A convergence result on fully asynchronous discrete-time discrete-state dynamic networks

Jacques M. Bahi and Sylvain Contassot-Vivier
LIFC, IUT de Belfort-Montbéliard, BP 527, 90016, Belfort cedex, France
E-mail: {Jacques.Bahi, Sylvain.Contassot-Vivier}@iut-bm.univ-fcomte.fr

Abstract

We consider networks of a large number of neurons (or units, processors, ...), whose dynamics are fully asynchronous with overlapping updating. We suppose that the neurons take a finite number of states (discrete states) and that the updating scheme is discrete in time. We make no hypotheses on the activation function of the neurons, the networks may have multiple cycles and basins and we derive conditions on the initialization of the networks which ensures convergence to fixed points only. Application to a fully asynchronous Hopfield neural network allows us to validate our study.

Keywords: Dynamic neural networks, asynchronism, stability.

1 Introduction

Consider a n-neuron network such that each neuron i takes a finite number of values x_i, $i \in \{1, ..., n\}$. Decompose it into α blocks, every one containing n_i neurons. Let X_i be the value of the block i and $X = (X_1, ..., X_\alpha)$ be the value of global network. Suppose that the dynamic of the network is described by an activation function F :

$$F(X) = (F_1(X_1, ..., X_\alpha), ..., F_\alpha(X_1, ..., X_\alpha)).$$

The global state of the network at the discrete time t is denoted by

$$X^t = \left(X_1^t, ..., X_\alpha^t\right).$$

The activation functions we consider are general so we are not restricted to threshold networks. We suppose that the network is fully asynchronous with overlapping communications in the sense of [20], that is

- the block neurons of the network may be updated in a random order and moreover it is possible that some neurons are not updated at some times.
- at a time t, if a neuron doesn't have the state at the time $t-1$ of a connected neuron, then it updates its own state using the last received information from this element rather than waiting for its $(t-1)^{th}$ state.

The result of this paper is of theoretical nature but it is directly related to a question of practical interest. The goal is not to built a network ensuring the global convergence to a stable state. Indeed, even if a network converges to a stable state, for example in fully parallel or block sequential modes, fully asynchronous with overlapping dynamics often destroy this convergence. So, we consider general networks which may have multiple cycles and multiple stable states, and then we give sufficient conditions on their initialization to ensure their convergence to stable states only, thus the potential cycles are avoided whatever the kind of their dynamics.

Fully asynchronous networks including overlapping updating were characterized by Herz and Marcus in [20], they are described by the following algorithm

Algorithm 1

Given an initial state $X^0 = \left(X_1^0, ..., X_\alpha^0\right)$
At each time t, for each element i :
either i doesn't update its state: $X_i^{t+1} = X_i^t$
or i updates its state using the available received states of connected elements.

In the sequel we denote by $J(t)$ the set of neurons updated at the time t and by $X_j^{s_j^i(t)}$ the state of the group j of neurons available for the group i at the time $t : s_j^i(t) = t - r_j^i(t) \leq t$, $r_j^i(t)$ denotes the delay of the group of neurons j with respect the group i. Some groups of neurons may be reduced to single neurons. Herz and Marcus studied the global dynamics of neural networks with distributed dynamics. They considered only non overlapping updating and they pointed out that the evolution of the neural network is strongly influenced by the communication delays between the individual neurons, and that the global stability analysis in the overlapping updating case remains an open problem. Indeed, the behavior of fully asynchronous networks is more complex and depends on the delays and on the activated neurons
In [29], the authors consider a discrete-time symmetric neural networks, they give sufficient conditions to have cycle-free dynamics of such networks. The distributed dynamics they study also corresponds to the non overlapping updating case.
The term asynchronous algorithm was used by several authors to refer to non synchronous algorithms. A. Bhaya et al. [18] studied the stability of continuous-valued discrete-time asynchronous Hopfield neural networks under the assumption of D-stability of the interconnection matrix and the standard assumptions on the activation functions. V.S. Kozyakin et al. [21] studied, in a more general context, global stability of a class of total asynchronous discrete-time continuous-valued nonlinear systems. In these last two papers, the model of asynchronism were introduced by Takeda and Goodman [27], asynchronous networks are also called by the authors desynchronized networks and are obtained as a particular case of fully asynchronous networks with overlapping by taking $r_j^i(t) = 0$.
Fully asynchronous networks with overlapping updating incorporate, as special cases, all the above networks and particularly, sequential, parallel and block-sequential networks (see e.g. [19, 16, 17, 26] in the case of threshold networks).

To study the stability of such asynchronous systems we define a distance between two states x and y of the network as follows

$$d\left(x, y\right) = \left(\delta_1\left(x_1, y_1\right), ..., \delta_n\left(x_n, y_n\right)\right) \tag{1}$$

where

$$\delta_i\left(x_i, y_i\right) = \begin{cases} 1 & if \quad x_i \neq y_i \\ 0 & if \quad x_i = y_i. \end{cases}$$

This so-called vectorial distance was introduced by F. Robert [11, 12] in the context of component-wise chaotic discrete boolean iterations, which is a particular case of blocks asynchronous iterations on finite sets. In the numerical analysis community, asynchronous algorithms have been addressed [8, 5, 9, 6, 23], however these algorithms do not apply to discrete-state neural networks where multiple basins of attraction are vital, because of the strong hypotheses the continuous-state model requires. Indeed, convergence results are based on contraction hypotheses with respect to the "maximum" scalar distance

$$\delta\left(x, y\right) = \max_{1 \leq i \leq n}\left(\delta_i\left(x_i, y_i\right)\right),$$

and it is easy to see that in the discrete finite context, only constant functions satisfy this hypothesis.
In [2, 3], we only study global stability of boolean networks (i.e. when there exists only one stable state), the result of [4, 7] becomes a particular case of the present results: we generalize the study to finite-state networks and mainly, under less restrictive hypotheses, we include new initial configurations of the network leading to its stabilization.

Furthermore, in order to give a practical use of our main result, we wrote a code based on it which test the convergence of finite discrete-time discrete-state fully asynchronous networks and we applied it to a special case of a Hopfield network. This is discussed in the second part of the paper.

The following of the article is organized as follows. In order to be more comprehensible, this article is self-contained and then preliminary results concerning boolean matrices and basic properties of the distance d are presented in section 2. The asynchronous discrete model is formulated in section 3. The next section is devoted to our main result and its application in the form of a testing code applied on a Hopfield neural network. The contribution of this result in the context of Hopfield networks is given in section 5. Finally, to evaluate the efficiency of our testing code, its results are confronted to a simulation code in section 6. The conclusion over this study is given in section 7 and the last section is devoted to the detailed and rigorous proofs of technical lemmas and our main result.

2 Preliminary tools

2.1 Specific results

Definition 2.1 *Consider a n-neuron network whose activation function is denoted by f. The boolean matrix $B = B(f)$ associated with this network is defined by its general term b_{ij}, $i, j \in \{1, ..., n\}$, such that*

$$b_{ij} = \begin{cases} 1 \text{ if the neuron } x_j \text{ informs the neuron } x_i \\ 0 \text{ otherwise.} \end{cases}$$

As $B(f)$ is a boolean matrix, its only possible eigenvalues are 0 or 1. $B(f)$ is not necessary symmetric since the graph of the network is not necessary undirected.

Example 1 *Consider a network with three neurons 1, 2 and 3. Assume that the notation $1 \to 2$, means that neuron 1 informs neuron 2. If these neurons are connected as in the figure 1*

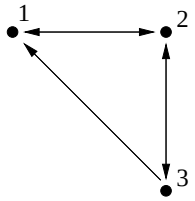

Figure 1: Connection graph

then its associated matrix is

$$B = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Definition 2.2 *A function f is a contraction if the associated matrix $B(f)$ has all eigenvalues equal to 0, i.e. if the spectral radius $\rho(B)$ of B is equal to 0. $B(f)$ is then called a contraction matrix.*

Proposition 2.3 *[11, 12] A square boolean matrix B with n rows and n columns is a contraction matrix if and only if $B^n = 0$. This is equivalent to the existence of a permutation matrix P such that $P^T BP$ is a strictly lower triangular matrix where P^T is the transpose of P.*

Remark 1 *An activation function f describing the dynamic of the system is a contraction if and only if its connection graph has no cycles. Note that there are an infinity of activation functions associated with a given connection graph.*

2.2 Decomposition of the discrete system

Consider n neurons i, $i \in \{1, ..., n\}$, decompose them into α groups $j \in \{1, ..., \alpha\}$, each of them containing n_j neurons. So $\sum_{j=1}^{\alpha} n_j = n$.
Suppose that each state x_i of a neuron i takes its states in a finite set E_i, then the n-network takes its global state in the product set $E = \prod_{i=1}^{n} E_i$.
Denote by X_j the value of the group j of neurons, and by $(X_1, ..., X_\alpha) \in E$ the global value of the network.

Definition 2.4 *We define the block vectorial distance as follows*

$$d(X, Y) = (\delta_1(X_1, Y_1), ..., \delta_\alpha(X_\alpha, Y_\alpha)), \tag{2}$$

where

$$\delta_i(X_i, Y_i) = \begin{cases} 1 & if \quad X_i \neq Y_i \\ 0 & if \quad X_i = Y_i. \end{cases}$$

Definition 2.5 *Let $X = (X_1, ..., X_\alpha)$ and $\widetilde{X}^j = (X_1, ..., X_{j-1}, Y_j, X_{j+1}, ..., X_\alpha)$ with $Y_j \neq X_j$. Let's Define the first neighborhood of X as the set*

$$V_1(X) = \left\{X, \quad \widetilde{X}^j, \quad j = 1, .., \alpha\right\}$$

Example 2 *This example gives an illustration of the above definition in $\{0, 1\}^3$. Consider a discrete system with 3 components. Therefore, there are 2^3 possible states as shown in figure 2: $A = (0, 0, 0)$, $B = (0, 0, 1)$, $C = (0, 1, 0)$, $D = (0, 1, 1)$, $E = (1, 0, 0)$, $F = (1, 0, 1)$, $G = (1, 1, 0)$, $H = (1, 1, 1)$*

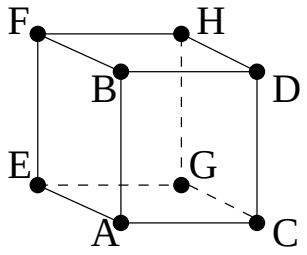

Figure 2: States of a discrete system with 3 components

then $V_1(D) = \{B, C, D, H\}$.

Definition 2.6 *A chain $C = [X, U_1, ..., U_{r-1}, Y] = [X, ..., Y]$ of length $l(C) = r$ is a sequence of elements of E, such that*

$$U_1 \in V_1(X), Y \in V_1(U_{r-1}), U_{i+1} \in V_1(U_i), \quad i \in \{1, ..., r-2\}.$$

Definition 2.7 *Denote D, the set of all possible chains $C = [X, ..., Y]$ for given $X, Y \in E$. A minimal chain is a chain of length m such that*

$$m = \min_{C \in D} l(C).$$

A minimal chain always exists because E is a finite set.

Proposition 2.8 *[11, 12]. For a minimal chain $[X, U_1, ..., U_{r-1}, Y]$,*

$$d(X, U_1) + ... + d(U_{r-1}, Y) = d(X, Y).$$

2.3 The discrete derivative of an activation function

We define the discrete derivative of F as follows

Definition 2.9 *The discrete derivative of F on X is defined by the matrix*

$$F'(X) = \left((F'(X))_{ij} \right)_{1 \leq i,j \leq \alpha},$$

such that

$$\begin{cases} (F'(X))_{ij} = 1 \ \ if \ \ \exists j \in \{1, ..., \alpha\} \ \ such \ that \ F_i(X) \neq F_i(\widetilde{X}^j) \\ (F'(X))_{ij} = 0 \ \ if \ \ \forall j \in \{1, ..., \alpha\}, F_i(X) = F_i(\widetilde{X}^j) \end{cases}$$

Remark 2 *$F'(X) = 0$ if and only if F is a constant function on $V_1(X)$.*

The two following results will be used in the proof of the main result of this paper.

Proposition 2.10 *[11, 12]. $\forall X \in E$, $\forall Y \in V_1(X)$,*

$$d(F(X), F(Y)) \leq F'(X) d(X, Y)$$

Proposition 2.11 *[11, 12]. Consider a minimal chain $[X, U_1 ... U_{r-1}, Y]$, then*

$$d(F(X), F(Y)) \leq \sup_{Z \in \{X, U_1 ... U_{r-1}\}} (F'(Z))\, d(X, Y)$$

where the sup is defined element by element.

3 Fully asynchronous networks with overlapping

Definition 3.1 *Consider a n-neuron network decomposed into α block-neurons. Let the strategy $J = \{J(t)\}_{t \in \mathbb{N}}$ be a sequence of non-empty subsets of the block-neurons set $\{1, ..., \alpha\}$.*
For $i \in \{1, ..., \alpha\}$, let $S^i = \left\{ s_1^i(t), ..., s_\alpha^i(t) \right\}_{t \in \mathbb{N}}$, be a sequence of $\mathbb{N}^\alpha$, such that

(c1) $s_j^i(t) = t - r_j^i(t)$ with $0 \leq r_j^i(t) \leq t$, $r_j^i(t)$ being the delay of the block j compared to the block i at the discrete time t.

(c2) $\forall i, j \in \{1, ..., \alpha\}$, $\lim_{t \to \infty} s_j^i(t) = \infty$, i.e. the delays associated with the block i are unbounded but follow the evolutions of the system.

(c3) No block-neurons may be neglected on average by the updating rule. This condition is called fair sampling condition and is equivalent to:
$\forall i \in \{1, ..., \alpha\}, \mathrm{Card}(\{t, i \in J(t)\}) = \infty$.

Then, the fully asynchronous dynamic of the n-neuron network according to the activation function F and to the strategy J and with initial configuration $X^0 = \left(X_1^0, ..., X_\alpha^0\right)$ is described by the algorithm

Algorithm 2

Given an initial state $X^0 = \left(X_1^0, ..., X_\alpha^0\right)$
at each time-step $t = 0, 1, ...$
For each block-components $i = 1, ..., \alpha$
if $i \in J(t)$ then
$X_i^{t+1} = F_i\left(X_1^{s_1^i(t)}, ..., X_\alpha^{s_\alpha^i(t)}\right)$
else
$X_i^{t+1} = X_i^t$

If the block i at the time t belongs to the strategy $J(t)$ then its state X_i^{t+1} is updated by F_i at the time $t+1$. X_i^{t+1} depends on the states $X_j^{s_j^i(t)}$ of blocks j available at the time t ($s_j^i(t) \leq t$). If the block i does not belong to the strategy $J(t)$ then the block i is not updated.
The following definition is useful in the proof of the convergence theorem 4.1.

Definition 3.2 *Consider the strictly increasing sequence of integers $\{p_l\}_{l\in\mathbb{N}}$ defined as follows:*

- $p_0 = 0$,
- *p_1 is the minimum integer such that all the components are updated at least one time,*
- *for all integer l, p_l is the minimum integer such that all the components are updated at least l times*

This sequence $\{p_l\}$ is well defined thanks to conditions (c2) and (c3) of definition 3.1.

4 Convergence result and its applications

4.1 The main result

The following result is proved in [1].

Theorem 4.1 *Let a discrete-time discrete-state dynamic n-neuron network be partitioned into α block of neurons and denote by $F = (F_1, ..., F_\alpha)$ its activation function defined on a finite product set $E = \prod_{i=1}^{\alpha} E_i$. Let X^* be a fixed point of F and $V = \prod_{i=1}^{\alpha} V_i$ where $V_i \subseteq E_i$ be a neighborhood of X^*.*
Suppose that

(h1) $F(V) \subset V$.

(h2) the following boolean matrix is a contraction

$$M = \sup_{z\in V} \{F'(z)\}.$$

Let $i_1, i_2, ..., i_q$ denote the null columns indexes of M. (Hypothesis (h2) ensures the existence of at least one null column).
Denote by

$$W^l = V_1 \times ... \times V_{i_l-1} \times E_{i_l} \times V_{i_l+1} \times ... \times V_\alpha,$$

then any fully asynchronous dynamic with overlapping updating of the network reaches the stable state X^* within at most $p_1 + p_\alpha$ steps if its initial state $X^0 \in W^l$ with $l \in \{1, ..., q\}$.

4.2 An arbitrary network example

Let a discrete dynamic network with $n = 3$ components on $E = \{0, 1, 2\}^3$ (three possible states for each neuron) described by the function G given in table 4.2.

The fully parallel graph of the network is given below
Denote by

$$V = \{0, 1\} \times \{0, 2\} \times \{0, 1\},$$

then

$$V = \{(0, 0, 0), (0, 0, 1), (0, 2, 0), (0, 2, 1), (1, 0, 0), (1, 0, 1), (1, 2, 0), (1, 2, 1)\},$$

and

$$M = \sup_{z\in V} \{F'(z)\} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}.$$

i	x	$G(x)$	i	x	$G(x)$	i	x	$G(x)$
0	000	000	9	100	021	18	200	012
1	001	000	10	101	021	19	201	012
2	002	000	11	102	021	20	202	210
3	010	001	12	110	022	21	210	211
4	011	002	13	111	021	22	211	012
5	012	202	14	112	222	23	212	212
6	020	001	15	120	020	24	220	012
7	021	001	16	121	020	25	221	012
8	022	001	17	122	020	26	222	112

Table 1: Function G

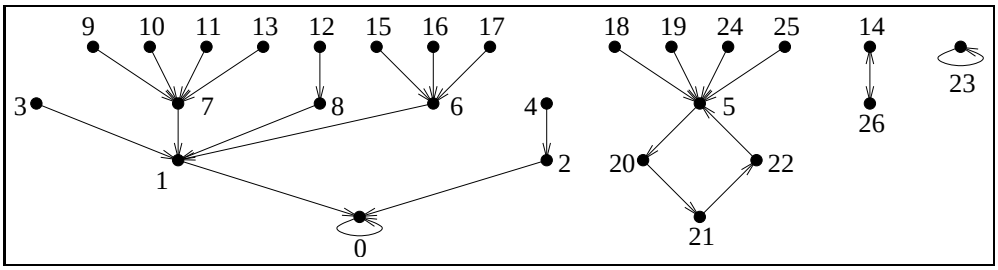

Figure 3: Parallel dynamic

The last column is null so

$$W = \{0,1\} \times \{0,2\} \times \{0,1,2\}\,.$$

Even if $F'(0,0,2)$, $F'(0,2,2)$, $F'(1,0,2)$, $F'(1,2,2)$ are not contractions, indeed

$$F'(0,0,2) = F'(0,2,2) = F'(1,0,2) = F'(1,2,2) = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}.$$

Theorem 4.1 claims that any asynchronous evolution started from any $X^0 \in W$ reaches the stable state 0.

For the practical use of the theorem, the following subsection shows how to verify the hypotheses in the case of Hopfield networks.

4.3 Particularization to Hopfield Networks

To construct the network, we have chosen the Hebb rule since it is the most common but we could have chosen any other way to construct the network and then any kind of interconnection matrix.
Let $X^{1*}, ..., X^{k*} \in \{0,1\}^n$, the patterns to be memorized by a Hopfield neural network of n boolean neurons. A symmetric matrix of dimensions $n \times n$ and real coefficients is then computed by $W = XX^T$ where $X = (X^{1*}, ..., X^{k*})$. This matrix informs us about the way the neurons are connected at each others. The activation function of the network composed of the n neurons is then

$$y = F(x) = Sgn(Wx - T),\ x = (x_1, ..., x_n) \in \{0,1\}^n$$

so

$$y_i = sgn(\sum_{k=1}^{n} w_{ik}x_k - T_i)$$

where $Sgn(x) = (sgn(x_1), ..., sgn(x_n))^T$ and $sgn(a)$ is the sign function defined by

$$sgn(a) = \begin{cases} +1 \ if \ a \geq 0 \\ 0 \ if \ a < 0 \end{cases}$$

- How to get fully parallel Hopfield networks ?
  - by taking $J(t) = \{1, ...\alpha\}$ and $s_j^i(t) = t$ in algorithm (2). To obtain single neurons dynamics then take $\alpha = n$, $(n_i = 1)$.

- How to get block-sequential Hopfield networks ?
  - By taking $J(t) = \{t + 1(\mathrm{mod}\, n)\}$ and $s_j^i(t) = t$ in (2).

- Can theorem generalizes Hopfield's results [13, 14] or Golès et al. [19] results ?
  - No, theorem 4.1 gives all the initial iterates which make the neural network globally convergent whatever its dynamic, so initial states which leads to the global convergence in the fully parallel or block-sequential cases and not in fully asynchronous case will not be obtained.

- How to know that the fully asynchronous network with overlapping updating initialized with $X^0 = \left(x_0^0, ..., x_n^0\right)$ leads to global convergence to an associative memory $X^* = (x_0^*, ..., x_n^*)$?
  - by verifying the hypotheses of theorem 4.1:

  1) Compute a cartesian product V which contains X^* and X^0 :

     1.a. for all i compare x_i^* to x_i^0, then
     - if $x_i^* = x_i^0$, then $V_i = \{x_i^0\}$
     - if $x_i^* \neq x_i^0$, then $V_i = \{0, 1\}$

     1.b. so $V = V_1 \times V_2 \times .... \times V_n$

  2) Verify that for all X being in V, $F(X) \in V$

  3) For all $X = (x_1, ..., x_j, ..., x_n)$ in V, compute $F'(X)$ (the derivative of F on X) :

     3.a. for all j in $\{1, ...n\}$ compare $F(x_1, ...x_j, ..., x_n)$ to $F(\widetilde{X}^j) = F(x_1, ...\overline{x}_j, ..., x_n)$, then:

     3.b. the j^{th} column of $M_X = F'(X)$ is equal to the column-vector $F(X) - F(\widetilde{X}^j)$.

  4) Compute $\sup M_X$ and denote by $i_1, i_2, ..., i_q$ the null columns indexes of M_X then:

     4.a. if $(M_X)^n$ is a null matrix (i.e. $\sup M_X$ is a contraction), then the fully asynchronous Hopfield network converges to X^* whatever its initialization in V and particularly from X^0.

  5) Theorem 4.1 claims moreover that if the Hopfield network is initialized by any initial state in the form

     $$Y^0 = \left(x_0^0, ..., y_{i_j}, ..., x_n^0\right) \ \ with \ j \in \{1, ..., q\}$$

     is globally convergent to X^* whatever the values of y_{i_j} and whatever its dynamics.

Remark that even if condition 4.a. is not satisfied, then the fully asynchronous Hopfield network will be globally convergent if X^0 is in the form of Y^0. The convergence procedure test introduced below, include the verification of this last point and then, gives us all the initial iterates which make the network globally convergent to one of its memories. This convergence procedure is applicable to any type of Hopfield network and includes the case where the matrix W is not symmetric.

4.4 Convergence procedure test

To simplify the study of these networks, the set $\{-1,+1\}^n$ is used rather than $\{0,1\}^n$. This is directly obtained by the following change of variable:
for $x \in \{0,1\}$, we note $x' = 2x-1$ their counterparts in $\{-1,+1\}$.
Then, we obtain

$$\begin{array}{rcl} x=1 & \Leftrightarrow & x'=+1 \\ x=0 & \Leftrightarrow & x'=-1 \end{array}$$

A program has been written which, starting from a set of vectors (patterns) to memorize and an initial vector X^0 (noisy pattern) verifies if the conditions of theorem 4.1 are verified and if this is the case, tells us the stable state where the convergence is achieved. Otherwise, the algorithm can not conclude about the convergence of the network initialized with X^0.

This algorithm is complementary to trivial simulation since in a lot of cases the convergence can be determined in a few steps and not all the iterations are necessary, it is then faster. Moreover, a single execution gives us the convergence for an initial state which is not the case in a simulation where several tests must be performed to ensure the convergence. Finally, it also facilitates the study and testing of new convergence conditions. This program is described in [1].

As a consequence of theorem 4.1, we deduce the result of papers [2, 3]

Proposition 4.2 *Let a discrete-time discrete-state dynamic n-neuron network be partitioned into α block-neurons and denote by $F = (F_1, ..., F_\alpha)$ its activation function defined on a finite product set $E = \prod_{i=1}^{\alpha} E_i$. If F is a contraction on the finite Cartesian product set E and if the three conditions of definition 3.1. are satisfied, then any fully asynchronous dynamic from any initial state $X^0 = \left(X_1^0, ..., X_\alpha^0\right)$ converges to the unique stable state X^* within at most p_α steps, i.e.*

$$X^* = X^t, \;\; t = p_\alpha, p_\alpha + 1, ...$$

where X^t is defined in algorithm 2 and $\{p_l\}_{l\in\mathbb{N}}$, in definition 3.2.

The results for boolean networks obtained in [4, 7] are obtained as a particular result of theorem 4.1 by taking $E_i = \{0,1\}$ for all $i \in \{1, ..., n\}$ and by remarking that for all $l \in \{1, ..., q\}$, $V \subset W^l$ and that if the hypotheses in [4, 7] are satisfied then it is also the same for the hypotheses of theorem 4.1.

It should be noted that theorem 4.1. doesn't require any information neither on the "delays" $s_j^i(t)$, none on the "strategy" $J(t)$. The delays and the strategy may be generated randomly as long as conditions (c1), (c2) and (c3) are satisfied.

5 The main result in the Hopfield networks context

Hopfield networks are widely used in pattern recognition. Because of this direct link to applications, numerous authors have addressed the problem of the convergence of neural networks. These studies can be classified in two main groups : one whose states are real-valued and the other whose states are boolean values. Our result given in section 4.1 clearly includes the second case.
Most of the studies concerning the convergence of neural networks deal with the synchronous evolution. Hopfield [13] has shown the convergence to stable state for symmetric neural networks with nonnegative diagonal operating in serial mode. Golès et al. [19] have extended this result by showing the convergence to stable state or cycle of length 2 in a fully parallel mode. In their paper [26] Y. Shrivastava et al. gave necessary and sufficient conditions on the initial iterate for a class of symmetric parallel Hopfield networks to converge in one step and they applied their results to solve the vertex cover problem. The majority of the known results shows the existence of a Lyapunov function (or energy function) and prove that this function converges. Convergence of the energy function doesn't imply the convergence of the network to a fixed point. Other interesting studies have used different tools than the energy function to prove these results [16, 29, 24]

and provided additional results with different constraints like, for example, the convergence to a cycle of length 4 for antisymmetric networks in a fully parallel mode [17].
As mentioned in the introduction, fully parallel, sequential, block-sequential, desynchronized and asynchronous networks are particular cases of the networks considered in this paper. Nevertheless, the results of these papers cannot be directly derived from our result and their results are not valid in the case of fully asynchronous networks with overlapping. Our approach is not based on the energy function, it doesn't guarantee the global convergence of the network to fixed points. This is due to the fact that the networks we consider are fully asynchronous with overlapping updating. Then the convergence is unpredictable, even if we consider standard hypotheses which guarantee the convergence in some particular dynamics.
Our goal is not to construct a network only having stable states. We do not even try to find the different kinds of cycles a network will converge to (length 1, 2 or more). Here, we are interested in finding the initial states from which the network will converge to a given stable state whatever its dynamic. Also, we do not make any assumptions over the properties of the network and our result is applicable to any kind of networks beyond the range of Hopfield networks. For example, the W matrix representing the weights of the neurons can be arbitrary in the scope of our study.
Thus, our study takes place in a quite different context which is more general in the case of discrete time and discrete state networks, and may contribute to fill the gap existing in the case of fully asynchronous networks.

6 Simulation of a fully asynchronous Hopfield network

In order to confirm the results of theorem 4.1 and of the test procedure, a simulation program of an asynchronous Hopfield network has been written. This program, written in C++, allows to verify the actual behavior of such a network and avoids an actual implementation of the network on a parallel system. This approach is widely used in the domain of networks of automata and especially neural networks (see for example [15]).

To build the Hopfield network, the method given at the beginning of section 4.3 is used. Then, the network is initialized with a vector chosen by the user. The possible delays are randomly generated by probabilistic laws increasing with the delay. Hence, the greater is the delay of a component, the greater is its probability to be updated at the following iteration. This ensures that all the components are actually updated within a finite time. Finally, for a given initial vector, the simulation of the network is performed several times in order to obtain a statistical result of the convergence. If two simulations of a same network with a same initial vector give two different results, then there is a divergence. Also, the number of iterations is taken into account to detect the cycles. If the maximal number of iterations (given in theorem 4.1) is achieved and no convergence has been found, then it can be deduced that the network has reached a cycle.

In the following, a Hopfield network is built from a given set of vectors to *memorize* (see table 6). Then, with the vectors given in table 6 (noisy patterns), the convergence of the asynchronous network is tested using the test procedure. Afterwards, the execution of the network initialized with the same vectors is simulated by our second tool. Both the results of the test of convergence and the simulation are given in table 6.

Names	memorized patterns														
M0	1	-1	-1	1	-1	-1	-1	-1	1	-1	-1	1	-1	-1	1
	○	●	●	○	●	●	●	●	○	●	●	○	●	●	○
M1	1	-1	-1	-1	1	1	-1	-1	1	1	1	-1	1	1	-1
	○	●	●	●	○	○	●	●	○	○	○	●	○	○	●
M2	1	1	1	-1	-1	-1	-1	1	-1	1	-1	1	-1	1	-1
	○	○	○	●	●	●	●	○	●	○	●	○	●	○	●

Table 2: Set of the network memories ($\circ \equiv 1$ and $\bullet \equiv -1$).

N°	Tested vectors	Differences	Prediction	Simulation
1	○○●○●●●●○●●○●●●	2 (M0)	M0	M0
2	●●●○●●●●○●●○●●●	2 (M0)	M0	M0
3	○●○●○○●●○○○●○●●	2 (M1)	M1	M1
4	○●●●●○●●○○●●○●●	3 (M1)	M1	M1
5	○○○○●●●●●○●○●○●	2 (M2)	M2	M2
6	●●●●●●●●●●●●●●●	5, 7 and 8	None	M0
7	○○○○○○○○○○○○○○○	7, 8 and 10	None	Another fixed point
8	●●●●○●●●○●●●●●●	5, 6 and 9	None	divergence

Table 3: Test of the convergence and validation by simulation, for 8vectors ($\circ \equiv 1$ and $\bullet \equiv -1$).

Table 6 shows the results of prediction and their validation by simulation. The simulation has been executed 1000 times for each initial state. The column *Differences* shows the number of components which are different between the tested vector and the memorized pattern in brackets, or with each of the memories.

It can be seen that when the test reaches a convergence, it is confirmed by the simulation. The power of theorem 4.1 can be pointed out by the five first vectors of table 6. Their convergence can be deduced from another vector. In fact, when the convergence of a vector is predicted in the first step of the test procedure, its associated matrix M is a contraction including the considered vector. Then, M can give us some informations allowing to deduce other converging vectors in the first neighborhood of this vector. The new vectors depends on the null columns of the matrix M. For example, in table 6, the two first vectors are deduced from another one (closer to $M0$) which is [○●●○●●●●○●●○●●●]. The convergence of this vector is predicted in the first step of the algorithm. Thus, looking at the null columns of M allows us to build new vectors whose convergence is the same as this vector. In this example, all the columns of M are null, then it can be deduced that all its first neighborhood converges to $M0$. For our two first vectors, the two first columns of M have been chosen.

On the opposite, for some vectors such as the 6^{th}, nothing can be said by our theorem whereas there is convergence to one of the memories. This comes from the fact that our convergence conditions are sufficient but not necessary.

Moreover, for the 7^{th} vector, it can be noted that the convergence may exist but toward another fixed point which is not in the initial set of memories. This is not directly linked to theorem 4.1 but to Hopfield networks: Spurious states, which are unexpected mixings of the initial memories, are quite common in the Hopfield networks. This is one of the open problems of these networks, and no building method of the matrix W avoiding such false memories is known nowadays. Thus, the rules generally used for the construction of W often do a trade-off between the number of actually wanted memories and the number of false memories generated.

Finally, the 8^{th} vector points out the fact that for some initial vectors the asynchronous Hopfield network may not converge. For each execution of the network initialized by such a vector, the convergence depends on the random nature of the delays. Thus, there is some indeterminism in this case which leads to a *divergence*. This also confirms the good quality of our theorem since it does not predict anything for these difficult cases.

This remark about the divergence implies another one concerning the results of synchronous and asynchronous modes. Starting from a vector X^0, a network may converge in synchronous mode toward a fixed point X^* but not in asynchronous mode. Moreover, if we suppose that the network converges to X^* when initialized with X^0 in synchronous mode, then if there is convergence in asynchronous mode starting from X^0, it is necessary toward X^*. This comes from the fact that in our formulation, the synchronous mode is a particular case of asynchronism where all the delays are always null. This can be very interesting for further works to enhance the determination of the convergence.

Finally, it is interesting to see the impact of the condition (c2) of definition 3.1. This condition implies that for the updates of the components of the system, the versions of the components used follow in average the time of the system. For example, the delays may be randomly chosen in the range $[\frac{t}{2}, t]$ which satisfies

this condition. Nevertheless, it is not satisfied for the range $[0, t]$. Thus, it is not possible to regularly use the same version of a component. This is essential for the convergence since in the opposite case, the network could have cyclic or oscillating behaviors.

7 Conclusion

Sufficient conditions on the initial state which guarantee the convergence on a stable state of fully asynchronous networks with overlapping updating has been presented. The networks considered are arbitrary, they include, for example threshold networks, they may, for example, have cycles. Application to symmetric Hopfield neural networks was given. The convergence of such networks are known when their dynamic is fully parallel, block sequential, but not in the fully asynchronous case. Thanks to a test procedure which is an application of theorem 4.1, we gave for the first time initial states ensuring the convergence of the Hopfield network to an associative memory whatever its dynamic. That test procedure can be applied to arbitrary networks with given stable states, it is, for example, directly applicable to Hopfield networks with non symmetric matrix W.
Our approach is not based on the energy function but on the discrete derivative on the neighborhood of a fixed point. Definition 2.9 shows that it is very easy to compute this derivative. Proposition 2.3 shows that it is very easy to determine if its spectral radius is null.
Asynchronism with overlapping updating seems to be natural in massively neural networks, an open problem is to built a network which globally converges to stable states when its dynamic is fully asynchronous with overlapping updating, this is a hard problem since the convergence of such networks depends on the delays between the neurons.
In further works we will study necessary conditions for the convergence of fully asynchronous networks and we will characterize the cycles in such networks.

References

[1] J. Bahi and S. Contassot-Vivier, Stability of fully asynchronous discrete-time discrete-state dynamic networks, *IEEE Transactions on Neural Networks*, 13(6):1353-1363, 2002.

[2] J. Bahi and C. J. Michel, Simulations of asynchronous evolution of discrete systems, *Simulation Practice and Theory*, 7:309-324, 1999.

[3] J. Bahi and C. J. Michel, Convergence of discrete asynchronous iterations, *Intern. J. Computer Math.*, Vol 74, pp. 113-125, 2000.

[4] J. Bahi, Boolean totally asynchronous iterations, *International J. of Mathematical Algorithms*, Vol. 1, pp. 331-346, 2000.

[5] G. M. Baudet, Asynchronous iterative methods for multiprocessors, *J. ACM*, 25:226-244, 1978.

[6] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation*, Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[7] S. Contassot-Vivier, and J.M. Bahi, Convergence dans les systèmes booléens asynchrones et application aux réseaux de Hopfield, *Calculateurs Parallèles*, Hermes science publications, 2001.

[8] D. Chazan and W. L. Miranker, Chaotic relaxation, *Linear algebra Appl.*, 2:199-222, 1969.

[9] M. N. El Tarazi, Some convergence results for asynchronous algorithms, *Numer. Math*, 39:325-340, 1982.

[10] F. Robert, Théorème de Perron-Frobenius et Stein-Rosenberg booléens, *Linear Algebra and Its Applications*, Vol 19, pp. 237-250, 1978.

[11] F. Robert, Discrete Iterations, a metric study, Springer-Verlag Series in Computational Mathematics, Vol. 6, Berlin Heidelberg New-York, 1986.

[12] F. Robert, Les Systèmes Dynamiques Discrets, Mathématiques & Applications 19, Springer-Verlag, Berlin Heidelberg, 1995.

[13] J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proc. Nat. Acad. Sci., vol. 79, pp. 2554-2558, 1982.

[14] J.J. Hopfield , Neurons with graded response have collective computational properties like those of two-state neurons, *Proc. Nat. Acad. Sci.*, 81:3088-3092, 1984.

[15] A.J. Kane, and D.J. Evans, Neural network software simulation, *Intern. J. Computer Math.*, Vol. 71, pp 475-494, 1999.

[16] J. Bruck, and J.W. Goodman, A generalized convergence theorem for neural networks. *IEEE Trans. Inform. Theory*, 34:1089–1092, 1998.

[17] J. Bruck. On the convergence properties of the Hopfield model. *Proc. IEEE*, 78(10): 1579–1585, October 1990.

[18] A. Bhaya, E. Kaszkurewicz, and V.S. Kozyakin. Existence and stability of a unique equilibrium in continuous-valued discrete-time asynchronous Hopfield neural networks. *IEEE Trans. Neural Networks*, 7(3):620–628, May 1996.

[19] E. Golès, F. Fogelman-Soulie, and D. Pellegrin. Decreasing energy functions as a tool for studying threshold networks. *Disc. Appl. Math.*, 12:261–277, 1985.

[20] A.V.M. Herz and C.M. Marcus. Distributed dynamics in neural networks. *Physical Review E*, 47(3):2155–2161, 1993.

[21] V.S. Kozyakin, A. Bhaya, and E. Kaszkurewicz. A global asymptotic stability result for a class of totally asynchronous discrete nonlinear systems. *Mathematics of Control, Signals and Systems*, 12(2):143–166, 1999.

[22] P. Koiran. Dynamics of discrete-time, continuous-state Hopfield networks. *Neural Computation*, 6:459–468, 1994.

[23] J.-C. Miellou. Algorithmes de relaxation chaotique à retard. *RAIRO, R-1*, pp. 55-82, 1975.

[24] A.N. Michel, J.A. Farrell, and H.-F. Sun. Analysis and synthesis techniques for Hopfield type synchronous discrete time neural networks with application to associative memory. *IEEE Transact. Circuits Syst.*, 37(11):1356–1366, 1990.

[25] C.M. Marcus, and R.M. Westervelt. Dynamics of iterated-map neural networks. *Physical Review A*, 40(1):501–504, 1989.

[26] Y. Shrivastava, S. Dasgupta, and S.M. Reddy. Guaranteed convergence in a class of Hopfield networks. *IEEE Trans. Neural Networks*, 3(6):951–961, November 1992.

[27] M. Takeda, and J.W. Goodman. Neural networks for computation: Number representations and programming complexity. *Appl. Opt.*, vol. 25, no. 18, pp. 3033–3046, 1986.

[28] L.P. Wang. On the dynamics of discrete-time, continuous-state Hopfield neural networks. *IEEE Trans. Circuits and Systems-II: Analog and Digital Signal Processing*, 45(6):747–749, June 1998.

[29] X. Wang, A. Jagota, F. Botelho, and M. Garzon. Absence of cycles in symmetric neural networks. *Neural Computation*, 10:1235–1249, 1998.