Chapter 13

Tracking Mean Field Dynamics by Synchronous Computations of Recurrent Multilayer Perceptrons

Jiann-Ming Wu, Jung-Chao Ban and Chun-Chang Wu

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/57217

1. Introduction

Mean field dynamics have been extensively applied for organizing neural networks in the field of computational neuroscience, since Hopfield pioneered collective decisions of interconnected processing elements for combinatorial optimization [1–6] and memory association [7, 8]. Both nonlinear transfer functions and synapses in a Hopfield neural network are a subsequence of mean field dynamics that characterize the mean configuration of a large scaled physical system at thermal equilibrium in the field of statistical mechanism. In the past decades, the mean field dynamics has been extensively applied for deriving interactive neural dynamics of solving complex tasks, such as combinatorial optimization [4, 6, 9], self-organization [10], clustering analysis [11][12], independent component analysis [13], and regression [14][15].

Mean field equations characterize feasible configurations for problem solving. Let $s_i \in \{-1, 1\}$ denote a binary random variable for modeling a stochastic two-alternative processing element and $\mathbf{s} = \{s_i\}_i$ represent a configuration for problem solving. The feasibility of $\mathbf{s}$ to the attacked problem is inversely quantified by an energy function $E(\mathbf{s})$. Minimizing $E(\mathbf{s})$ with respect to $\mathbf{s}$ means to seek the optimal solution. Under the Boltzmann assumption, the joint probability of all s_i is proportional to $exp(-\beta E(\mathbf{s}))$, where β denotes the inverse of a temperature-like parameter. As in previous works [4][5][6], the Kullback-Leibler divergence between the product of marginal probabilities and the joint probability of all s_i induces a tractable free energy function ψ that depends on the expectation of s_i, denoted by $\langle s_i \rangle$, for all i.

The following mean field dynamics exactly characterize the saddle point of a typical tractable free energy function,

$$u_i = -\frac{\partial E(\langle \mathbf{s} \rangle)}{\partial \langle s_i \rangle} \tag{1}$$

$$\langle s_i \rangle = f(u_i) \equiv \tanh(\beta u_i) \tag{2}$$

where u_i denotes an external field, $f \equiv \tanh$ is a sigmoid-like transfer function and $\langle s_i \rangle$ denotes the mean activation. In previous works [4][6], E is quadratic and u_i measures a weight sum of activations other than $\langle s_i \rangle$, such as

$$u_i = \sum_{j \neq i} w_{ij} \left\langle s_j \right\rangle + c_i \tag{3}$$

where w_{ij} denotes the synapse that connects neural processing elements i and j. For fixed β, equation (2) defines the transfer function of interconnected processing elements and equation (3) sketches synapses. The realized information processes are distributed and with computational features of fault tolerance and collective decision. All interconnected processing elements in a Hopfield neural network asynchronously operate to seek a stable configuration under an annealing process [6] that carefully scheduling β from sufficiently small to large values.

At each intermediate β, a stable configuration means a result of minimizing the mean energy function against maximizing the entropy for emulating thermal equilibrium of statistical mechanism. At the end of the annealing process, by equation (2), $\langle s_i \rangle \in \{-1, 1\}$ and the mean configuration $\langle \mathbf{s} \rangle$ is a vector of N binary values, well representing a feasible solution for problem solving. Empirical results in previous works [4][6] have extensively shown that the physical-like annealing process guarantees effectiveness and reliability of seeking the global or near global minimum of $E(\mathbf{s})$ for problem solving. In previous works [15–18], mean field dynamics have been extended for multi-state Potts modeling and applied for unsupervised learning and supervised learning of neural networks toward solving self organization, independent component analysis, function approximation and discriminate analysis.

However from the perspective of numerical simulations, asynchronous operation of interconnected processing elements means one-by-one sequential updating of neural variables. It is more efficient to simulate synchronous and parallel updating of neural variables by vector codes. Multilayer perceptrons or Adalines have been organized for parallel and synchronous processes. Significant computational features include synchronous data transmission and parallel signal processes through multilayer perceptrons. A network of multilayer perceptrons is typically composed of input, hidden, output layers as well as inter-connections among consecutive layers. The input $\mathbf{x} \in R^d$ transmits through interconnections to form external fields,

$$h = A\mathbf{x} + \mathbf{c}, \tag{4}$$

and the nonlinear transfer function translates h to activations of hidden units,

$$\mathbf{v} = F(h) = [f(h_1), ..., f(h_M)]^T, \tag{5}$$

which is multiplied by a matrix of posterior weights, denoted by R, to form the network output

$$\mathbf{y} = R\mathbf{v} \tag{6}$$

http://dx.doi.org/10.5772/57217

Equations (4)-(6) describe synchronous data transmissions and parallel signal processes, by which it only takes three time clocks to translate $\mathbf{x}$ to $\mathbf{y}$.

A recurrent network of multilayer perceptrons is further equipped with circular connections from the output to input layers. By feedback circular connections, the current output becomes the network input at the upcoming time step. Let R be an identity matrix. Setting $\mathbf{x}$ to $\mathbf{y}_n$ and $\mathbf{y}$ to $\mathbf{y}_{n+1}$ leads to the following recursive function realized by recurrent multilayer perceptrons

$$\mathbf{y}_{n+1} = F(A\mathbf{y}_n) \tag{7}$$

Since perceptrons and adalines perform post-nonlinear projection, the organized multilayer neural network realizes a high dimensional nonlinear mapping from the input domain to the output range, which has been shown significant for solving complex tasks against traditional linear systems. Recurrent multilayer perceptrons perform parallel and synchronous computations for realizing the behavior of MIMO (multiple input multiple output) recurrent relation or characterizing nonlinear autoregression of time series. Recurrent multilayer perceptrons have been applied for nonlinear autoregressive modeling of chaotic time series prediction [19] and financial time series [20].

This work applies recurrent multilayer perceptrons for tracking mean field dynamics by synchronous and parallel computations. A systematic approach is proposed for translating mean field dynamics (1) and (2) to the nonlinear recursive function (7) such that recurrent multilayer perceptrons can track the saddle point of ψ by parallel and synchronous computations. The strategy is to introduce time delays and auxiliary variables for expanding local memories of storing individual states, and translate loosely coupled or densely coupled first order mean field equations to a system of post-nonlinear recursive functions, which can be evaluated directly by iterative synchronous computations of recurrent multilayer perceptrons.

Section 2 applies parallel and synchronous computations of recurrent multilayer perceptrons for tracking mean field dynamics. Asynchronous updating of tracking linear dynamics and mean field dynamics is translated to equivalent synchronous updating. Section 3 applies the transformation to derive synchronous updating of tracking mean field dynamics for solving graph bisection problem and verifies the proposed approach by numerical simulations. Section 4 further presents a hybrid of asynchronous and synchronous processes for tracking sparse large scaled mean field dynamics of sparse connectivity for problem solving.

2. Synchronous computation of tracking mean field dynamics

By asynchronous updating at each time step numerical simulations select one processing element and refine its mean activation under fixed mean activations of the others. Let $\psi_i(\langle s_i \rangle)$ denote ψ with fixed $\left\langle s_j \right\rangle$, $j \neq i$,

$$\psi_i(\langle s_i \rangle) = h_i \langle s_i \rangle + c_i - \left[\frac{1+\langle s_i \rangle}{2} \log \frac{1+\langle s_i \rangle}{2} + \frac{1-\langle s_i \rangle}{2} \log \frac{1-\langle s_i \rangle}{2} \right], \tag{8}$$

where $\beta = 1$ is considered. $\langle s_i \rangle = \tanh(h_i)$ minimizes the above equation. $\psi_i(\langle s_i \rangle)$ is an approximation to the one-dimensional function obtained by cutting functional surface of ψ along the direction of $\langle s_i \rangle$ for fixed $\left\langle s_j \right\rangle$, $j \neq i$. By asynchronous updating the coefficient h_i of the linear term in equation (8) always maintains an instance determined by fixing most recently updated mean activations. Asynchronous updating is represented by

$$\langle s_i \rangle \leftarrow f\left(\sum_{j \neq i} w_{ij} \left\langle s_j \right\rangle + c_i\right). \tag{9}$$

The asynchronous cutting and approximating strategy is very different from synchronous updating that directly combines equations (2) and (3) for all i, such as

$$\langle s \rangle \leftarrow \tanh(W \langle s \rangle) + c \tag{10}$$

where W collects all w_{ij}. By synchronous updating, all h_i use the copy formed by all mean activations synchronously determined at the previous step. Numerical simulations have verified synchronous updating based on equation (10) infeasible for relaxing of mean field dynamics.

2.1. Linear system

Let f be a linear function and the asynchronous updating rule (9) is equivalent to

$$x_i \leftarrow \sum_{j \neq i} a_{ij} x_j + c_i \tag{11}$$

where $A = [a_{ij}]$ is a $N \times N$ matrix with $a_{ii} = 0$, $\forall i = 1, \cdots, N$. To facilitate our presentation, we first give an example with $N = 4$ for illustration. Figure 1 shows data flow of asynchronous updating (11), where directed edges indicate the latest mean activations employed for updating. Each time asynchronous updating insists on revising only one mean activation. Without losing generality, consecutive steps of updating mean activations can be listed as follows,

$$\begin{array}{llll} x_1[k+1] = 0 & + a_{12}x_2[k] & + a_{13}x_3[k] & + a_{14}x_4[k] + c_1 \\ x_2[k+2] = a_{21}x_1[k+1] + 0 & & + a_{23}x_3[k] & + a_{24}x_4[k] + c_2 \\ x_3[k+3] = a_{31}x_1[k+1] + a_{32}x_2[k+2] + 0 & & & + a_{34}x_4[k] + c_3 \\ x_4[k+4] = a_{41}x_1[k+1] + a_{42}x_2[k+2] + a_{43}x_3[k+3] + 0 & & & + c_4 \end{array} \tag{12}$$

The system (12) is translated to synchronous updating by replacing k with $k-1$, $k-2$, $k-3$ and $k-4$ respectively in the four rows of equation (13)

$$\begin{array}{llll} x_1[k] = 0 & + a_{12}x_2[k-1] & + a_{13}x_3[k-1] & + a_{14}x_4[k-1] + c_1 \\ x_2[k] = a_{21}x_1[k-1] + 0 & & + a_{23}x_3[k-2] & + a_{24}x_4[k-2] + c_2 \\ x_3[k] = a_{31}x_1[k-2] + a_{32}x_2[k-1] + 0 & & & + a_{34}x_4[k-3] + c_3 \\ x_4[k] = a_{41}x_1[k-3] + a_{42}x_2[k-2] + a_{43}x_3[k-1] + 0 & & & + c_4 \end{array} \tag{13}$$

http://dx.doi.org/10.5772/57217

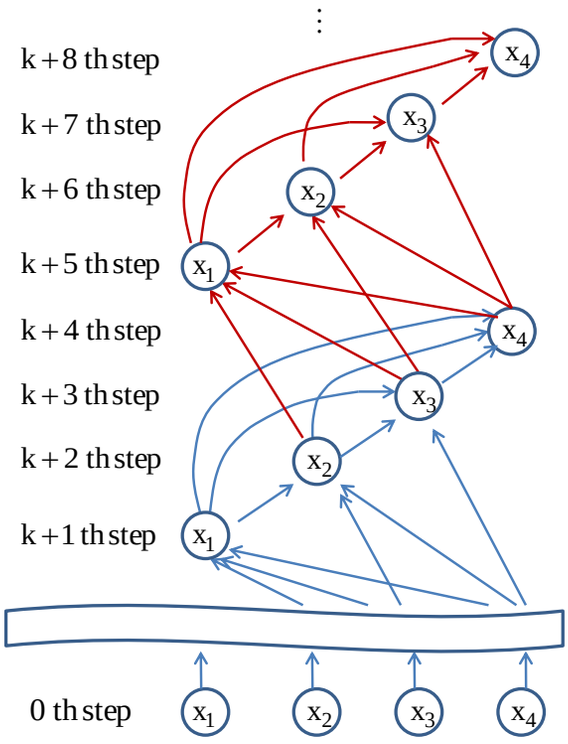


Figure 1. Asynchronous update.

for $k \geq 3$. The matrix form is expressed by

$$\mathbf{x}[k] = B\mathbf{u}[k] + \mathbf{c} \tag{14}$$

where $\mathbf{x}[k] = (x_1[k], \cdots, x_4[k])^T$ and

$$\mathbf{u}[k] = \begin{pmatrix} \mathbf{x}[k-1] \\ \mathbf{x}[k-2] \\ \mathbf{x}[k-3] \end{pmatrix},$$

T denotes transpose and

$$B = \left[\begin{array}{cccc|cccc|cccc} 0 & a_{12} & a_{13} & a_{14} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{21} & 0 & 0 & 0 & 0 & 0 & a_{23} & a_{24} & 0 & 0 & 0 & 0 \\ 0 & a_{32} & 0 & 0 & a_{31} & 0 & 0 & 0 & 0 & 0 & 0 & a_{34} \\ 0 & 0 & a_{42} & 0 & 0 & a_{42} & 0 & 0 & a_{41} & 0 & 0 & 0 \end{array}\right]$$

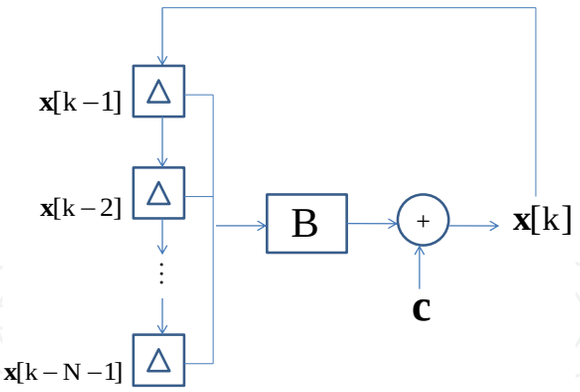


Figure 2. A linear recurrent system for synchronous computations. The triangle denotes time delay.

For initialization, $\mathbf{x}[0]$ is copied three times to form $\mathbf{u}[N]$ where $N = 4$. Figure 2 shows a recurrent linear network for synchronous computations of equation (14). The circular connection transmits the current output to the input layer at the upcoming step. In general, $\mathbf{u}[k]$ is given by

$$\mathbf{u}[k] = \begin{pmatrix} \mathbf{x}[k-1] \\ \mathbf{x}[k-2] \\ \vdots \\ \mathbf{x}[k-N+1] \end{pmatrix}$$

which concatenates $N-1$ consecutive steps of mean activations and $B = [B_1 B_2 \cdots B_{N-1}]$ is composed of $N-1$ submatrices. Figure 3 and 4 show the structure of matrices $\{B_n\}_{n=1}^{N-1}$. Distinct colors represent nonzero entries. Figure 5 shows the flow chart of creating matrix B. Figure 6 shows the flow chart of simulating asynchronous updating by linear recurrent computations where repmat is a matlab bulit-in function for matrix replication.

2.2. Mean field dynamics

Asynchronous updating (11) can be regarded as a special case of asynchronous updating (9) of mean field dynamics. Let f denote a post-nonlinear function and $v_i = \langle s_i \rangle$ for general situations. Synchronous parallel updating is explored for emulating asynchronous updating (9) for tracking mean field dynamics.

Asynchronous updating rule is rewritten as follows,

$$v_i \leftarrow f\left(\beta\left[\sum_{j\neq i} w_{ij}v_j + c_i\right]\right) \equiv g(v_1, \cdots, v_N, c_i)$$

http://dx.doi.org/10.5772/57217

$$B_1 = \begin{bmatrix} 0 & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} & a_{18} \\ a_{21} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{32} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{43} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{54} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{65} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & a_{76} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & a_{87} & 0 \end{bmatrix} \quad B_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{34} & a_{35} & a_{36} & a_{37} & a_{38} \\ a_{41} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{52} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{63} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{74} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{85} & 0 & 0 & 0 \end{bmatrix}$$

$$\vdots$$

$$B_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{23} & a_{24} & a_{25} & a_{26} & a_{27} & a_{28} \\ a_{31} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{42} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{53} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{64} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{75} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & a_{86} & 0 & 0 \end{bmatrix} \quad B_7 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{78} \\ a_{81} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 3. The representation of matrix $\{B_k\}_{k=1}^{7}$ for $N = 8$.

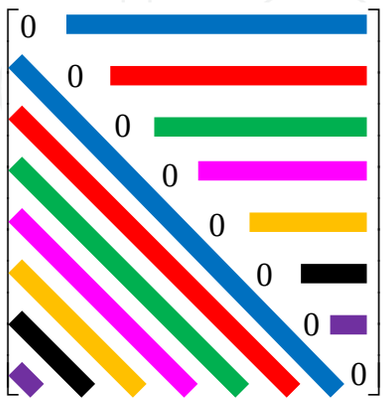


Figure 4. A diagram for illustrating the structure of $\{B_k\}_{k=1}^{7}$.

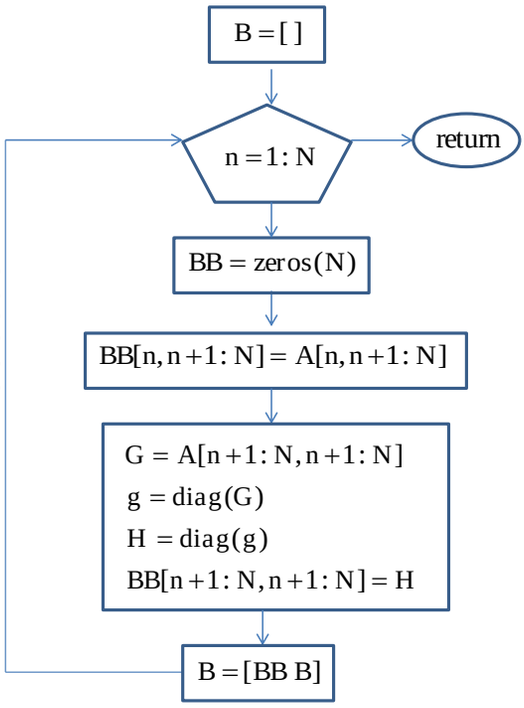


Figure 5. The flow chart of forming B.

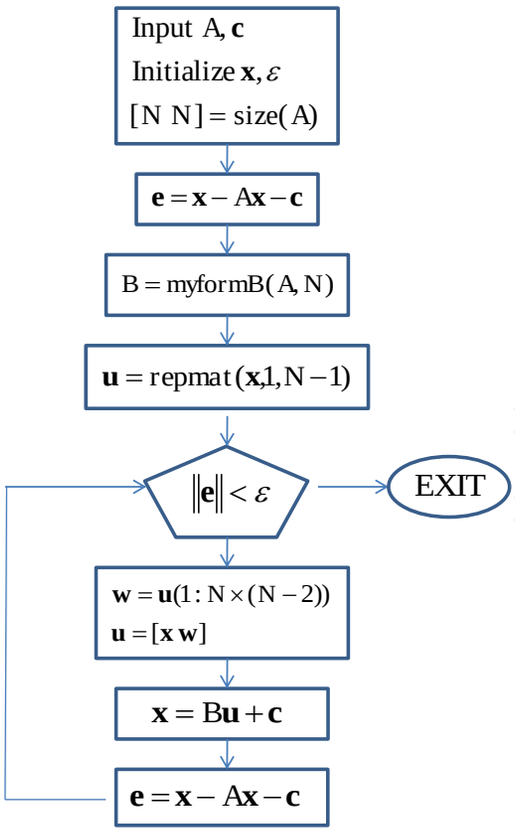


Figure 6. The flow chart of solving linear system by synchronous parallel computations.

http://dx.doi.org/10.5772/57217

where $v_i = \langle s_i \rangle$. Let $\mathbf{v}[0] = (v_1[0], v_2[0], \cdots, v_N[0])$ denote the initial mean configuration. The leave-one-out asynchronous updating is expressed as

$$\begin{aligned}
v_1[k+1] &= g(v_2[k], v_3[k], v_4[k], \cdots, v_N[k], c_1) \\
v_2[k+2] &= g(v_1[k+1], v_3[k], v_4[k], \cdots, v_N[k], c_2) \\
v_3[k+3] &= g(v_1[k+1], v[k+2], v_4[k], \cdots, v_N[k], c_3) \\
&\vdots \\
v_n[k+n] &= g(v_1[k+1], v_2[k+2], \cdots, v_{n-1}[k+n-1], v_{n+1}[k], \cdots, v_N[k], c_n) \\
&\vdots \\
v_N[k+N] &= g(v_1[k+1], v_2[k+2], v_3[k+3], \cdots, v_{N-1}[k+N-1], c_N)
\end{aligned} \tag{15}$$

where $v_i[k]$ is the instance of v_i at the kth step for $k \geq 0$ and c_i is a constant. The mean activation of each processing element is asynchronously updated. The system (15) is translated to synchronous updating by replacing index $k+n$ with k in the row of updating v_n

$$\begin{aligned}
v_1[k] &= g(v_2[k-1], v_3[k-1], v_4[k-1], \cdots, v_N[k-1], c_1) \\
v_2[k] &= g(v_1[k-1], v_3[k-2], v_4[k-2], \cdots, v_N[k-2], c_2) \\
v_3[k] &= g(v_1[k-2], v_2[k-1], v_4[k-3], \cdots, v_N[k-3], c_3) \\
&\vdots \\
v_n[k] &= g(v_1[k-n+1], v_2[k-n+2], \cdots, v_{n-1}[k-1], v_{n+1}[k-n], \cdots, v_N[k-n], c_n) \\
&\vdots \\
v_N[k] &= g(v_1[k-N+1], v_2[k-N+2], v_3[k-N+3], \cdots, v_{N-1}[k-1], c_N)
\end{aligned} \tag{16}$$

where $k \geq N$.

The matrix B can be determined by the flow chart in figure 5 for translating mean field dynamics to the following form

$$\mathbf{v}[k] = \tanh(\beta B \mathbf{u}[k]) \tag{17}$$

where

$$\mathbf{u}[k] = \begin{pmatrix} \mathbf{v}[k-1] \\ \mathbf{v}[k-2] \\ \vdots \\ \mathbf{v}[k-N+1] \end{pmatrix}$$

and

$$\mathbf{v}[k] = (v_1[k], v_2[k], \cdots, v_N[k])^T$$

denotes the mean configuration at the kth step.

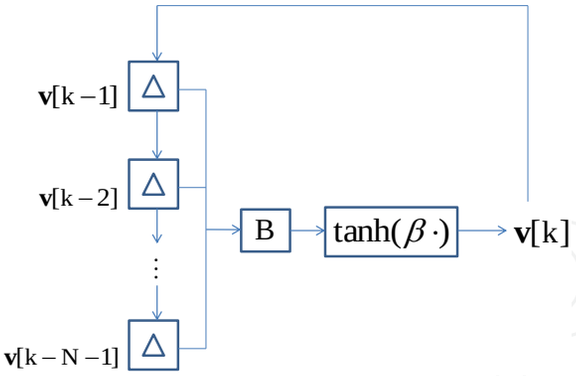


Figure 7. Nonlinear recurrent multilayer perceptrons.

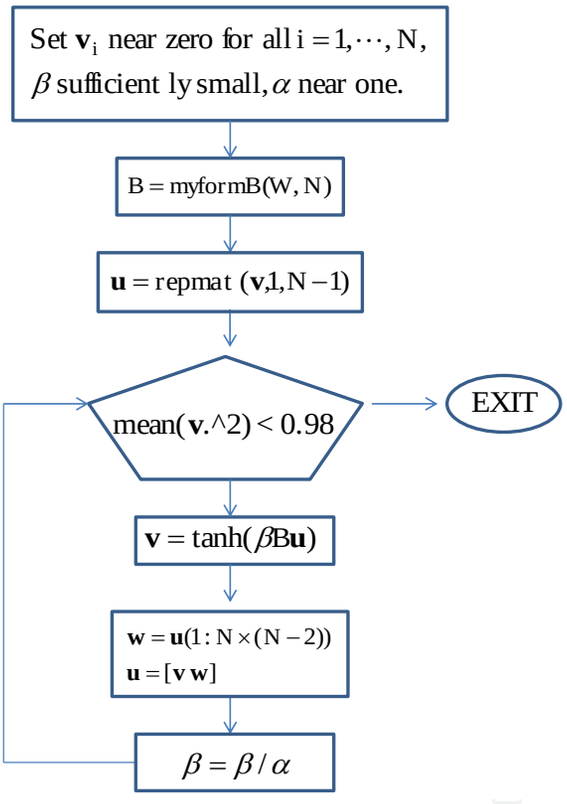


Figure 8. The flow chart of synchronous evolutionary simulations of mean field dynamics.

The structure of MIMO recurrent multilayer perceptrons is shown in Figure 7. The derived recurrent multilayer perceptrons track mean field dynamics by parallel and synchronous computations. As in the previous work [6], an annealing process is employed to schedule β from sufficiently small to large values for problem solving. Figure 8 shows the flow chart of simulating synchronous and parallel computations of recurrent multilayer perceptrons for tracking mean field dynamics.

http://dx.doi.org/10.5772/57217

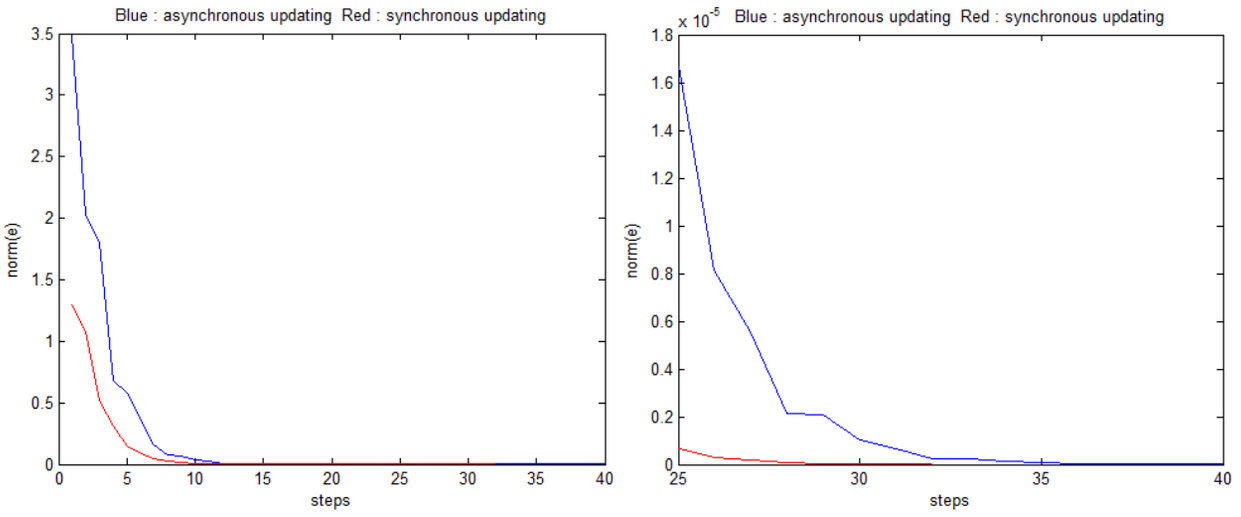


Figure 9. Errors of asynchronous update and asynchronous update along time steps.

3. Numerical simulation

3.1. Solving linear systems

The linear recurrent relation (14) is verified by numerical simulations for solving the following linear system,

$$\begin{aligned} x_1 &= 0 + \tfrac{1}{10}x_2 - \tfrac{1}{5}x_3 + 0x_4 + \tfrac{3}{5} \\ x_2 &= \tfrac{1}{11}x_1 + 0 + \tfrac{1}{11}x_3 - \tfrac{3}{11}x_4 + \tfrac{25}{11} \\ x_3 &= -\tfrac{1}{5}x_1 + \tfrac{1}{10}x_2 + 0 - \tfrac{1}{10}x_4 - \tfrac{11}{10} \\ x_4 &= 0 - -\tfrac{3}{8}x_2 + \tfrac{1}{8}x_3 + 0 + \tfrac{15}{8} \end{aligned}$$

The flow charts in figures 5 and 6 are implemented in Matlab codes. The initial value $\mathbf{x}[0] = [x_1[0], x_2[0], x_3[0], x_4[0]]$ is sampled from the hypercube $[-1,1]^4$ uniformly. The experiment simultaneously simulates asynchronous updating (11) and synchronous updating (14) of linear recurrence. Both asynchronous updating and synchronous updating attain the numerical solution $[1.0404, 1.991, -1.2067, 0.9775]^T$. Figure 9(a) shows errors of asynchronous updating and synchronous updating along time steps and (b) shows errors after the 25th step. The numerical results show the error of asynchronous updating coverages slower than that of synchronous updating. This illustrates the advantage of synchronous updating. When parallel computations like vector codes are employed, synchronous updating is more efficient than asynchronous updating for numerical simulations.

3.2. Graph bisection problem

The graph bisection problem [4] is stated to partition N nodes into two equal sets such that net edges crossing two sets in size is minimized. Let $s_i \in \{-1, 1\}$ denote the membership of the ith node to two non-overlapping sets and T_{ij} denote the connectivity, where

$$T_{ij} = \begin{cases} 1, \text{ if nodes } i \text{ and } j \text{ are connected} \\ 0, \text{ otherwise} \end{cases}$$

s_i denotes the partition of node i to two disjoint subsets. Node i is in one subset if $s_i = 1$ and belongs to the other if $s_i = -1$. As in [4], $E(\mathbf{s})$ for problem solving is given by ,

$$E(\mathbf{s}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} T_{ij}s_i s_j + \frac{a}{2}\left(\sum_{i=1}^{N} s_i\right)^2 \tag{18}$$

where a is the Lagrange multiplier which forces $\sum_{i=1}^{N} s_i$ to zero. $T_{ij}s_i s_j$ is zero if T_{ij} =0. Otherwise, it is 1 if nodes i and j belong the same subset and -1 if node i belongs to one set and node j to the other. Therefore, the first term quantifies the number of net edges crossing two subsets. The last forces equal cut. As in Appendix A, $E(s)$ can be rewritten as

$$E(s) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} W_{ij}s_i s_j \tag{19}$$

where $W_{ij} = T_{ij} - A$ and $W_{ii} = 0$.

We further explore the performances of synchronous updating by annealed recurrent multilayer perceptrons for graph bisection. In our simulations, each connection T_{ij} between nodes i and j is set to one if a uniform random number within $(0, 1)$ less than 0.2 is generated, and zero otherwise. The parameter a is 2. The halting condition is set to $\chi(\mathbf{v}) > 0.99$ where

$$\chi(\mathbf{v}) = \frac{1}{N}\sum_{i=1}^{N} v_i^2.$$

The temperature-like parameter β is always scheduled from sufficiently low to high values.

Figure 10 shows the convergence of annealed asynchronous updating (9) and annealed synchronous updating (17) for tracking mean field dynamics of solving a 100-nodes graph bisection problem, where the blue and red curves respectively show the change of the stability and $1/\beta$ along time steps. Figure 11 shows the change of cutsize and free energy by blue and red curve, respectively. The histograms of cutsize obtained by 50 executions of annealed asynchronous updating and annealed synchronous updating are plotted in Figure 12, where the mean of cutsize by annealed synchronous updating is 361.84, which is compatible to 358.5 of annealed asynchronous updating.

http://dx.doi.org/10.5772/57217

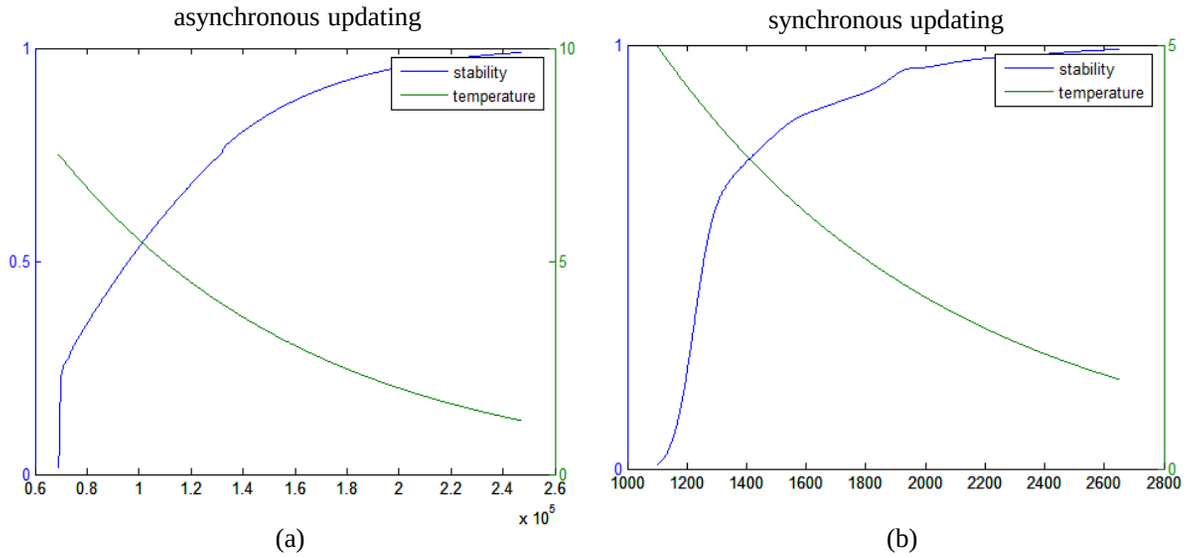


Figure 10. The change of the stability and $1/\beta$ for solving graph bisection problem by synchronous update and asynchronous update.

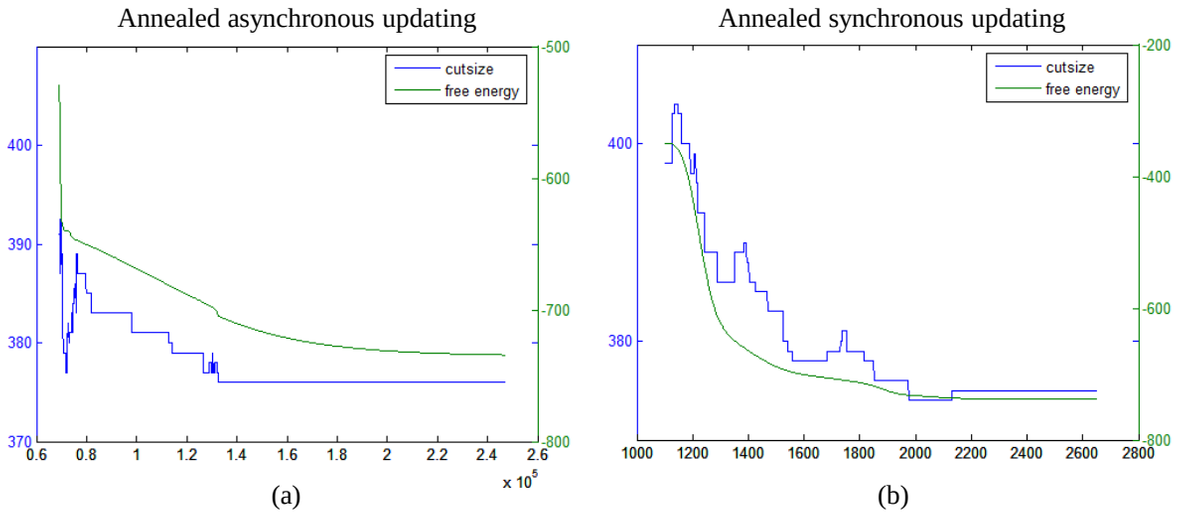
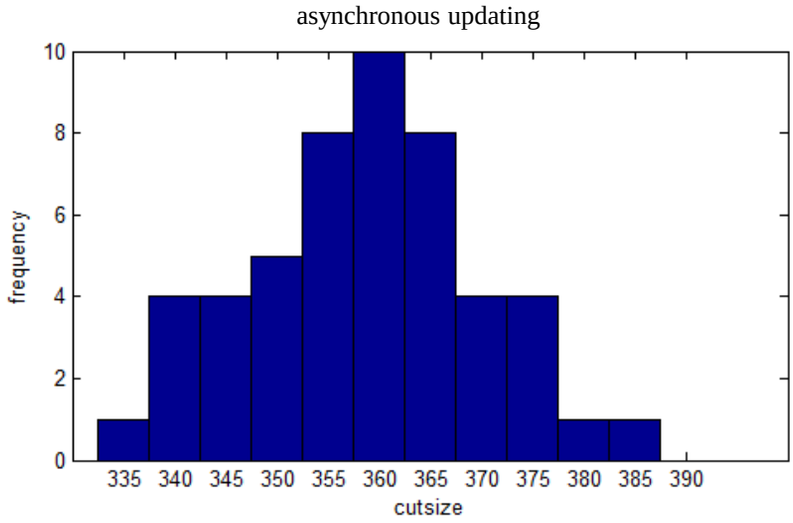


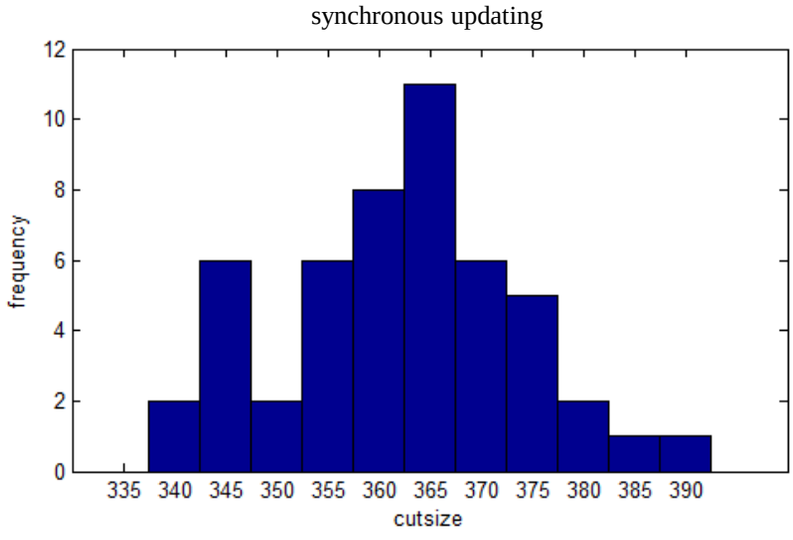
Figure 11. The change of cutsize and free energy for solving graph bisection problem by synchronous update and asynchronous update.

4. Parallel and distributed processes of tracking mean field dynamics of sparse connectivity

This section discusses the case of sparse interconnection among processing units. In the case, a processing connects only with processing units in a small neighborhood. Sparsely interconnected processing units are partitioned to K clusters such that the cutting size of interconnections crossing distinct clusters is minimized. This formulates a typical problem of K-set partition to a sparse graph. Mean field dynamics for K-set graph partition has been proposed in [6]. As argued previously, parallel and synchronous computations by recurrent multilayer perceptrons can be obtained for tracking mean field dynamics of resolving K-set graph partition. Let $\{S_k\}_{k=1}^{K}$ be the partitioned K clusters of sparsely interconnected



(a)



(b)

Figure 12. The histograms of cutsize obtained by 50 executions of synchronous update and asynchronous update.

processing units and $\mathbf{c}_k$ be the outer-input of processing units in S_k. $\mathbf{c}_k$ contains nonzero elements if there exists a processing unit in S_k that is connected with units not in S_k and those nonzero elements are determined by mean activations of processing units outside S_k. After K-set graph partition, all nodes are reindexed according to $\{S_k\}_{k=1}^K$. Ideally, there is dense connectivity among processing units inside each S_k and sparse connectivity among $\{S_k\}_{k=1}^K$ through $\{\mathbf{c}_k\}_{k=1}^K$ as illustrated in Figure 13.

In each cluster S_k when there is a processing unit connecting to processing units outside S_k according to the approach in section 2, all processing units inside S_k are evaluated directly by synchronous computations for fixed $\mathbf{c}_k$. The approach which combines synchronous update of mean activations in side each S_k and sequential update among $\{S_k\}_{k=1}^K$ is proposed for

http://dx.doi.org/10.5772/57217

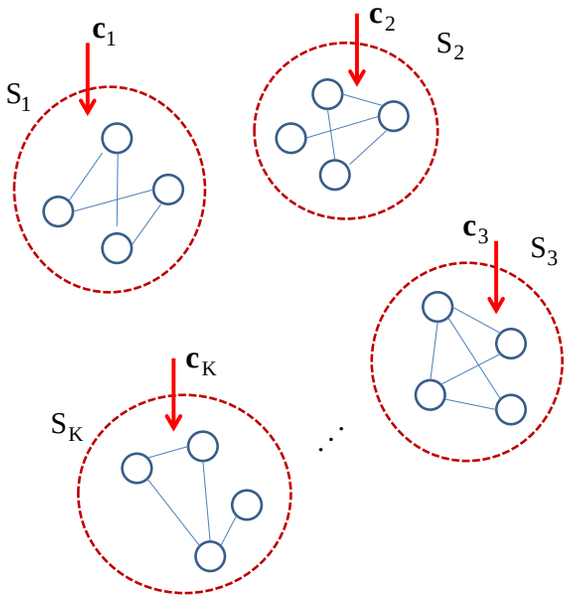


Figure 13. Partition of all nodes into K clusters to attain dense interconnection in each cluster.

tracking mean field dynamics sparse connectivity. The idea follows parallel and distributed processes. This approach decomposes a large system to several sparsely connected small systems, updates mean activations inside each small system synchronously and updates decomposed systems sequentially. Suppose that each S_k has the same number of nodes. The size of nodes in S_k is $|S_k| = \frac{N}{K} \ll N$. Figure 14 shows the flow chart of the proposed approach. The halting condition states to compare the stability $\chi(\mathbf{v})$ with a threshold. An example with $N = 12$ for illustrating decomposition of a sparse system to three small systems is given in Appendix C.

5. Conclusions

This paper has proposed a novel approach for tracking mean field dynamics by synchronous computations of recurrent multilayer perceptrons. The strategy is to introduce time delays and auxiliary variables and constructs equivalent recursive relations. This strategy essentially constructs recurrent multilayer perceptrons for tracking densely coupled mean field dynamics. The proposed approach is also extended to deal with large-scale sparsely interconnected mean field dynamics. In the beginning, all processing units are partitioned into K clusters by solving graph partition. The task is decomposed to K subtasks of synchronous computations and different clusters are sparsely connected by outer-inputs. The work combines synchronous updating inside each cluster with sequential updating among K clusters.

Numerical simulations show that the proposed approach has successfully translated mean field equations of solving the graph bisection problem to a system of post-nonlinear recursive functions, and verified the consistency between the original mean field equations and corresponding recurrent computations.

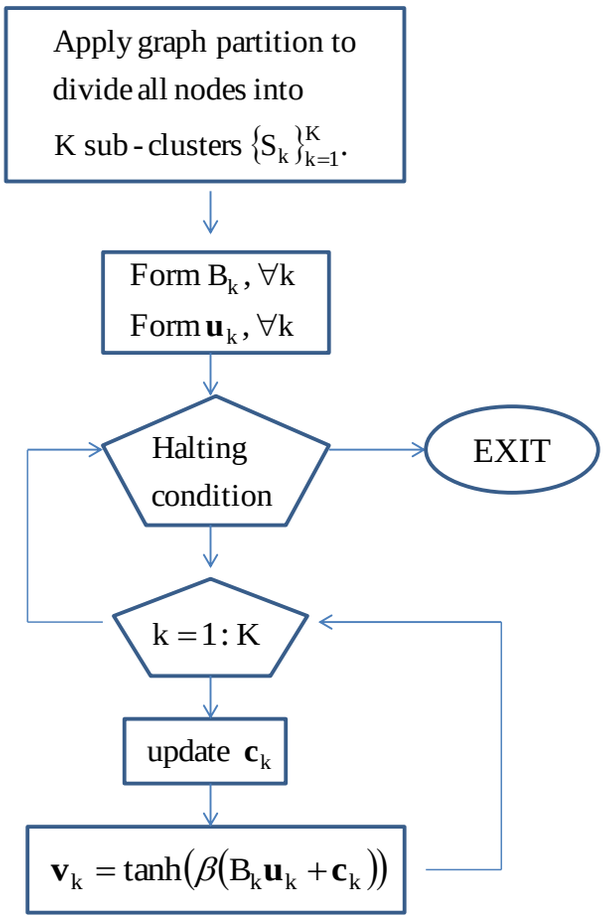


Figure 14. The flow chart of parallel and distributed processes for tracking mean field dynamics of sparse connectivity.

6. Appendix

6.1. Appendix A. Rewriting energy function of graph bisection problem

$$
\begin{aligned}
E(S) &= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} T_{ij}s_i s_j + \frac{A}{2}\left(\sum_{i=1}^{N} s_i\right)^2 \\
&= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} T_{ij}s_i s_j + \frac{A}{2}\left(\sum_{i=1}^{N} s_i^2 + \sum_{i=1}^{N}\sum_{j\neq i}^{N} s_i s_j\right) \\
&= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} (T_{ij} - A)s_i s_j + \frac{A}{2}\left(\sum_{i=1}^{N} s_i^2\right) \\
&= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} (T_{ij} - A)s_i s_j + \frac{A}{2}N
\end{aligned}
$$

http://dx.doi.org/10.5772/57217

Let $W_{ij} = T_{ij} - A$ where $W_{ii} = 0$. Since $\frac{A}{2}N$ is a constant, the energy function is rewritten as

$$E(S) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} W_{ij}s_i s_j$$

6.2. Appendix B. An example decomposing sparse interconnection

A linear system is given by

$$\begin{array}{rcccccccccccccccc}
x_1 = & 0 & + & a_{12}x_2 & + & a_{13}x_3 & + & 0 & + & 0 & + & 0 & + & 0 & + & 0 & + & 0 \\
x_2 = & a_{21}x_1 & + & 0 & + & a_{23}x_3 & + & 0 & + & 0 & + & 0 & + & 0 & + & 0 & + & 0 \\
x_3 = & a_{31}x_1 & + & a_{32}x_2 & + & 0 & + & 0 & + & 0 & + & 0 & + & 0 & + & a_{38}x_8 & + & 0 \\
x_4 = & 0 & + & 0 & + & 0 & + & 0 & + & a_{45}x_5 & + & a_{46}x_6 & + & 0 & + & 0 & + & 0 \\
x_5 = & 0 & + & 0 & + & 0 & + & a_{54}x_4 & + & 0 & + & a_{56}x_6 & + & 0 & + & 0 & + & 0 \\
x_6 = & 0 & + & a_{62}x_2 & + & 0 & + & a_{64}x_4 & + & a_{65}x_5 & + & 0 & + & 0 & + & 0 & + & 0 \\
x_7 = & a_{71}x_1 & + & 0 & + & 0 & + & 0 & + & 0 & + & 0 & + & 0 & + & a_{78}x_8 & + & a_{79}x_9 \\
x_8 = & 0 & + & 0 & + & 0 & + & 0 & + & 0 & + & 0 & + & a_{87}x_7 & + & 0 & + & a_{89}x_9 \\
x_9 = & 0 & + & 0 & + & 0 & + & 0 & + & 0 & + & 0 & + & a_{97}x_7 & + & a_{98}x_8 & + & 0
\end{array}$$

Let

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix}
0 & a_{12}x_2 & a_{13}x_3 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_{21}x_1 & 0 & a_{23}x_3 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_{31}x_1 & a_{32}x_2 & 0 & 0 & 0 & 0 & 0 & a_{38}x_8 & 0 \\
0 & 0 & 0 & 0 & a_{45}x_5 & a_{46}x_6 & 0 & 0 & 0 \\
0 & 0 & 0 & a_{54}x_4 & 0 & a_{56}x_6 & 0 & 0 & 0 \\
0 & a_{62}x_2 & 0 & a_{64}x_4 & a_{65}x_5 & 0 & 0 & 0 & 0 \\
a_{71}x_1 & 0 & 0 & 0 & 0 & 0 & 0 & a_{78}x_8 & a_{79}x_9 \\
0 & 0 & 0 & 0 & 0 & 0 & a_{87}x_7 & 0 & a_{89}x_9 \\
0 & 0 & 0 & 0 & 0 & 0 & a_{97}x_7 & a_{98}x_8 & 0
\end{bmatrix}$$

be a sparse matrix and

$$\begin{aligned}
S_1 &= \{x_1, x_2, x_3\} \\
S_2 &= \{x_4, x_5, x_6\} \\
S_3 &= \{x_7, x_8, x_9\} \\
\mathbf{v}_1 &= [x_1\ x_2\ x_3]^T \\
\mathbf{v}_2 &= [x_4\ x_5\ x_6]^T \\
\mathbf{v}_3 &= [x_7\ x_8\ x_9]^T
\end{aligned}$$

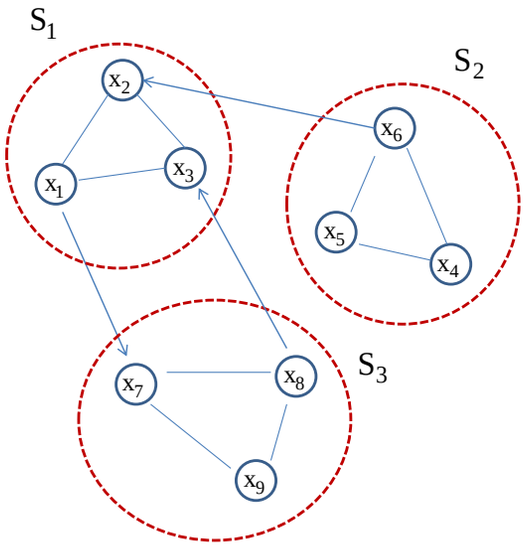


Figure 15. Dense interconnection in each cluster and sparse interconnection among three clusters.

Based on graph partition of $K = 3$, the system $\mathbf{x} = A\mathbf{x}$ has dense interconnection of S_k, $\forall k = 1, 2, 3$ and sparse interconnection among $\{S_k\}_{k=1}^3$ as shown in Figure 15. Let

$$\begin{aligned} d_1 &= d_2 = d_4 = d_5 = d_8 = d_9 = 0 \\ d_3 &= a_{38}x_8 \\ d_6 &= a_{62}x_2 \\ d_7 &= a_{71}x_1 \end{aligned}$$

and $\mathbf{c}_1 = [d_1\ d_2\ d_3]^T$, $\mathbf{c}_2 = [d_4\ d_5\ d_6]^T$ and $\mathbf{c}_3 = [d_7\ d_8\ d_9]^T$ be the outer-input of three clusters $\{S_k\}_{k=1}^3$. d_i is nonzero if there is a node x_j connected to x_i with weight $a_{ij} \neq 0$ where x_i and x_j belong to different clusters. Therefore, the updating rule of $\{\mathbf{c}_k\}_{k=1}^3$ is

$$\mathbf{c}_k = \sum_{j \neq k}^{3} A_{kj}\mathbf{v}_j$$

Author details

Jiann-Ming Wu*, Jung-Chao Ban and Chun-Chang Wu

*Address all correspondence to: jmwu@livemail.tw

Department of Applied Mathematics, National Dong Hwa University, Shoufeng, Taiwan

http://dx.doi.org/10.5772/57217

References

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", Proceedings of the National Academy of Sciences 79:2554-2558, 1982.

[2] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons", Proceedings of the National Academy of Sciences 81: 3088-3092, 1984.

[3] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," Biol. Cybern., vol. 52, p. 141, 1985.

[4] C. Peterson and B. Söerberg, "A new method for mapping optimization problems onto neural network," Int. J. Neural Syst., vol. 1, p. 3, 1989.

[5] J. M. Wu, "Potts models with two sets of interactive dynamics," Neurocomput., vol. 34, pp. 55–77, Sept. 2000.

[6] J. M. Wu, "Annealing by Two Sets of Interactive Dynamics," IEEE Trans. on Systems Man and Cybernetics Part B-Cybernetics 34 (3): 1519-1525, Jun 2004.

[7] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh, "The capacity of the Hopfield associative memory," IEEE Truns. Inform. Theory, vol. IT-33, pp. 1-33, July 1987

[8] T. Isokawa, H. Nishimura, N. Kamiura, and N. Matsui, "Associative memory in quaternionic Hopfield neural network," Int. J. Neural Syst., vol. 18, no. 2, pp. 135–145, 2008

[9] D. W. Tank and J. J. Hopfield, "Collective computation in neuronlike circuits," Sci. Amer., vol. 257, no. 6, pp. 104–115, 1987

[10] C. Y. Liou and J. M.Wu, "Self-organization using Potts models," Neural Netw., vol. 9, no. 4, pp. 671–684, 1996.

[11] K. Rose, E. Gurewitz, and G. C. Fox, "Statistical mechanics and phase transitions in clustering," Phys. Rev. Lett., vol. 65, no. 8, pp. 945–948, 1990

[12] K. Rose, "Constrained clustering as an optimization method," IEEE Trans. Pattern Anal. Machine Intell., vol. 15, pp. 785–794, Aug. 1993.

[13] J. M. Wu and S. J. Chiu, "Independent component analysis using Potts models," IEEE Trans. Neural Networks, vol. 12, pp. 202–211, Mar. 2001.

[14] A. V. Rao, D. J. Miller, K. Rose, and A. Gersho, "A deterministic annealing approach for parsimonious design of piecewise regression models," IEEE Trans. Pattern Anal. Machine Intell., vol. 21, pp.169–173, Feb. 1999.

[15] J. M. Wu, Z. H. Lin and P. H. Hsu, "Function approximation using generalized adalines," IEEE Trans. On Neural Networks,Vol.17 No.3, 541-558, May 2006

[16] J. M. Wu, "Fetal electrocardiogram extraction by annealed expectation maximization," Neurocomputing 71, pp 1500-1514, 2008

[17] J. M. Wu, "Multilayer Potts perceptrons with Levenberg-Marquardt learning", accepted by IEEE Trans. on Neural Networks, 2008

[18] J. M. Wu, M.H. Chen, Lin Z.H., "Independent component analysis based on marginal density estimation using weighted Parzen windows", accepted by Neural Networks, 2008

[19] Dudul SV, "Prediction of Lorenz chaotic attractor using two layer perceptron neural network," Applied Soft Computing 2005; 5:333–355

[20] G. Peter Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," Neurocomputing 50 (2003) 159 – 175.