interaction.[1]

In the first part of simulations, we use two one-dimensional networks (one has a fixed size and the other has a dynamic size) to present the time history of the learning processes of both the modified algorithm and the original one in the case that the original algorithm usually get failed in the mapping of this input distribution.

In these two learning processes, we use different initial values of the learning factor. A larger value is used in the modified case because the Coulomb effect will actually reduce the average strength of the learning factor. The initial value of each parameter in both algorithm is selected by trial to make the results as good as possible. So these two processes are both representative examples.

With the comparison of Fig-5 and Fig-6, it is found that the modified algorithm really makes the network growing along the distribution region, while the original algorithm just have the network swell out until it is "attached" to the input distribution region, so the ill-mapping results.

In the second part of simulations, we have both of two networks with different algorithm learn various input distribution. In the learning of every input distribution we take 100 trials with different input sequence order and initial weights for each network, and count the times of case in which they successfully learn the mapping. Fig-2 to Fig-4 show the successful result of each distribution and table-1 list the counting results. Again, the initial value of each parameter in both algorithm is chosen by trial to make the result as good as possible.

The results in table-1 show the performance improvement achieved by the modified algorithm in these simulation experiment. On the other hand, the counting times in which it succeeds with modified algorithm in those four input distribution regions tend to go down while the input distribution region gets "sharper" folds (with larger curvature, for example a "w" shaped distribution region). This is because the sharp folds of distribution region causes confusion when the network grow along the input distribution region. But anyway, the proposed algorithm has a much better performance than the original algorithm.

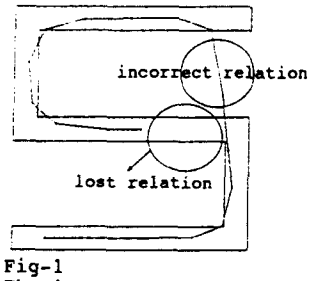

Fig-1
The incorrect and the lost phenomenon in the feature map.

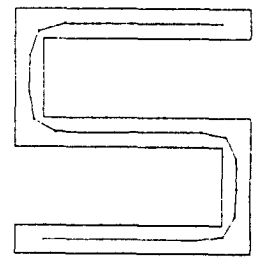

Fig-2
The correct map in the "S" shaped distribution

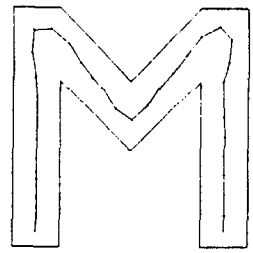

Fig-4
The correct map in the "M" shaped distribution

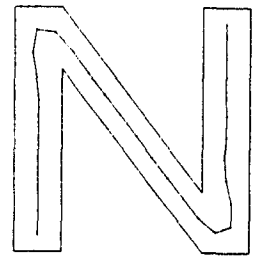

Fig-3
The correct map in the "N" shaped distribution

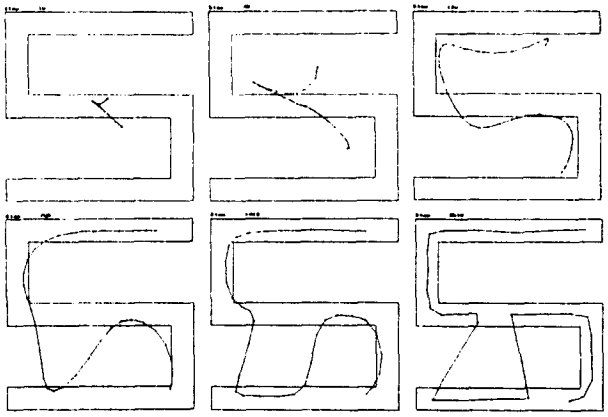

Fig-5 Learning time history of the original algorithm with the initial values: a(0)=0.1, R(0)=8, M=36, number of input samples=1092.

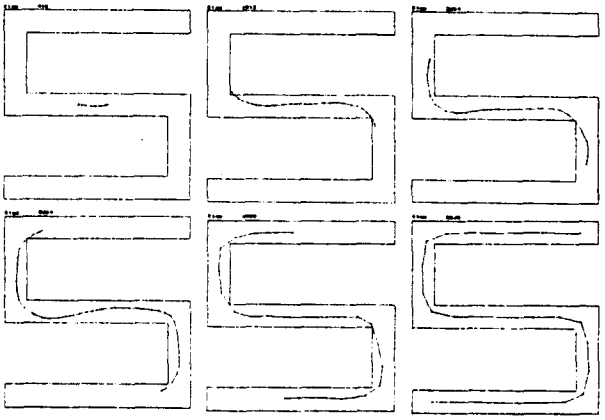

Fig-6 Learning time history of the modified algorithm with the initial values: a(0)=0.8, R(0)=8, C(0)=1000, upper-bound=60, lower-bound=30, final number of cells M=27, number of input samples=1092.

Times / Algorithm	Distrib. / Initial	"S" shaped samples' no.=1092	"M" shaped samples' no.=1464	"N" shaped samples' no.=1374
Original	a(0)=0.1 R(0)=10 M=36	7	46	14
Modified	a(0)=0.9 R(0)=6 C(0)=1000	100	66	63

Table-1 The successful times of both algorithm counted in 100 trials in mapping three different distribution

5. Conclusion and Discussion

This paper presents a modified algorithm for the learning procedure of the self-organizing topology-preserving maps to improve the "piecewise-correct" problem which happened frequently in the original self-organizing maps. The "piecewise-correct" problem are generally caused by two dominant factors existing in the learning procedure of the original algorithm. The one is the initial-weight problem and the other is the input-sequence-order problem. While our algorithm can efficiently reduce the influence of these two factors and successfully "guide" the network to form a topologically correct map.

In the new proposed algorithm, we adopts a dynamic network which allows cells to be inserted and deleted and it adds the Coulomb effect to the learning factor. The insertion and deletion procedures are controlled by the upper-bound and lower-bound of the winning times to try to keep all cells mapping a roughly equiprobable region of the input distribution, and since most cells' initial weights are assigned to those of their "father" cells, the initial weight problem becomes no more serious. While the Coulomb effect adaptively changes the value of the learning factor to have all cells adapt themselves more to the "nearer" input pattern, therefore it reduce the importance of input sequence order.

Simulation results indicate excellent performance of the modified algorithm in the learning of the mapping of two-dimensional input vector distribution using a one-dimensional network. In the learning process the network is guided to trace the distribution instead of just blindly matching it. In the simulation results show, a great improvement has been achieved by the modified algorithm. Moreover, the distribution tracing of the modified algorithm may also fail as the input distribution region gets "sharp" folds which easily causes a confusion in topological relations.

Here we would like to point out that the selection of initial values of all parameters in these simulations are made by experience, but since the example is a typical case with input space normalized to unit space (square, cube or hypercube), so these initial values used here can be treated as typical values for other cases. Whereas, some modifications might be needed for insertion and deletion procedures in the learning of a two-dimensional network.

Many application using self-organizing maps have been succeeded because of the important property of automatic formation of topologically correct maps. Since the proposed algorithm can greatly improve the correctness of the mapping, so we can get a better performance via our algorithm in many applications.

Reference

[1] T. kohonen, "Self-organized formation of topologically correct feature maps," Biol. Cybern., vol.43, pp.59-69, 1982.

[2] --------, "Self-organization and Associative Memory," Springer-Verlag, New York, 1984.

[3] --------, "The 'neural' phonetic typewriter," Computer, vol.21, pp.11-22, March 1988.

[4] H. Ritter and K. Schulten, "Extending Kohonen's self-organizing mapping algorithm to learn ballistic movements," NATO ASI Series, vol.F41, pp. 393-406, 1988.

[5] B.Angeniol, G. del la Croix Vaubois, and J.-Y. Le Texier, "Self-organizing feature maps and the travelling salesman problem," Neural Networks, vol.1, pp.289-293, 1988.

[6] Chau-Yun Hsu, M-H Tsai and W-M Chen,"A study of feature- mapped approach to the multiple travelling salesmen problem," Proc. IEEE Int Symp. of Circuit and System, vol.3 pp.1589-1592, Tvme, 1991.

[7] T. Kohonen, "The self-organizing map," Proc. IEEE, vol.78, no.9, pp.1464-1480, Sep. 1990.

[8] --------, movie shown at the 1985 annual meeting of the Optical Society of America in Washington, D.C.

[9] R. Hecht-Nielsen, "Neurocomputing," Addison-Wesley, 1990.

[10] D. DeSieno, "Adding a conscience to competitive learning" Proc. IEEE Int. Conf. on Neural Network, (New-York, July 1988) pp.I117-124.

[11] C.L.Scofield, "Learning internal representations in the Coulomb energy network," Proc. IEE, Int. Conf. on Neural Networks, ICNN-88, pp.I271-276.