A comprehensive survey on two associative memory models

Cheng-Yuan Liou†, Jiun-Wei Liou, Yng-Kae Tzeng
Department of Computer Science and Information Engineering, National Taiwan University

***Abstract*— This paper surveys two advanced associative memory models[8][5]. The first model was derived from the projection on a closed convex set spanned by patterns. The second model was derived from training weights to improve the error tolerance of the Hopfield network. Both models are designed to resolve the insufficiencies of the Hopfield network. These insufficiencies are loading capacity, limit cycles, and stability with respect to noisy patterns.**

***Index Terms*— Neural network, Hopfield model, associative memory, recurrent network**

I. Introduction

Associative memory(AM) is a mechanism to store patterns. The pattern stored in the memory can be recalled even if a portion of this pattern is corrupted. We will discuss binary patterns only. The Hopfield model[2], was designed to realize associative memory collectively. This model carries very rich meaning in both biology and physics. But it has insufficiencies in loading capacity, limit cycles, and stability with respect to noisy patterns. Its capacity is roughly $0.14m$, here m is the total number of bits of the pattern. The projection learning rule[6] was proposed to increase the capacity of the associative memory to m. This rule stores patterns as equilibrium points (or states), but an equilibrium state may not be asymptotically stable. The eigenstructure method[3] was later proposed to store the patterns as asymptotically stable states, but this method produced extra asymptotically stable states. These extra states which are not patterns are called spurious asymptotically stable states. Tao's AM[8] was designed to realize the associative memory without the drawbacks of the projection learning rule and the eigenstructure method. This AM was further improved to serve as a BSB model[10] with much reduced computation.

Tao's AM applied the projection[6] of a continuous-time system[11] on a closed convex set[7]. This set is spanned by all patterns. Tao's AM is governed by an energy function[8]. Those stored patterns have local minimal values of this function.

The Hopfield model is a fully connected network with m neurons and $m \times m$ connection weights. This model can be viewed as an m-D hypercube with m hyperplanes. The weights and threshold of each neuron configure the hyperplane. All binary patterns are stored on certain corners of the hypercube. The weight matrix of the model is a symmetric matrix with 0 values in all diagonal elements, and the threshold of each neuron is also 0. This means that the hyperplane passes the origin. Since its loading capacity is too low, this model may not store all patterns correctly. The etAM (error tolerant associative memory)[5] was proposed to improve its error tolerance and resolve the three insufficiencies. The work[5] also proposed a modified error correction rule (mECR) to improve the ECR model. We review and compare Tao's AM and the etAM in rest sections.

This research was supported by National Science Council under project number NSC 93-2213-E-002-081.

†Correspondent, cyliou@csie.ntu.edu.tw, Dept. of CSIE, NTU, Taipei, Taiwan, 106, ROC.

A. Tao's AM

Tao's AM is based on an energy function and a projection learning rule. It starts with setting closed convex on the projection space. Let $p_1, p_2, \ldots, p_n \in R^m$ be the n patterns (column vectors) that we plan to store in the AM. Let Q be the smallest closed convex set containing all n patterns. Denote $\|\cdot\|$ as the Euclid norm, and $\inf$ as the lower bound of a value. Set a quadratic progamming problem

$$\begin{cases} x \in Q; \\ \min f(x) = \min(-\frac{1}{2}\sum_{i=1}^{m} x_i^2). \end{cases} \quad (1)$$

It is obvious that the solutions of the equation (1) are $p_1, p_2, \ldots, p_n$, which are precisely the patterns. After setting the problem, we define the projection operator. For any x, $x = (x_1, x_2, \ldots, x_m)^T \in R^m$, there exists a point (or state) $P(x) \in Q$ such that

$$\|x - P(x)\| = \inf_{y \in Q} \|x - y\|.$$

$P(x)$ is defined as the projection of x on Q. A dynamical system is giving as

$$\frac{dx}{dt} = P(x - f'(x)) - x. \quad (2)$$

Let D^m be the space, $D^m = \{x | 1 = x_i = -1, \forall i = 1, 2, ..., m\}$. The energy function $E(x) : D^m \to R$ is defined as

$$E(x) = \begin{cases} f(x), & 2x \in Q; \\ \frac{1}{4}\|P(2x) - 2x\|^2 - \frac{1}{2}\|x\|^2, & \text{otherwise.} \end{cases} \quad (3)$$

$E(x)$ is continuous and differentiable. The evolution of $E(x)$ with respect to time t is

$$\frac{dE(x(t))}{dt} \leq -\|P(2x(t)) - x(t)\|^2 \leq 0.$$

So, $E(x)$ is a Lyapunov function of the system (2) on Q and this system is globally stable in D^m.

The projection function $P(p_0)$, $p_0 \in R^m$ an arbitrary initial state p_0 in R^m, is a solution of the quadratic programming problem

$$\min \frac{1}{2}\left\| p_0 - \sum_{i=1}^{m} \lambda_i p_i \right\|^2, \ \sum_{i=1}^{n} \lambda_i = 1, 1 \geq \lambda_i \geq 0. \quad (4)$$

This problem (4) can be solved by a globally evolving network as

$$\begin{cases} \frac{dr}{dt} = -(I + A_0 A_0^T)[r - P_0(r - D_0^T s - A_0 A_0^T r + A_0 p_0)] \\ \qquad\quad -D_0^T(D_0 r - 1) \\ \frac{ds}{dt} = -D_0 P_0(r - D_0^T s - A_0 A_0^T r + A_0 p_0) + 1 \end{cases} \quad (5)$$

where P_0 is the projection function on an unit hypercube Q_0, D_0 is an $n \times 1$ matrix with all the elements being 1, and $A_0 = (p_1, p_2, \ldots, p_n)^T$. $r(t)$ is a column vector contains $\lambda_i(t)$ as its i^{th} element. Solution $\lambda_i = \lambda_i(\infty)$, $P(p_0) = r(\infty)$, can be obtained when $\frac{dr}{dt} \approx 0$ after many evolutions. $P(p_0) = r(\infty)$. The system (2) is shown in Fig. 1.

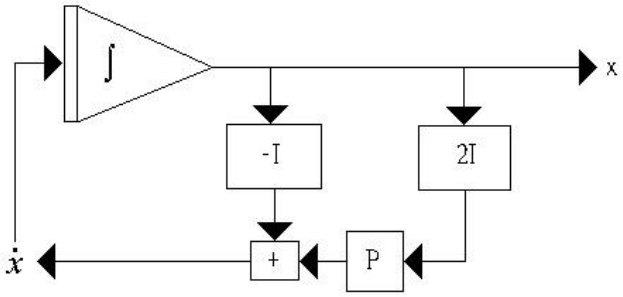

Fig. 1. The block diagram for the dynamical system (2).

An improved AM based on this quadratic programming problem (1) was proposed[9] to serve as a BSB model. The system equation of the state is improved[10] as

$$x_{t+1} = P(2x_t). \quad (6)$$

Accordingly[9], the iteration(5) is rewritten as

$$\begin{cases} \frac{dr}{dt} = P_0(r - D_0^T s - A_0 A_0^T r + A_0 p_0) - r, \\ \frac{ds}{dt} = -D_0 P_0(r - D_0^T s - A_0 A_0^T r + A_0 p_0) + 1. \end{cases} \quad (7)$$

We plot two simulations to depict the evolution of Tao's AM. To simplify notations, each pattern is encoded into a simple form by successive transformations. As an example, the two patterns with three binary bits in each pattern, $\{p_1 = (1, -1, -1),\ p_2 = (-1, 1, -1)\}$, can be encoded as

$$\begin{aligned} & \{(1, -1, -1),\ (-1, 1, -1)\}, \\ \Leftrightarrow\ & \{(1, 0, 0),\ (0, 1, 0)\}, \\ \Leftrightarrow\ & \{001,\ 010\}, \\ \Leftrightarrow\ & (1, 2)_3. \end{aligned} \quad (8)$$

The following simulations are for the cases $(0, 3)_2$ in 2D and $(0, 3, 5)_3$ in 3D. According to (2), we record the states during the system evolution. The closed convex set of $(0, 3)_2$ is a line as shown in Fig. 2, and several evolution traces with different initial states, denoted by small circles '$\circ$', are plotted in Fig. 3.

,

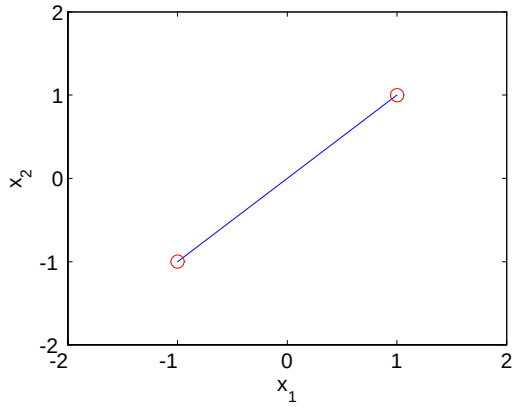

Fig. 2. The closed convex set of the case $(0, 3)_2$.

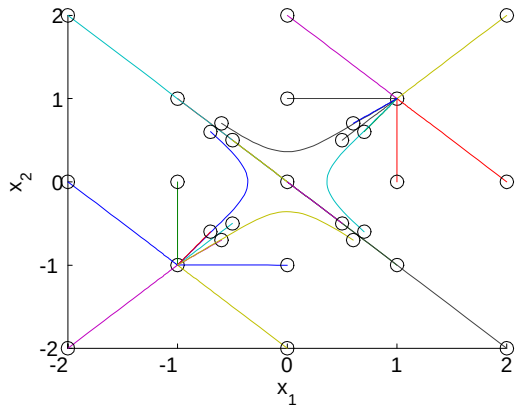

Fig. 3. The system traces with different initial states for the case $(0, 3)_2$.

From these evolution traces, we see that there are 3 equilibrium states. These states are the two patterns and an extra state (or an unrecognizable state). This unrecognizable state is in the mid-point of the line connected the two patterns. When an initial state has equal distance from both patterns, the system will evolve to this unrecognizable state.

The convex set of $(0,3,5)_3$ is an equidistant triangle as in Fig. 4, and several evolution curves are recorder in Figs. 5 and 6.

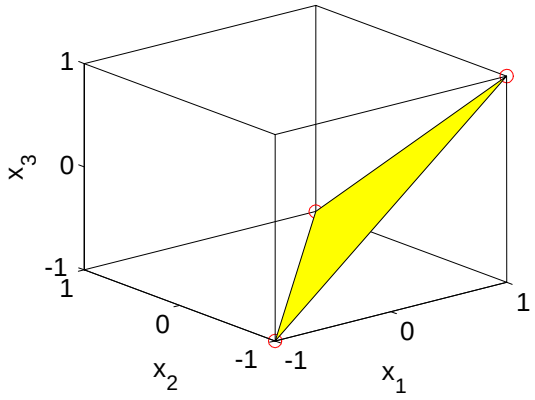

Fig. 4. The convex set of the case $(0,3,5)_3$.

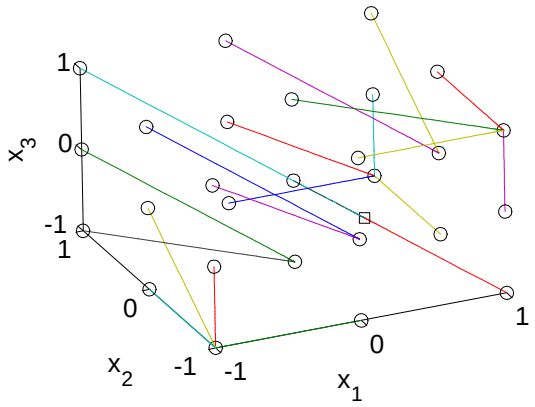

Fig. 5. Evolution curves with different initial states of the case $(0,3,5)_3$.

The number of equilibrium states is 7 in this case $(0,3,5)_3$ whose convex set is an equidistant triangle. The square points in Figs. 5 and 6 are the equilibrium states which are unstable and are those unrecognizable states. These states can be monitored by inspecting broken binaries. Whenever the system evolves to such state, one can further evolve the system by perturbing this state to reach all those binary states which underlie this state.

Three typical system evolution curves (3) for the case $(0,3,5)_3$ are recorded in Fig. 7. These curves evolve from three different initial states and reach three different states which are a pattern, denoted by 1st, a mid-point between any two patterns, denoted by 2nd, center of the three patterns, denoted by 3rd.

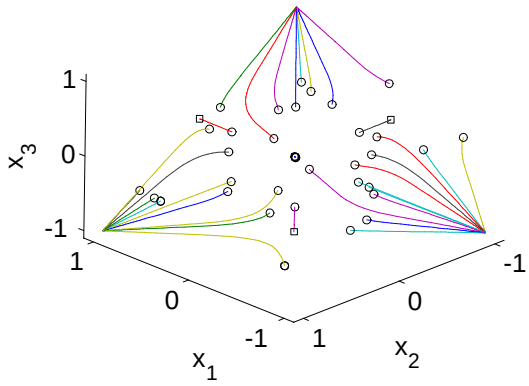

Fig. 6. Several other evolution curves of the case $(0,3,5)_3$.

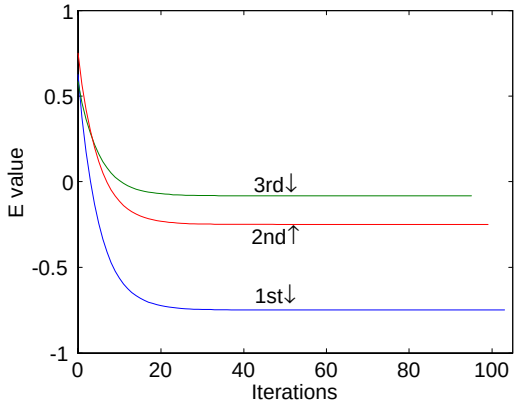

Fig. 7. The energy value E is decreased during all three kinds of evolution.

B. The etAM

The weight matrix of the Hopfield model is

$$w_{ji} = \begin{cases} \frac{1}{n}\sum_{k=1}^{n} p_{k,j}p_{k,i}, & j \neq i, \\ 0, & j = i \end{cases}.$$

The model will iterate with an input noisy pattern $v(0)$ according to the equation below

$$v_j(t+1) = sgn(\sum_{i=1}^{m} w_{ji}v_i(t)),\ j = 1,\ldots,m. \qquad (9)$$

The iteration will stop when all $v_j(t)$ are evolved to certain fixed values. Here the sgn function gives a value -1 when the value in the bracket is smaller than 0, gives 1 when the value in the bracket is smaller than 0, and $v_i(t)$ when the value in the bracket is exactly 0.

In the etAM the i^{th} neuron's weights, $W_i = (w_{i1}, w_{i2}, w_{i3}, \ldots, w_{im})^T$, and threshold, θ_i, configure the i^{th} hyperplane. This hyperplane is given by the solution space

of v of the equation

$$\begin{aligned} & w_{i1}v_1 + w_{i2}v_2 + w_{i3}v_3 + \ldots + w_{im}v_m - \theta_i \\ = \quad & W_i v - \theta_i = 0, \ i = 1, \ldots, m. \end{aligned}$$

The weight vector $W_i = (w_{i1}, w_{i2}, w_{i3}, \ldots, w_{im})^T$ is a normalized column vector with length 1. W_i is the unit direction of the i^{th} hyperplane. This hyperplane separate the hypercube into a positive section, where $W_i v > \theta_i$, and a negative section, where $W_i v < \theta_i$. Each pattern is located in one section according to the sign of the i^{th} bit of this pattern. That is a pattern with a positive i^{th} bit is in the positive section and a pattern with a negative i^{th}bit is in the negative section. The etAM trains the weight vector and threhold to maximize the minimum distance among all patterns in both sections and the hyperplane, that is we maximize the distance between the two closest patterns in two sections to the hyperplane. The algorithm for tuning the hyperplane is in below.

1) Initialize the weights according to the Hopfield model:

$$\begin{aligned} w_{ji}(0) = \sum_{k=1}^{n} p_{k,j}p_{k,i}, \quad i, j = 1, \ldots, m; \\ \theta_i(0) = 0, \qquad\qquad i = 1, \ldots, m. \end{aligned} \tag{10}$$

   Then normalize W_i, $i = 1, \ldots, m$, and initialize q to 1. $w_{ij}(t)$ is the weight between the i^{th} neuron and the j^{th} neuron at training time step t.

2) For the q^{th} neuron, calculate the distances from all patterns to the hyperplane and find the two patterns, p_a and p_b, from both sections which have minimal distances that is,

$$\begin{cases} d_q^k = \sum_{j=1}^{m} w_{qj}p_{k,j} - \theta_q, \quad k = 1, \ldots, n; \\ d_q^a = \min\{d_q^k \mid p_{k,q} = 1, k = 1, \ldots, n\}, \\ d_q^b = \max\{d_q^k \mid p_{k,q} = -1, k = 1, \ldots, n\}. \end{cases} \tag{11}$$

3) If all the patterns have $p_{k,q} = 1$, then set θ_q to a value less than $-\sqrt{m}$, increase q by 1, and goto 2. If all the patterns have $p_{k,q} = -1$, then set θ_q to a value greater than $\sqrt{m}$, increase q by 1, and goto 2. In this step, we move the hyperplane outside of the hypercube. Note that the distance between each corner and the origin is $\sqrt{m}$.

4) Shift the hyperplane to the middle of pattern a and b to maximize the minimal distance as follows

$$\theta_q(t+1) = \theta_q(t) + \frac{(d_q^a + d_q^b)}{2}.$$

   After adjusting, the minimal distance d_q^* is $(d_q^a - d_q^b) / 2$.

5) Rotate the hyperplane to ffurther increase the distances from patterns p_a and p_b to the hyperplane by

$$w_{qj}(t+1) = w_{qj}(t) + \alpha(p_{a,q}p_{a,j} + p_{b,q}p_{b,j}), \ j = 1, \ldots, m, \tag{12}$$

   where α is the learning rate, and then normalize W_q.

6) Repeat (11) to compute the new d_q^a and d_q^b. If the new $(d_q^a - d_q^b) / 2$ is larger than the previous d_q^*, go to 2. If not, undo (12), increase q by 1, and if q is no greater than m, go to 2, else stop.

After obtaining the weights and thresholds, they are then used to restore noisy patterns.

C. mECR

In the ECR, the weights and thresholds are adjusted according to the equations in below

$$w_{ij}(t+1) = w_{ij}(t) + \eta(p_{k,i} - v_i(t))p_{k,j}, \tag{13}$$

$$\theta_i(t+1) = \theta_i(t) - \eta(p_{k,i} - v_i(t)), \tag{14}$$

$$v_i(t) = sgn(\sum_{j=1}^{m} w_{ij}(t)p_{k,j} - \theta_i(t)), \tag{15}$$

where η is the learning rate, $v_i(t)$ is the value of the i^{th} neuron's output at time t, $p_{k,j}$ is the j^{th} bit of the k^{th} pattern used to train the weights and threshold. Since the error tolerance of the ECR is weak, a modified ECR (mECR) is introduced by replacing equation (15) with (16)

$$v_i(t) = sgn(\sum_{j=1}^{m} w_{ij}(t)p_{k,j} - \theta_i(t) - \gamma p_{k,i}). \tag{16}$$

Here γ is a small positive number. The mECR utilizes the extra term $-\gamma p_{k,i}$ to further improve the error tolerance. This term moves the hyperplane further away from the pattern to ensure the error tolerance.

II. Summary and Comparison

In the following experiments, the projection function and the system (2) are solved using the fourth order Runge-Kutta method. For the projection function, the time step in the Runge-Kutta method is set to 0.1 and the total simulation time is set to 2000. For the system (2), the time step in the Runge-Kutta method is set to 0.01.

The solved values will be rounded to 1 or -1 whenever they are close to these binary values. In etAM, the learning rate α is set to 0.005. In mECR, the learning rate η is set to 0.2, and γ is set to 1. Since the trained weights and thresholds are not the same for each execution of the mECR, the results obtained by the mECR are averaged by 5 independent executions.

The three tables in the next page contain the results obtained by Tao's AM, etAM, and mECR. The explanation for each column item of the tables is in below.

- SS denotes the total number of binary stable states.
- TS denotes the total number of unstable binary states evolving to the binary stable states.
- TP denotes the total number of binary states evolving to the patterns, TP= n.
- US denotes the total number of unrecognizable states which can be reached from all binary states in Tao's AM. There may exist extra unrecognizable states which

can be reached from non-binary states. Note that an unrecognizable state do not have all binary elements.
- TU denotes the total number of binary states evolving to the unrecognizable states in Tao's AM.
- C denotes the total number of limit cycles.
- TC denotes the total number of binary states evolving to limit cycles.
- RP denotes the total number of noisy patterns with one bit error that can be restored(/n^*). n^* is the number of one bit error patterns, n^*, where we will exclude those patterns with one or two bit Hamming distances.
- Time denotes the total CPU time in seconds by the Tao's AM to obtain the results listed in a whole row. In the last column, 'Time*' denotes the time by the improved Tao's AM[10].

The bottom rows of the above three tables use the 10-bits pattern example in [4]. From the Table, all three AMs can store patterns correctly. There is no limit cycle found in Tao's AM and etAM. Tao's AM generates no spurious states and has good recovery capability. The etAM and mECR have no unrecognizable state or equilibrium states. They generate spurious stable states. These spurious states have no basin and cannot withstand thermal noise perturbation. A large number in TP roughly shows the basin size of a stored pattern. A pattern with large basin can be recovered from noisy patterns and can withstand thermal noise. All three models generate pattern basins. This fact is shown by the large numbers under iterms TS and RP. In several cases the mECR generates limit cycles. The mECR is weak in error tolerance in most cases.

When an equilibrium state reached by using Tao's AM is not a stored pattern, this state has broken binaries. These broken binaries can indicate such equilibrium state and can be monitored during the system evolution. Whenever the system evolves to such state, one can further evolve the system by perturbing this state to reach all those binary states which underlie this state.

Since Tao's AM uses the Runge-Kutta method, the computation time is roughly $O(mn^2 + \beta n^2)$, here mn^2 is the cost to calculate $A_0 A_0^T$ in equation (5) and β is a huge number, 20000×4 in our experiments. The computation time for the improved Tao's AM[10] is reduced to $O(\beta' n^2)$, β' is much smaller than β. The computation time for the etAM and mECR is $O(m^2)$ in both training and restoring.

The memory spaces needed for the etAM and mECR are $O(m^2 + mn)$ which contain the weight matrix, threshold vector, inputs, and patterns. The spaces required for Tao's AM are $O(mn + n^2)$ which contain the patterns, the space required for the Runge-Kutta method, and inputs.

As for parallel distributed processing, all three AMs can be programmed parallelly. According to the block diagram in Fig. 1, except the projection function, all other components in Tao's AM are obviously parallel. Since the projection function is solved by using equation (5), all the operations on matrices in this equation can be calculated parallelly.

As for the temporal associative memory, the etAM can be modified[5] to accomplish temporal tasks. The mECR can also be modified for the temporal associative memory as

$$w_{ij}(t+1) = w_{ij}(t) + \eta(p_{k+1,i} - v_i(t))p_{k,j}, \quad (17)$$

$$\theta_i(t+1) = \theta_i(t) - \eta(p_{k+1,i} - v_i(t)), \quad (18)$$

$$v_i(t) = sgn(\sum_{j=1}^{m} w_{ij}(t)p_{k,j} - \theta_i(t) \quad (19)$$

$$-\gamma p_{k+1,i}). \quad (20)$$

Here, $p_{k+1,i}$ is the i^{th} bit of the next pattern after p_k. The overall specification is shown in Table IV.

REFERENCES

[1] J. Bruck, "On the convergence properties of the Hopfield model," *Proc. IEEE*, vol. 78, pp.1579-1585, Oct. 1990.

[2] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Academy Sci.*, vol. 79, pp. 2554-2558, Apr. 1982.

[3] J. Li, A. N. Michael, and W. Porod, "Qualitative analysis and synthesis of a class of neural networks," *IEEE Trans. Circuits System*, vol. 35, pp. 976-986, Aug. 1988.

[4] W. E. Lillo, D. C. Miller, and S. H. Zak, "Synthesis of brain-state-in-a-box(BSB) based on associative memory," *IEEE Trans. Neural Networks*, vol. 5, no. 5, pp. 730-737, Sep. 1994.

[5] C. Y. Liou, and S. K. Yuan, "Error tolerant associative memory," *Biological Cybernetics*, vol. 81, pp.331-342, 1999.

[6] L. Personnaz, I. Guyon, and G. Drefus, "Information storage and retrieval in spin-glass like neural networks," *J. Phys. Lett.*, vol. 46, pp. 359-366, Apr. 1985.

[7] Q. Tao and T. Fang, "The neural network based on constraint domain and its applications" (in Chinese), *Chinese Artificial Intell. Pattern Recognition*, vol. 11, no. 4, pp. 472-478, 1998.

[8] Q. Tao, T. Fang, and H. Qiao,"A novel continuous-time neural network for realizing associative memory," *IEEE Trans. Neural Networks*, vol. 12, no.2, pp. 418-423, Mar. 2001.

[9] Q. Tao, J. Cao, D. Sun, "A simple and high performance neural network for quadratic programming problems", *Applied Mathematics and Computation*, vol.124, pp. 251–260, 2001.

[10] Q. Tao, J. Cao, and X. Liu, "The BSB neural network in the convex body spanned by the prototype patterns for associative memory," *Applied Mathematics and Computation*, vol. 132, pp. 175-187, 2002.

[11] Y. Xia, "A new neural network for solving linear and quadratic programming problems," *IEEE Trans. Neural Networks*, vol. 7, no. 6, pp. 1544-1547, Nov. 1996.

TABLE I

SIMULATIONS OF TAO'S AM.

Tao's AM	SS	US	TS	TP	TU	RP	Time(s)	Time*(s)
$(0,1,2)_5$	3	1	21	24	8	9/9	1845	8.51
$(0,1,6)_5$	3	1	21	24	8	9/9	1844	9.70
$(0,1,14)_5$	3	1	23	26	6	13/13	1839	9.91
$(0,1,30)_5$	3	1	23	26	6	13/13	1824	11.82
$(0,3,5)_5$	3	4	9	12	20	6/6	1763	7.55
$(0,3,12)_5$	3	3	11	14	18	7/7	1773	8.05
$(0,3,13)_5$	3	1	21	24	8	11/11	1806	10.05
$(0,3,28)_5$	3	1	21	24	8	11/11	1833	12.21
$(0,3,29)_5$	3	4	12	15	17	11/11	1744	7.86
$(0,7,25)_5$	3	1	23	26	6	15/15	1829	10.12
$(62,78,235,291,473,834)_{10}$	6	25	628	634	390	60/60	150900	1292

TABLE II

CONVERGENCE OF ETAM.

etAM	SS	US	TS	TP	TU	RP	Time(s)
$(0,1,2)_5$	4	0	28	24	0	9/9	0.007
$(0,1,6)_5$	4	0	28	26	0	10/9	0.007
$(0,1,14)_5$	4	0	28	24	0	12/13	0.007
$(0,1,30)_5$	4	0	28	27	0	12/13	0.007
$(0,3,5)_5$	8	0	24	9	0	6/6	0.007
$(0,3,12)_5$	4	0	28	30	0	11/7	0.007
$(0,3,13)_5$	7	0	25	16	0	9/11	0.007
$(0,3,28)_5$	4	0	28	28	0	13/11	0.007
$(0,3,29)_5$	7	0	25	13	0	9/11	0.007
$(0,7,25)_5$	7	0	25	23	0	12/15	0.007
$(62,78,235,291,473,834)_{10}$	157	0	867	246	0	34/60	0.009

TABLE III

SIMULATIONS OF MECR.

mECR	SS	C	TS	TP	TC	RP	Time(s)
$(0,1,2)_5$	9	0.2	22.4	18.6	0.6	9/9	0.007
$(0,1,6)_5$	10	0	22	26	0	7.8/9	0.007
$(0,1,14)_5$	7	0	25	22.2	0	10/13	0.007
$(0,1,30)_5$	5.4	0	26.6	25.8	0	11.2/13	0.007
$(0,3,5)_5$	12	0	20	14.2	0	6/6	0.007
$(0,3,12)_5$	9.4	0	22.6	18.2	0	6.4/7	0.007
$(0,3,13)_5$	11	0.2	20.4	16.6	0.6	6.6/11	0.007
$(0,3,28)_5$	5.8	0	26.2	23	0	10/11	0.007
$(0,3,29)_5$	8	0.2	23.4	18.2	0.6	7.6/11	0.007
$(0,7,25)_5$	8.6	0	23.4	19.8	0	7.8/15	0.007
$(62,78,235,291,473,834)_{10}$	161.8	0	862.2	106.6	0	16/60	0.009

TABLE IV

OVERALL COMPARISON

	Tao's AM	Improved Tao's AM	etAM	mECR
Temporal associative memory	N/A	N/A	A	A
Parallel distributed processing	A	A	A	A
Computation complexity	$O(mn^2 + \beta n^2)$	$O(\beta^0 n^2)$	$O(m^2)$	$O(m^2)$
Memory complexity	$O(mn + n^2)$	$O(mn + n^2)$	$O(mn + m^2)$	$O(mn + m^2)$
Spurious stable states	0	0	(SS - n) in Table II	(SS - n) in Table III
Unrecognizable states	US in Table I	US in Table I	0	0
Neuron	Unclear	Unclear	A	A
Synapse	Unclear	Unclear	A	A