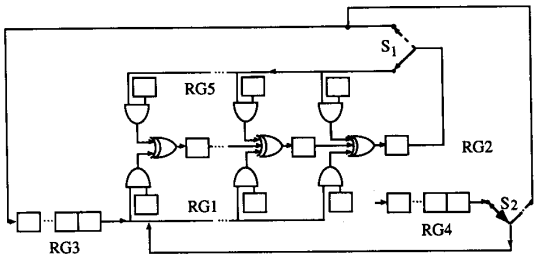

Fig. 5. The terminal circuit corresponding to the circuit of Fig. 4.

and the rows of P^{-1} are generated successively by multiplications of $(\alpha_2)^z$ by itself. ($(\alpha_2)^z$ is a fixed vector associated with y, and can be stored in a ROM.) The operation $\beta \cdot P^{-1}$ is performed by adding the successively generated rows of P^{-1} according to the bits of β.

The circuit of Fig. 4 calculates $(\alpha_2)^x$ in n squaring operations, each one taking $3n$ clock cycles (including loading RG3). The "square" operations then take altogether $3n^2$ clock cycles. The "multiply" operations take *altogether* a negligible $n/2$ extra cycles, on the average. The circuit of Fig. 5 calculates $[(\alpha 1)^x]^y$ in n squaring operations and $n/2$ multiplications, each operation taking $2n$ clock cycles (including the loading of RG3). The rows of P^{-1} are generated by extra $n-2$ multiplications and up to $n-1$ additions. The entire operation takes $5n^2$ clock cycles. Had the circuit of Fig. 5 been based on that of Fig. 3, it would have yielded its final results within $6n^2$ clock cycles.

c) Threats and Limitations: The public key used by a terminal is the minimum polynomial f_2 of $(\alpha_1)^y$, where it is required that $\gcd(2^n-1, y) = 1$ (in order for f_2 to be primitive.). As security considerations require that $2^n - 1$ should be a Mersenne prime, or at least have a large prime factor, there is in practice no limitation on the number of possible different keys used by a terminal. Also, there are n private keys y_i associated with the public key f_2, which also does not pose a threat, as $(2^n/n) \approx 2^n$ for a large n.

Since the card cannot validate the authenticity of the public key f_2 sent by the terminal, it cannot be sure whether f_2 is primitive. (If the card has to verify the primitivity of f_2, it would be better to return to the original schemes.) This poses a possibility that a dishonest terminal gains knowledge (complete or partial) of the secret key x of the card. The terminal can send the card a polynomial f_2 having a low periodicity. (That is, successive powers of its root α_2 form a short cycle.) If the card sends back $(\alpha_2)^x$, the terminal can reconstruct x from various polynomials f_2 having different periodicities, based on the Chinese Remainder Theorem. The proposed scheme is then applicable in those cases where the key $(\alpha_2)^x$ calculated at the card is not sent back to the terminal.

REFERENCES

[1] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory,* vol. IT-22, pp. 644–654, 1976.
[2] T. El-Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inform. Theory,* vol. IT-31, pp. 469–472, 1985.
[3] T. Beth *et al.*, "Architectures for exponentiation in GF(2^n)," in *Advances in Cryptology—CRYPTO '86,* LNCS 263, pp. 302–310.
[4] T. Beth and D. Gollman, "Algorithm engineering for public key algorithms," *IEEE J. Select. Areas Commun.,* vol. SAC-7, pp. 458–466, 1989.
[5] P. A. Scott *et al.*, "Architectures for exponentiation in GF(2^m)," *IEEE J. Select. Areas Commun.,* vol. SAC-6, pp. 578–586, 1988.
[6] C. C. Wang and D. Pei, "A VLSI design for computing exponentiations in GF(2^m) and its application to generate pseudorandom number sequences," *IEEE Trans. Comput.,* vol. 39, pp. 258–262, 1990.
[7] G. B. Agnew *et al.*, "Fast exponentiation in GF(2^n)," in *Advances in Cryptology—EUROCRYPT '88,* LNCS 330, pp. 251–256.
[8] J. J. Quisquater and M. De Soete, "Speeding up smartcard RSA computations with insecure coprocessors," in *Proc. SMART CARD 2000,* Amsterdam, Oct. 1989.
[9] S. Kawamura and A. Shimbo, "Performance analysis of server-aided secret computation protocols for the RSA cryptosystem," *Trans. IEICE,* vol. E73, pp. 1073–1080, 1990.
[10] W. W. Peterson and E. J. Weldon, *Error-Correcting Codes,* 2nd ed. Cambridge, MA: M.I.T. Press, 1972.
[11] J. L. Massey and J. K. Omura, "Computational method and apparatus for finite field arithmetic," U.S. Patent 4587627.
[12] C. C. Wang *et al.*, "VLSI architectures for computing multiplications and inverses in GF(2^m)," *IEEE Trans. Comput.,* vol. C-34, pp. 709–716, 1985.
[13] B. Arazi, "Connection between primitive generators of GF(2^n)," *Electron. Lett.,* vol. 16, pp. 223–225, 1980.
[14] J. A. Gordon, "Very simple method to find the minimal polynomial of an arbitrary nonzero element of a finite field," *Electron. Lett.,* vol. 12, pp. 663–664, 1976.

Comparisons of Seven Neural Network Models on Traffic Control Problems in Multistage Interconnection Networks

Nobuo Funabiki, Yoshiyasu Takefuji, and Kuo Chun Lee

***Abstract*— This paper presents performance comparisons of seven neural network models on traffic control problems in multistage interconnection networks. The decay term, three neuron models, and two heuristics were evaluated. The goal of the traffic control problems is to find conflict-free switching configurations with the maximum throughput. Our simulation results show that the hysteresis McCulloch–Pitts neuron model without the decay term and with two heuristics has the best performance.**

***Index Terms*—Decay term, hysteresis McCulloch–Pitts neuron model, McCulloch–Pitts neuron model, multistage interconnection network, neural network, optimization, parallel algorithm, sigmoid neuron model.**

I. INTRODUCTION

Neural networks for solving optimization problems have been extensively studied since Hopfield and Tank introduced them [1]–[9]. A neural network has a large number of neurons or processing elements where neuron $\#i$ has an input U_i and an output V_i. The goal of the neural network is to minimize the energy function $E(V_1, V_2, \cdots, V_n)$ by using the motion equation. The energy function represents the objective function and the constraints of the problem.

Manuscript received January 4, 1991; revised July 26, 1991 and April 21, 1992.

N. Funabiki is with the System Engineering Division, Sumitomo Metal Industries, Ltd., Amagasaki 660, Japan.
Y. Takefuji is with the Department of Electrical Engineering and Applied Physics, Case Western Reserve University, Cleveland, OH 44106, and with the Faculty of Environmental Information, Keio University, Japan.
K. C. Lee is with the R&D Department, Cirrus Logic Inc., Fremont, CA 94538.

IEEE Log Number 9204394.

Hopfield proposed that the motion equation is composed of the partial derivative term of the energy function as the gradient descent method, and the decay term with a time constant τ [1]:

$$\frac{dU_i}{dt} = -\frac{\partial E(V_1, V_2, \cdots, V_n)}{\partial V_i} - \frac{U_i}{\tau}. \quad (1)$$

Takefuji *et al.* showed that the decay term increases the energy function under some conditions [4]. They modified the motion equation in order to guarantee the local minimum convergence:

$$\frac{dU_i}{dt} = -\frac{\partial E(V_1, V_2 \cdots, V_n)}{\partial V_i}. \quad (2)$$

The following three neuron models have been used for optimization problems:

1) the sigmoid neuron model [1]:

$$V_i = \frac{1}{2}\left[1 + \tanh\left(\frac{U_i}{U_0}\right)\right] (U_0 \text{ is a constant parameter}) \quad (3)$$

2) the McCulloch–Pitts neuron model [10]:

$$V_i = 1 \text{ if } U_i > 0, \text{ and } V_i = 0 \text{ if } U_i \leq 0. \quad (4)$$

3) the hysteresis McCulloch–Pitts neuron model [5]:

$$V_i = 1 \text{ if } U_i > \text{UTP}, V_i = 0 \text{ if } U_i < \text{LTP},$$

$$\text{and } V_i = \text{unchanged if LTP} \leq U_i \leq \text{UTP}$$

(UTP and LTP are constant parameters satisfying UTP > LTP.) (5)

However, under certain conditions, for example, when a small value such as 0.02 is used for U_0 in the sigmoid neuron model [1], it behaves like the McCulloch–Pitts neuron model. The McCulloch–Pitts neuron model sometimes introduces undesirable oscillatory behavior. Therefore, the hysteresis McCulloch–Pitts neuron model was proposed, and it has been empirically shown to suppress the oscillation and to shorten the convergence time [5].

Because only the local minimum convergence is guaranteed in neural networks [4], the hill-climbing heuristic and the omega function heuristic have been introduced to improve the global minimum convergence [6]. The two heuristics make it easier for the state of neural networks to escape local minima, and increase the frequency of the global minimum convergence.

In this paper, we have compared the performance of the following seven neural network models on traffic control problems in multistage interconnection networks, in order to reevaluate the variations of neural networks mentioned above:

1) case #1: the McCulloch–Pitts neuron model with the decay term ($\tau = 2$)

2) case #2: the McCulloch–Pitts neuron model with the decay term ($\tau = 5$)

3) case #3: the McCulloch–Pitts neuron model with the decay term ($\tau = 10$)

4) case #4: the McCulloch–Pitts neuron model without the decay term

5) case #5: the sigmoid neuron model without the decay term

6) case #6: the hysteresis McCulloch–Pitts neuron model without the decay term

7) case #7: the hysteresis McCulloch–Pitts neuron model without the decay term and with the two heuristics mentioned above

Since Goke *et al.* defined the class of Banyan networks [11], multistage interconnection networks have been widely used in telephone networks, parallel processing computers, and integrated service digital networks (ISDN) [11]–[18]. The $n \times n$ network consists of $(\log_2 n)$ switching stages. Each stage consists of $n/2$ 2 × 2 crossbar switching elements. Several topologically equivalent variations have been proposed such as data manipulator [12], omega network [13], flip network [14], indirect binary n-cube network [15], regular Banyan network, baseline network, and reverse baseline network [17].

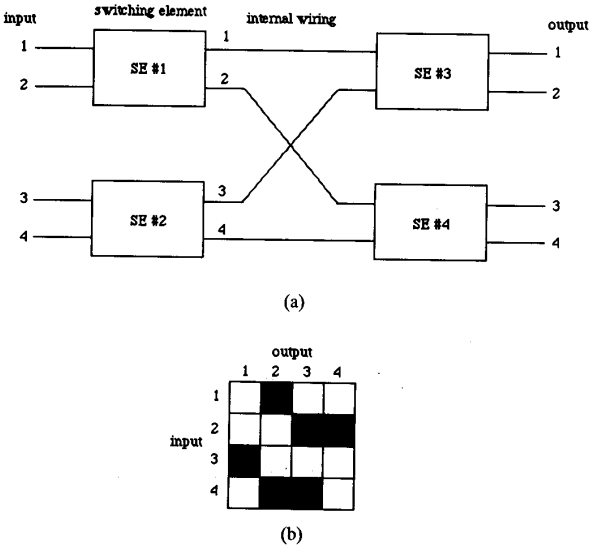

Fig. 1. A traffic control problem in a 4 × 4 reverse baseline network. (a) A 4 × 4 reverse baseline network. (b) A 4 × 4 traffic matrix.

For this simulation, we selected the reverse baseline network, shown in Fig. 4, with three constraints: 1) any input can be simultaneously connected with at most one output (input blocking constraint), 2) any output can be simultaneously connected with at most one input (output blocking constraint), and 3) any internal wire can be simultaneously used by at most one input-output connection (internal blocking constraint) [3]. The network is assumed to have buffers for switching configurations, and to be operated synchronously [18]. Transmission demands through the $n \times n$ network are represented by an $n \times n$ traffic matrix. Each element t_{ij} represents whether input $\#i$ and output $\#j$ need to be connected ($t_{ij} = 1$) or not ($t_{ij} = 0$). The traffic control problem is to select the maximum number of demands satisfying the constraints from the given traffic matrix. Fig. 1 shows the 4 × 4 reverse baseline network, and the traffic matrix where black squares indicate one elements and white squares zero elements.

II. Neural Network Representation

We use the two-dimensional neural network composed of n^2 neurons for the $n \times n$ reverse baseline network problem. The output V_{ij} of neuron $\#ij$ in binary neuron models represents whether the demand t_{ij} is selected ($V_{ij} = 1$: nonzero output) or not ($V_{ij} = 0$: zero output). In the sigmoid neuron model, we consider $V_{ij} \geq 0.9$ as the nonzero output and $V_{ij} < 0.9$ as the zero output. Fig. 2(a) shows the neural network composed of 16 neurons for the problem in Fig. 1. Fig. 2(a) also shows the global minimum solution where black squares indicate nonzero outputs and white squares zero outputs. The corresponding switching configuration is shown in Fig. 2(b).

The nonnegative energy function representing the three constraints in Section I is given by

$$E = \frac{A}{2}\sum_{i=1}^{n}\left(\sum_{k=1}^{n} V_{ik} - 1\right)^2 + \frac{A}{2}\sum_{j=1}^{n}\left(\sum_{k=1}^{n} V_{kj} - 1\right)^2 + B\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{\substack{p=1 \\ p\neq i}}^{n}\sum_{\substack{q=1 \\ q\neq j}}^{n} s_{ijpq} V_{ij} V_{pq} \quad (6)$$

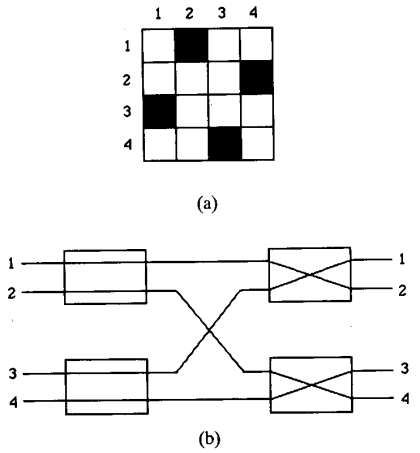

Fig. 2. Neural network representation for the problem in Fig. 1. (a) A 4×4 neural network for the problem in Fig. 1. (b) The corresponding states of switching elements.

where A and B are constant coefficients, s_{ijpq} is 1 if the connection between input $\#i$ and output $\#j$ shares an internal wire with that between input $\#p$ and output $\#q$, and s_{ijpq} is 0 if not. The first term represents the input blocking constraint. It is zero if and only if one demand is selected for every input. The second term represents the output blocking constraint. It is zero if and only if one demand is selected for every output. The third term represents the internal blocking constraint. It is zero if and only if any two selected connections share no internal wire. Note that s_{ijpq} is calculated by the following two-step procedure: 1) the internal wire number W_{kij} of the connection between input $\#i$ and output $\#j$ on stage $\#k$ is given by

$$W_{kij} = 2^k \left(\left\lceil \frac{i}{2^k} \right\rceil - 1 \right) + \left\lceil \frac{j}{\left(\frac{n}{2^k}\right)} \right\rceil \tag{7}$$

where the function $\lceil x \rceil$ gives the minimum integer more than or equal to x, and 2) s_{ijpq} is given by

$$\begin{aligned} s_{ijpq} &= 1 \text{ if there exists } k \epsilon \{1, \cdots, (\log n - 1)\} \\ &\quad \text{such that } W_{kij} = W_{kpq} \text{ for } i \neq p \text{ and } j \neq q \\ &= 0 \text{ otherwise.} \end{aligned} \tag{8}$$

Therefore the motion equation with the decay term for neuron $\#ij$ is given by

$$\frac{dU_{ij}}{dt} = -A\left(\sum_{k=1}^{n} V_{ik} - 1\right) - A\left(\sum_{k=1}^{n} V_{kj} - 1\right) - B \sum_{\substack{p=1 \\ p \neq i}}^{n} \sum_{\substack{q=1 \\ q \neq j}}^{n} s_{ijpq} V_{pq} - \frac{U_{ij}}{\tau}. \tag{9}$$

In 1990, Brown *et al.* proposed the Hopfield neural network application for Banyan networks, with the sigmoid neuron model and the decay term [3]. Unfortunately, they did not discuss the time complexity and the convergence frequency of the neural network, which are always controversial.

III. Two Heuristics for the Global Minimum Convergence

We have examined the following two heuristics in this paper.

1) The hill-climbing heuristic:

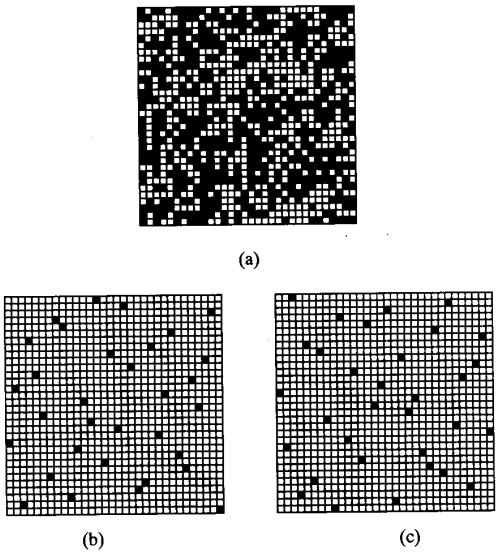

Fig. 3. A 32×32 traffic matrix with 50% density and two global minimum solutions. (a) A 32×32 traffic matrix with 50% density. (b) The global minimum solution #1. (c) The global minimum solution #2.

The following two functions are added to the motion equation:

$$+Ch\left(\sum_{k=1}^{n} V_{ik}\right) + Ch\left(\sum_{k=1}^{n} V_{kj}\right) \tag{10}$$

where $h(x)$ is 1 if $x = 0, h(x)$ is 0 if $x \neq 0$, and C is a constant coefficient. The hill-climbing heuristic helps the neural network to escape the local minimum, by encouraging neuron $\#ij$ to have nonzero output when no neuron for input $\#i$ and/or for output $\#j$ has nonzero output.

2) The omega function heuristic:

The following two forms of B-term in the motion equation are periodically used:

$$\begin{aligned} &\text{if } (t \bmod T_0) < \omega, \text{then } (B - \text{term}) \\ &= -B \sum_{\substack{p=1 \\ p \neq i}}^{n} \sum_{\substack{q=1 \\ q \neq j}}^{n} s_{ijpq} V_{pq} V_{ij}, \text{ else } (B - \text{term}) \\ &= -B \sum_{\substack{p=1 \\ p \neq i}}^{n} \sum_{\substack{q=1 \\ q \neq j}}^{n} s_{ijpq} V_{pq} \end{aligned} \tag{11}$$

where t is the number of iteration steps, and T_0 and ω are constant parameters. The omega function heuristic makes the local minimum shallower, so the state of the neural network can easily escape it.

IV. Simulation Results and Discussion

We have developed a simulator based on parallel procedures proposed in [4]–[9]. The seven neural network models in Section I were simulated in this simulator for the performance comparison. In this simulator, the following set of parameters was adopted:

$$\begin{aligned} &A = B = 1, C = 2, UTP = 5, LTP = -5, U_0 = 0.02, \\ &T_0 = 10, \omega = 5, T\_\max 1 = 500, \text{ and } T\_\max 2 = 1000 \end{aligned} \tag{12}$$

where $T\_\max 1$ is the maximum number of iteration steps for the global minimum convergence, and $T\_\max 2$ is that for the local

TABLE I
SUMMARY OF SIMULATION RESULTS FOR THE 50% DENSITY TRAFFIC MATRIX PROBLEM

	Convergence to Global Minimum		Convergence to Local Minimum		
	Average Iteration Steps	Frequency	Average Iteration Steps	Frequency	Solution Quality
case #1	—	0%	715.4	34%	1.8
case #2	41.2	25%	363.8	82%	24.4
case #3	77.4	41%	309.0	90%	27.1
case #4	286.5	40%	423.0	99%	29.7
case #5	284.0	42%	418.3	99%	29.9
case #6	307.2	45%	417.9	100%	30.9
case #7	224.0	87%	261.1	100%	31.8

TABLE II
SUMMARY OF SIMULATION RESULTS FOR THE 80% DENSITY TRAFFIC MATRIX PROBLEM

	Convergence to Global Minimum		Convergence to Local Minimum		
	Average Iteration Steps	Frequency	Average Iteration Steps	Frequency	Solution Quality
case #1	—	0%	728.6	52%	1.3
case #2	34.5	4%	461.7	48%	13.9
case #3	41.9	31%	315.8	77%	23.5
case #4	181.6	86%	226.3	100%	31.7
case #5	184.3	89%	219.1	100%	31.8
case #6	198.6	96%	204.9	98%	31.3
case #7	138.3	100%	138.3	100%	32.0

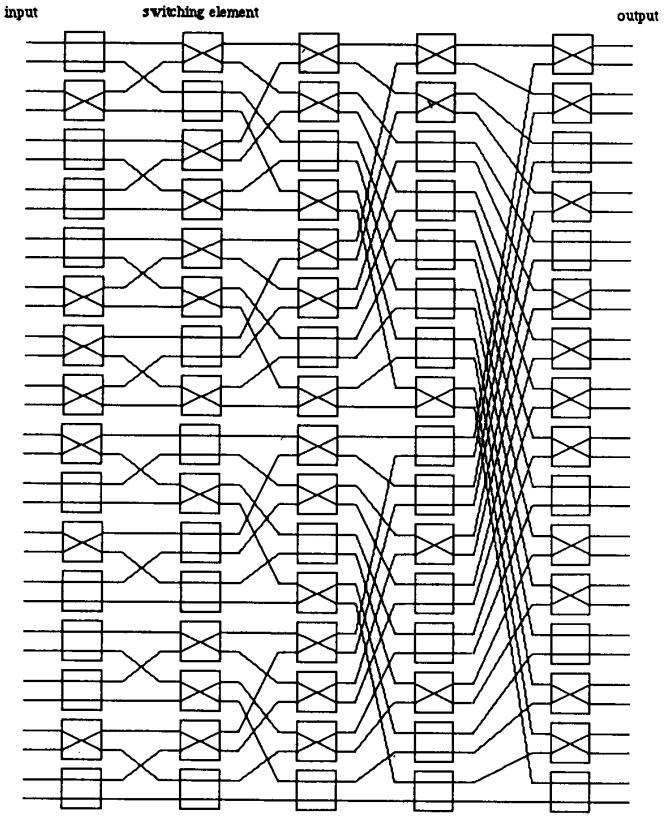

Fig. 4. States of switching elements corresponding to the solution in Fig. 3(b).

minimum convergence. The same set of parameters have been used in several other papers.

Two 32 × 32 traffic matrices with 50% density and 80% density were randomly generated. Fig. 3(a) shows the traffic matrix with 50% density. From different initial values of U_{ij}, our simulator found several global minimum solutions, shown in Fig. 3(b) and (c). Fig. 4

shows the switching configuration corresponding to Fig. 3(b). Tables I and II summarize the simulation results, where the average number of iteration steps required for the convergence and the convergence frequency, and the average number of demands in the local minimum solutions are compared in seven models. Note that for each model, 100 simulation runs were performed from different initial values of U_{ij}.

We conclude the simulation results as follows:

1) The comparisons of cases #1–#4 show that the decay term disturbs the convergence of the neural network to solutions. Although cases #2 and #3 are superior in the average number of iteration steps for the convergence to case #4, they are inferior in the frequency of the local minimum convergence and the solution quality to case #4. The decay term seems to make the local minimum deeper, so some initial states of U_{ij} can be quickly converged to the global minima.

2) The comparisons of cases #4 and #5 show that the McCulloch–Pitts neuron model and the sigmoid neuron model have similar performance in terms of the average number of iteration steps for the convergence and the convergence frequency. However, because of the exponential calculation in the sigmoid neuron model, it requires much longer computation time than the McCulloch–Pitts neuron model on a digital computer. The simple McCulloch–Pitts neuron model is superior to the sigmoid neuron model for practical uses.

3) The comparisons of cases #4–6 show that the hysteresis McCulloch–Pitts neuron model is superior to the McCulloch–Pitts neuron model and the sigmoid neuron model in terms of the frequency of the global minimum convergence.

4) The comparisons of cases #6–7 show that the two heuristics increase the frequency of the global minimum convergence, reduce the number of iteration steps for the convergence, and improve the solution quality. The hysteresis McCulloch–Pitts neuron model without the decay term and with the two heuristics provides the best performance among the seven models. We have observed similar behavior in other instances.

V. Conclusion

This paper presents performance comparisons of seven neural network models on traffic control problems in multistage interconnection networks. The simulation results show that 1) the decay term in the motion equation disturbs the convergence, 2) with less computation time on a digital computer, the McCulloch–Pitts neuron model achieves the same performance as the sigmoid neuron model, and 3) the hysteresis McCulloch–Pitts neuron model and the two heuristics greatly improve the performance of the neural network computation.

References

[1] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.

[2] A. Marrakchi and T. Troudet, "A neural net arbitrator for large crossbar packet-switches," *IEEE Trans. Circuits Syst.*, vol. 36, no. 7, pp. 1039–1041, July 1989.

[3] T. X. Brown and K. H. Liu, "Neural network design of a Banyan network controller," *IEEE J. Select. Areas Commun.*, vol. 8, pp. 1428–1438, Oct. 1990.

[4] Y. Takefuji and K. C. Lee, "Artificial neural networks for four-coloring map problems and K-colorability problems," *IEEE Trans. Circuits Syst.*, vol. 38, no. 3, pp. 326–333, Mar. 1991.

[5] ——, "An artificial hysteresis binary neuron: A model suppressing the oscillatory behaviors of neural dynamics," *Biol. Cybern.*, vol. 64, pp. 353–356, 1991.

[6] ——, "A parallel algorithm for tiling problems," *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp. 143–145, Mar. 1990.

[7] N. Funabiki and Y. Takefuji, "A parallel algorithm for spare allocation problems," *IEEE Trans. Reliability*, vol. 40, no. 3, pp. 338–346, Aug. 1991.

[8] ——, "A parallel algorithm for solving 'Hip' games," *Neurocomputing*, vol. 3, pp. 97–106, 1991.

[9] K. C. Lee, N. Funabiki, and Y. Takefuji, "A parallel improvement algorithm for the bipartite subgraph problem," *IEEE Trans. Neural Networks*, vol. 3, no. 1, pp. 139–145, Jan. 1992.

[10] W. S. McCulloch and W. H. Pitts, "A logical calculus of ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, p. 115–133, 1943.

[11] L. R. Goke and G. J. Lipovski, "Banyan networks for partitioning multiprocessor systems," in *Proc. 1st. Annu. Int. Symp. Comput. Architecture*, Dec. 1973, pp. 21–28.

[12] T. Y. Feng, "Data manipulating functions in parallel processors and their implementations," *IEEE Trans. Comput.*, vol. C-23, pp. 309–318, Mar. 1974.

[13] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. Comput.*, vol. C-24, no. 12, pp. 1145–1155, Dec. 1975.

[14] K. E. Batcher, "The flip network in STARAN," in *Proc. Int. Conf. Parallel Processing*, 1976, pp. 65–71.

[15] M. C. Pease, "The indirect binary n-cube microprocessor array," *IEEE Trans. Comput.*, vol. C-26, no. 5, pp. 458–473, May 1977.

[16] H. J. Siegel and S. D. Smith, "Study of multistage SIMD interconnection networks," in *Proc. 5th Annu. Int. Symp. Comput. Architecture*, Apr. 1978, pp. 223–229.

[17] C. L. Wu and T. Y. Feng, "On a class of multistage interconnection networks," *IEEE Trans. Comput.*, vol. C-29, no. 8, pp. 694–702, Aug. 1980.

[18] D. M. Dias and M. Kumar, "Packet switching in N log N multistage networks," in *Proc. IEEE GLOBECOM '84*, Nov. 1984, pp. 114–120.

A Neural Network Implementation of the Moment-Preserving Technique and Its Application to Thresholding

Shyi-Chyi Cheng and Wen-Hsiang Tsai

***Abstract*—A neural-network implementation of the moment-preserving technique which is widely used for image processing is proposed. The moment-preserving technique can be thought of as an information transformation method which groups the pixels of an image into classes. The variables in the so-called moment-preserving equations are determined iteratively by a recurrent neural network and a connectionist neural network which work cooperatively. Both of the networks are designed in such a way that the sum of square errors between the moments of the input image and those of the output version is minimized. The proposed neural network system is applied to automatic threshold selection. The experimental results show that the system can threshold images successfully. The performance of the proposed method is also compared with those of four other histogram-based multilevel threshold selection methods. The simulation results show that the proposed technique is at least as good as the other methods.**

***Index Terms*—Connectionist neural networks, gradient descent, image thresholding, moment-preserving principle, recurrent neural networks.**

I. Introduction

Recently, a new image processing technique called *moment preservation* has been successfully applied to many image processing

Manuscript received January 22, 1991; revised August 26, 1991 and April 22, 1992. This work was supported in part by the National Science Council, Republic of China under Contract NSC80–0404–E009–13.

S.-C. Cheng is with the Telecommunications Laboratories, Ministry of Transportation and Communications, Hsinchu, Taiwan, R.O.C.

W.-H. Tsai is with the Department of Computer and Information Science, National Chiao Tung University, Taiwan, R.O.C.

IEEE Log Number 9204393.