Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Computers & Operations Research 32 (2005) 2179–2196

computers & operations research

www.elsevier.com/locate/dsw

A continuous Hopfield network equilibrium points algorithm

Pedro M. Talaván[a], Javier Yáñez[b,*]

[a]*Instituto Nacional de Estadística, Josefa Valcárcel 46, 28027 Madrid, Spain*
[b]*Department of Statistics and Operations Research, Faculty of Mathematics, Universidad Complutense de Madrid, Madrid 28040, Spain*

Abstract

The continuous Hopfield network (CHN) is a classical neural network model. It can be used to solve some classification and optimization problems in the sense that the equilibrium points of a differential equation system associated to the CHN is the solution to those problems. The Euler method is the most widespread algorithm to obtain these CHN equilibrium points, since it is the simplest and quickest method to simulate complex differential equation systems. However, this method is highly sensitive with respect to initial conditions and it requires a lot of CPU time for medium or greater size CHN instances. In order to avoid these shortcomings, a new algorithm which obtains one equilibrium point for the CHN is introduced in this paper. It is a variable time-step method with the property that the convergence time is shortened; moreover, its robustness with respect to initial conditions will be proven and some computational experiences will be shown in order to compare it with the Euler method.
© 2004 Elsevier Ltd. All rights reserved.

Keywords: Neural networks; Continuous Hopfield network

1. Introduction

The *continuous Hopfield network* (CHN) of size n is a fully connected neural network with n continuous valued units. Following the notation of Aiyer [1], let $T_{i,j}$ be the strength of the connection from neuron j to neuron i; each neuron i has also an offset bias of i_i^{b}; let u_i and v_i be the current state and the output of the unit i $\forall i \in \{1, \ldots, n\}$.

* Corresponding author. Tel.: +34-91-394-4522; fax: +34-91-394-4606.
E-mail addresses: ptalavan@ine.es (P.M. Talaván), jayage@mat.ucm.es (J. Yáñez).

0305-0548/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2004.02.008

Consequently, $\mathbf{u}, \mathbf{v}, \mathbf{i}^{\mathrm{b}}$ will be the vectors of neuron states, outputs and biases of the network. The dynamics of the CHN is described by the differential equation system

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = -\frac{\mathbf{u}}{\tau} + \mathbf{T}\mathbf{v} + \mathbf{i}^{\mathrm{b}} \tag{1}$$

and the output function $v_i = g(u_i)$ is a hyperbolic tangent

$$g(u_i) = \frac{1}{2}\left(1 + \tanh\left(\frac{u_i}{u_0}\right)\right) \quad u_0 > 0. \tag{2}$$

If, for an input vector $\mathbf{u}^0$, a point $\mathbf{u}^{\mathrm{e}}$ exists such that $\mathbf{u}(t) = \mathbf{u}^{\mathrm{e}}\ \forall t \geqslant t_{\mathrm{e}}$, for some $t_{\mathrm{e}} \geqslant 0$, this point is called an equilibrium point defined by differential equation system 1. The existence of equilibrium points for the CHN is guaranteed if a Lyapunov or Energy function exists. Hopfield [2] showed that, if matrix $\mathbf{T}$ is symmetric, then the following Lyapunov function exists:

$$E = -\frac{1}{2}\mathbf{v}^{\mathrm{t}}\mathbf{T}\mathbf{v} - (\mathbf{i}^{\mathrm{b}})^{\mathrm{t}}\mathbf{v} + \frac{1}{\tau}\sum_{i=1}^{n}\int_0^{v_i} g^{-1}(x)\,\mathrm{d}x. \tag{3}$$

The CHN will solve those classification and optimization problems which can be expressed as the constrained minimization of

$$E = -\tfrac{1}{2}\mathbf{v}^{\mathrm{t}}\mathbf{T}\mathbf{v} - (\mathbf{i}^{\mathrm{b}})^{\mathrm{t}}\mathbf{v}, \tag{4}$$

which has its minimum at the corners of the n-dimensional hypercube $[0,1]^n$, denoted as the Hamming hypercube. See, for instance, Ghosh [3], Nasrabadi [4], Wasserman [5] and Wu [6].

The idea is that the network's Lyapunov function, when $\tau \to \infty$, is associated with the cost function to be minimized in the problem. If the value of τ is large, the continuous system performs much like a discrete binary system, ultimately stabilizing with all outputs near to zero or one, i.e. the CHN converges to a corner of the Hamming hypercube; for further details, see Wassermann [5].

For example, Hopfield and Tank [7] proposed the following CHN in order to solve the N-cities traveling salesman problem (TSP): there are $n = N \times N$ neurons, whose output will be denoted as $v_{x,i}$, where $x \in \{1, \ldots, N\}$ is the city index and $i \in \{1, \ldots, N\}$ is the visit order. The matrix $V \in H$ will represent the system state of the CHN.

Let

$$H = \{V \in [0,1]^{N\times N}\}, \quad H_{\mathrm{C}} = \{V \in \{0,1\}^{N\times N}\}$$

be, respectively, the hypercube of $\mathbb{R}^n$, where $n = N \times N$, and its corner set.

A tour for the TSP will be characterized by the set

$$H_{\mathrm{T}} = \left\{ V \in H_{\mathrm{C}} \Big/ \sum_i v_{x,i} = 1,\ \sum_x v_{x,i} = 1\ \forall x, i \in \{1, \ldots, N\} \right\}.$$

Given any state $V \in H$, the energy function proposed by Hopfield for the TSP is

$$E(V) = \frac{A}{2}\sum_x \sum_i \sum_{j\neq i} v_{x,i}v_{x,j} + \frac{B}{2}\sum_i \sum_x \sum_{y\neq x} v_{x,i}v_{y,i} + \frac{C}{2}\left(\sum_x \sum_i v_{x,i} - N'\right)^2$$
$$+ \frac{D}{2}\sum_x \sum_{y\neq x} \sum_i d_{x,y}v_{x,i}(v_{y,i+1} + v_{y,i-1}) \tag{5}$$

(the $i+1$ and $i-1$ subscripts are given modulo N).

Given an arbitrary set of positive parameters A, B, C, D and $N' > N$, the three first terms of $E(V)$ will be dropped for any tour for the TSP ($V \in H_T$) and there only remains the fourth term, which is proportional to the length of this tour.

Developing the four terms of this energy function and comparing them with Eq. (4), we have

$$E(V) = -\frac{1}{2}\sum_{x,i}\sum_{y,j} v_{x,i}T_{xi,yj}v_{y,j} - \sum_{x,i} i^{\mathrm{b}}_{x,i}\, v_{x,i}, \tag{6}$$

where the weights and thresholds for the CHN are

$$T_{xi,yj} = -[A\delta_{x,y}(1-\delta_{i,j}) + B(1-\delta_{x,y})\delta_{i,j} + C + D(\delta_{i,j-1} + \delta_{i,j+1})d_{x,y}], \tag{7}$$

$$i^{\mathrm{b}}_{x,i} = CN' \tag{8}$$

($\delta_{x,y} = 1$ if $x = y$ and is equal to 0 otherwise).

The TSP is so stated as a parameter setting problem for the A, B, C, D and N' appropriate values for a given TSP instance. Afterwards, the network evolves until it is trapped in some equilibrium point. A set of analytical conditions of the CHN parameters was obtained by Talaván and Yáñez [8] so that any equilibrium point of the CHN characterizes a tour—indeed, a local optimum—for the TSP.

In general, after an appropriate parameter setting procedure—see Talaván [9]—some other combinatorial optimization problems can be solved through the computation of some equilibrium point for the parameterized differential equation system 1.

This differential equation system is very difficult to solve analytically, thereby being solved by numerical approaches. However, the CHN is very hard to simulate, since the model has a differential equation for each neuron. For this reason, a simple and quick method, as Euler's, is necessary.

This paper introduces a new algorithm to obtain a CHN equilibrium point. This algorithm computes an optimal time-step in each iteration so that a rapid convergence to an equilibrium point is provided; in this way, it is more efficient than the Euler's method. Moreover, this method is robust with respect to the initial conditions, and although the non-increasing energy function is guaranteed, it can be easily modified so that the problem of trapping in local minima is avoided jumping off to a different atractor basin.

The use of the Euler algorithm is discussed in Section 2 so that the proposed algorithm is justified. The variable integration step computation of this new algorithm is analyzed in Section 3. The algorithm must deal specifically with the borders of the Hamming hypercube and its neighborhoods. This task is performed in Section 4. The algorithm robustness is verified in Section 5, in this sense, the independence of the algorithm with respect to the gain parameter u_0 is proven. From these results, the algorithm which obtains an equilibrium point of the CHN is detailed in Section 6. Finally, some computational experiences are shown in Section 7.

2. Euler method discussion

As was pointed out above, the CHN is associated to the solution of the *Cauchy problem*

$$\frac{\mathrm{d}u_i(t)}{\mathrm{d}t} = \sum_{j=1}^{n} T_{i,j} v_j(t) + i_i^{\mathrm{b}} \quad \forall i \in \{1,\ldots,n\},$$

$$u_i(0) = u_i^0 \quad \forall i \in \{1,\ldots,n\}, \tag{9}$$

where $\mathbf{T}$ and $\mathbf{i}^{\mathrm{b}}$ are CHN (known) weight matrix and input bias vector, respectively, and, u_i and v_i represent, respectively, the current state and the output of the neuron i

$$v_i(t) = \frac{1}{2}\left(1 + \tanh\left(\frac{u_i(t)}{u_0}\right)\right) \quad \forall i \in \{1,\ldots,n\},$$

$$v_i^0 = \frac{1}{2}\left(1 + \tanh\left(\frac{u_i^0}{u_0}\right)\right) \quad \forall i \in \{1,\ldots,n\}, \tag{10}$$

where $u_0 > 0$ is the gain parameter.

Following the Euler method, the above differential equation system is discretized, see Demidowitsch [10] for further details, from a given Δt

$$u_i(t+\Delta t) = u_i(t) + \left(\sum_{j=1}^{n} T_{i,j} v_j(t) + i_i^{\mathrm{b}}\right)\Delta t \quad \forall i \in \{1,\ldots,n\},$$

$$v_i(t+\Delta t) = \frac{1}{2}\left(1 + \tanh\left(\frac{u_i(t+\Delta t)}{u_0}\right)\right) \quad \forall i \in \{1,\ldots,n\} \quad u_0 > 0.$$

This method has a local truncating error of magnitude $O(\Delta t^2)$ in each step, which is relatively large, thereby requiring the use of a very small step size to the detriment of a lower speed and a more rounded error in the floating point computations.

Moreover, it is sensitive to scale modification, that is to say, if CHN weights $T_{i,j}$ and input bias i_i^{b} are scaled

$$T'_{i,j} = KT_{i,j} \quad \text{and} \quad i_i^{\mathrm{b}'} = Ki_i^{\mathrm{b}}$$

then, the scaled energy function becomes $E'(\mathbf{v}) = KE(\mathbf{v})$ and it has a surface similar to original energy function, except for slopes which are more sharp; however, taking into account that

$$u'_i(t+\Delta t) = u'_i(t) + \left(\sum_{j=1}^{n} T'_{i,j} v_j(t) + i_i^{\mathrm{b}'}\right)\Delta t$$

$$= u_i'(t) + \left(\sum_{j=1}^{n} T_{i,j} v_j(t) + i_i^{\mathrm{b}}\right) K\Delta t \quad \forall i \in \{1,\dots,n\},$$

the step size Δt would be scaled by $\Delta t/K$ so that the new trajectory will be the same as the original.

The same effect has a scale modification of the gain parameter u_0. In effect, let

$$u_0' = Ku_0$$

be the new gain parameter, then

$$v_i'(t+\Delta t) = \frac{1}{2}\left(1 + \tanh\left(\frac{u_i(t) + \left(\sum_{j=1}^{n} T_{i,j} v_j(t) + i_i^{\mathrm{b}}\right)\Delta t}{Ku_0}\right)\right)$$

and computing $u_i(t) = K(u_0/2)\ln(v_i(t)/(1-v_i(t)))$ as the inverse function in Eq. (10), it is deduced

$$v_i'(t+\Delta t) = \frac{1}{2}\left(1 + \tanh\left(\frac{K(u_0/2)\ln(v_i(t)/1 - v_i(t)) + \left(\sum_{j=1}^{n} T_{i,j} v_j(t) + i_i^{\mathrm{b}}\right)\Delta t}{Ku_0}\right)\right)$$

$$= \frac{1}{2}\left(1 + \tanh\left(\frac{(u_0/2)\ln(v_i(t)/(1 - v_i(t))) + \left(\sum_{j=1}^{n} T_{i,j} v_j(t) + i_i^{\mathrm{b}}\right)\Delta t/K}{u_0}\right)\right).$$

In this way, the CHN size, the magnitudes of weights, biases and the gain parameter of the CHN must be considered to assign the size step Δt in Euler method.

In spite of all these objections, some authors, from their computational experiences, have proposed a step-size value of 10^{-5} or 10^{-6}; but with these values, a lot of CPU time is needed to obtain a CHN equilibrium point. For example, Bagherzadeh et al., see [11], in order to compute an equilibrium point of a CHN with $n = 10,000$ spent more than 3 h computing with a one-processor computer and 20 min with an eight-processors computer.

Hence, some authors propose to increase Δt to save computer time. For example, Brandt et al. [12] consider as an appropriate value $\Delta t = 0.005$, Wu [6] works with $\Delta t = 0.0001$ and Abe [13] uses $\Delta t \approx 0.01$ in his computational experiences. But, with these values, the trajectories can show erratic trajectories and jump-off to other basin attractions. This last property, however, can be used to improve solution quality by escaping from local minima, see Wang and Smith [14].

Taking into account the increasing size of the CHN differential equation system, the Euler method —and also the Runge–Kutta method—perform this task very slowly. This drawback can be explained by the *complete trajectory* simulation of these methods.

Indeed, in the CHN context, the objective is not the computation of the trajectory followed by the solution to the Cauchy problem 9, but the final point attained, the equilibrium point. This will be the objective of the proposed algorithm introduced in the next sections.

This algorithm has a variable integration step Δt along the iterations, depending on the energy function shape and guaranteeing a fast descent of this function.

3. The integration step

The dynamical system 9 can also be expressed as

$$\frac{\mathrm{d}v_i(t)}{\mathrm{d}t} = \frac{2}{u_0} v_i(t)(1 - v_i(t)) \frac{\mathrm{d}u_i(t)}{\mathrm{d}t} \quad \forall i \in \{1, \dots, n\},$$

$$v_i(0) = v_i^0 \in (0, 1) \quad \forall i \in \{1, \dots, n\}, \tag{11}$$

where

$$\frac{\mathrm{d}u_i(t)}{\mathrm{d}t} = \sum_{j=1}^{n} T_{i,j} v_j(t) + i_i^{\mathrm{b}}.$$

In this way, the state variable will be the output neuron set, computing from this set the input neuron set describing the potential of the neurons.

Since the energy function is non-increasing, given the instant t_0, the step size Δt will be chosen in such a way that the energy function will decrease for any $t \in [t_0,\ t_0 + \Delta t]$.

The neuron states and their energy function are computed in the following way:

$$v_i(t + \Delta t) = v_i(t) + \frac{\mathrm{d}v_i(t)}{\mathrm{d}t} \Delta t \quad \forall i \in \{1, \dots, n\},$$

$$E(t + \Delta t) = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} v_i(t + \Delta t) T_{i,j} v_j(t + \Delta t) - \sum_{i=1}^{n} v_i(t + \Delta t) i_i^{\mathrm{b}}.$$

Taking into account that $\mathbf{T}$ is a symmetric matrix,

$$E(t + \Delta t) = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} v_i(t) T_{i,j} v_j(t) - \Delta t \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\mathrm{d}v_i(t)}{\mathrm{d}t} T_{i,j} v_j(t)$$
$$-\Delta t^2 \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\mathrm{d}v_i(t)}{\mathrm{d}t} T_{i,j} \frac{\mathrm{d}v_j(t)}{\mathrm{d}t} - \sum_{i=1}^{n} v_i(t) i_i^{\mathrm{b}} - \Delta t \sum_{i=1}^{n} \frac{\mathrm{d}v_i(t)}{\mathrm{d}t} i_i^{\mathrm{b}}.$$

If the variables S_1 and S_2 are introduced

$$S_1 \equiv \sum_{i=1}^{n} \frac{\mathrm{d}v_i(t)}{\mathrm{d}t} \left(\sum_{j=1}^{n} T_{i,j} v_j(t) + i_i^{\mathrm{b}} \right),$$

$$S_2 \equiv -\sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\mathrm{d}v_i(t)}{\mathrm{d}t} T_{i,j} \frac{\mathrm{d}v_j(t)}{\mathrm{d}t},$$

the energy function is a parabolic function with respect to Δt

$$E(t + \Delta t) = E(t) - S_1 \Delta t + \tfrac{1}{2} S_2 \Delta t^2.$$

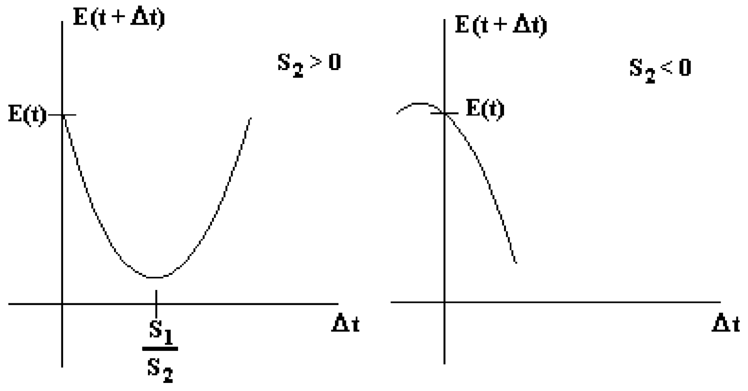

Fig. 1. Energy function versus Δt.

The term S_1 is always non-negative

$$S_1 = \sum_{i=1}^{n} \frac{\mathrm{d}v_i(t)}{\mathrm{d}t} \left(\sum_{j=1}^{n} T_{i,j} v_j(t) + i_i^{\mathrm{b}} \right) = \sum_{i=1}^{n} \frac{\mathrm{d}v_i(t)}{\mathrm{d}t} \frac{\mathrm{d}u_i(t)}{\mathrm{d}t}$$

$$= \sum_{i=1}^{n} \frac{2}{u_0} v_i(t)(1 - v_i(t)) \frac{\mathrm{d}u_i(t)}{\mathrm{d}t} \frac{\mathrm{d}u_i(t)}{\mathrm{d}t} \geqslant 0$$

because $0 < v_i(t) < 1$ and $u_0 > 0$.

The last expression takes the value 0 only for the stable points. Indeed, for a small enough value of $\Delta t > 0$, and independently of S_2, it is always verified that

$$E(t + \Delta > t) < E(t).$$

The appropriate value of Δt will depend on the term S_2: for the case $S_2 \leqslant 0$, the energy function will decrease for every $\Delta t > 0$ and, the greater value of Δt must be chosen; otherwise, if $S_2 > 0$, then the value

$$\Delta t = \frac{S_1}{S_2}$$

will produce the largest decrease of the energy function. See Fig. 1.

Remark 3.1. A jumping off to a different atractor basin can be obtained if

$$\Delta t = k \frac{S_1}{S_2}$$

is computed by a $k \geqslant 2$.

However, the integration step size must consider that the neuron states belong to the Hamming hypercube, i.e.

$$v_i(t + \Delta t) \in [0, 1] \quad \forall i \in \{1, \ldots, n\}.$$

In this way, the following term is introduced:

$$\Delta t^* = \max\left\{\delta \geqslant 0 / v_i(t) + \delta \frac{\mathrm{d}v_i(t)}{\mathrm{d}t} \in [0,1] \quad \forall i \text{ such that } v_i \in (0,1)\right\}. \tag{12}$$

Consequently, if the integration step Δt in instant t_0 is computed as

$$\Delta t = \begin{cases} \Delta t^* & \text{if } S_2 \leqslant 0, \\ \min\left\{\dfrac{S_1}{S_2}, \Delta t^*\right\} & \text{if } S_2 > 0, \end{cases} \tag{13}$$

then, the energy function decreases in the interval $[t_0, t_0 + \Delta t]$.

Finally, taking into account that large size steps can be obtained at first iterations, in order to avoid that the state variable jumps off an initial basin of attraction to other basins, in these iterations, the integration step size should be reduced by a factor q, with $0 < q < 1$:

$$\Delta t = q \cdot \Delta t.$$

4. Hamming border analysis

Because of the continuous nature of the dynamical system 9, the state variable $\mathbf{v}(t)$ always belongs to the interior of the Hamming hypercube $\mathrm{H}^\circ = \{V \in (0,1)^n\}$; therefore, $v_i(t)$ will never take a value 0 or 1 for all $i \in \{1, \ldots, n\}$.

However, due to round-off errors, the simulated trajectory can intersect this border set. From a theoretical point of view, there is no problem in such approximation, since the energy function is continuous and differentiable in H. But, from a computational point of view, four special treatments are needed:

- If, at some instant t, $\mathbf{v}(t)$ belongs to the neighborhood of the Hamming border, the next value of Δt^*—computed by Eq. (12)—could be very small or negative (due to round-off errors), and the state variable could not evolve adequately or it could evolve slowly.
  In order to avoid this problem, those $\mathbf{v}(t)$ in the neighborhood of the Hamming border must be rounded up. With this idea, let $\varepsilon > 0$ be a small parameter which controls the approximation scheme

$$v_i(t + \Delta t) = \begin{cases} 1 & \text{if } v_i(t) + \Delta t \dfrac{\mathrm{d}v_i}{\mathrm{d}t} > 1 - \varepsilon, \\ 0 & \text{if } v_i(t) + \Delta t \dfrac{\mathrm{d}v_i}{\mathrm{d}t} < \varepsilon, \\ v_i(t) + \Delta t \dfrac{\mathrm{d}v_i}{\mathrm{d}t} & \text{otherwise.} \end{cases} \tag{14}$$

- If at some instant t and for some component $i \in \{1, \ldots, n\}$ any value $v_i(t)$ verifying

$$v_i(t) = 1 \text{ and } \frac{\mathrm{d}u_i(t)}{\mathrm{d}t} < 0 \quad \text{or} \quad v_i(t) = 0 \text{ and } \frac{\mathrm{d}u_i(t)}{\mathrm{d}t} > 0$$

will never change because

$$\frac{\mathrm{d}v_i(t)}{\mathrm{d}t} = \frac{2}{u_0}\, v_i(t)\,(1 - v_i(t))\,\frac{\mathrm{d}u_i(t)}{\mathrm{d}t} = 0,$$

although such value is unstable in the dynamical system sense. One point $\mathbf{v}(t)$ verifying this property is denoted as unstable border point.

This problem can be solved by computing the new state variables through the potential variables

$$u_i(t+\Delta t) = u_i(t) + \Delta t\,\frac{\mathrm{d}u_i}{\mathrm{d}t}$$
$$v_i(t+\Delta t) = \frac{1}{2}\left(1 + \tanh\left(\frac{u_i(t+\Delta t)}{u_0}\right)\right).$$

- For such unstable border points, the stopping rule must be modified because larger variations of the potential variables could be associated to smaller variations of the state variables and, consequently, the stopping rule

$$\max_{i\in\{1,\dots,n\}} \{|v_i(t) - v_i(t+\Delta t)|\} \leqslant \varepsilon$$

is verified and the process finishes. This situation can also occur when the CHN model is simulated under classical approaches.

This revised stopping rule is the following:

$$\begin{aligned} &\max_{i\in\{1,\dots,n\}} \{|v_i(t) - v_i(t+\Delta t)|\} < \varepsilon^{1.5} && \text{if } \mathbf{v}(t) \text{ is an unstable border point,} \\ &\max_{i\in\{1,\dots,n\}} \{|v_i(t) - v_i(t+\Delta t)|\} < \varepsilon && \text{otherwise.} \end{aligned} \tag{15}$$

- The time-step Δt cannot be computed by Eq. (13) for the Hamming hypercube corner points $\mathbf{v} \in H_\mathrm{C}$, because there does not exist any component i such that $v_i \in (0,1)$ and $S_1 = S_2 = 0$; in this way, the time-step size procedure must be modified.

This modification must guarantee a variation from $\mathbf{v}(t)$ to $\mathbf{v}(t+\Delta t)$; but, on the other hand, such variation must be bounded so that an increase of the energy function will be avoided. A trade-off could be the following:

$$\Delta t = \min\left\{\delta \geqslant 0 \Big/ \left|u_i(t) + \delta\cdot\frac{\mathrm{d}u_i(t)}{\mathrm{d}t}\right| = u^\varepsilon \quad \forall i \text{ such that } v_i \in \{0,1\} \text{ unstable}\right\},$$

where u^ε is the potential associated to the value state ε, i.e.,

$$u^\varepsilon \equiv \frac{u_0}{2}\ln\left(\frac{\varepsilon}{1-\varepsilon}\right).$$

Although this modification could increase in some situations the energy function, their magnitudes are very small because $\mathrm{d}v_i(t)/\mathrm{d}t$ is close to 0 when $\mathbf{v}(t)$ is a corner point.

5. Robustness properties

The output function $v_i = g(u_i)$ is a hyperbolic tangent defined in Eq. (2) which depends on the gain parameter $u_0 > 0$. Its influence on the system trajectory depends also on the value of the parameter τ.

The gain parameter u_0 relates the state and the potential variables: given the state value v_i, the associated potential value u_i increases with u_0

$$u_i(t) = g^{-1}(v_i(t)) = \frac{u_0}{2} \ln\left(\frac{v_i(t)}{1 - v_i(t)}\right).$$

Consequently, if $\tau \ll \infty$, the energy function

$$E = -\frac{1}{2}\,\mathbf{v}^{\mathrm{t}}\mathbf{T}\mathbf{v} - \mathbf{v}^{\mathrm{t}}\mathbf{i}^{\mathrm{b}} + \frac{1}{\tau}\sum_{i=1}^{n}\int_{0.5}^{v_i} g^{-1}(x)\,\mathrm{d}x$$

is moderated by the integral term which depends on the parameter u_0.

Moreover, the system trajectory depends on the shape of the gradient $\nabla\mathbf{v}$, whose components are

$$\frac{\mathrm{d}v_i(t)}{\mathrm{d}t} = \frac{2}{u_0} v_i(t)(1 - v_i(t))\frac{\mathrm{d}u_i(t)}{\mathrm{d}t} = \frac{2}{u_0} v_i(t)(1 - v_i(t))\left(\sum_j T_{i,j}v_j(t) + i_i^{\mathrm{b}} - \frac{u_i(t)}{\tau}\right)$$

and, therefore, the gradient direction depends also on the parameter u_0.

On the other hand, if $\tau \to \infty$, the energy function

$$E = -\frac{1}{2}\,\mathbf{v}^{\mathrm{t}}\mathbf{T}\mathbf{v} - \mathbf{v}^{\mathrm{t}}\mathbf{i}^{\mathrm{b}}$$

only depends on the state variable $\mathbf{v}$ while it does not depend on the potential variable $\mathbf{u}$.

Taking into account the expression of the gradient

$$\nabla\mathbf{v} = \frac{2}{u_0}\begin{pmatrix} v_1(1 - v_1)\left(\sum_j T_{1,j}v_j + i_1^{\mathrm{b}}\right) \\ \vdots \\ v_i(1 - v_i)\left(\sum_j T_{i,j}v_j + i_i^{\mathrm{b}}\right) \\ \vdots \\ v_n(1 - v_n)\left(\sum_j T_{n,j}v_j + i_n^{\mathrm{b}}\right) \end{pmatrix}.$$

It is concluded that the parameter u_0 has no influence on the direction of the trajectory, but it influences on $\|\nabla\mathbf{v}\|$, i.e., the step size along this direction.

One desirable property for the variable integration step procedure proposed here will be the independence of the state variable trajectory with respect to the gain parameter u_0.

Proposition 5.1. *The algorithm proposed here is independent of gain parameter u_0 when $\tau \to \infty$.*

Proof. In the limit case $\tau \to \infty$, it is verified that

$$\frac{\mathrm{d}u_i(t)}{\mathrm{d}t} = \sum_j T_{i,j} v_j(t) + i_i^{\mathrm{b}},$$

which depends on the state variable $\mathbf{v}$ and does not depend on the gain parameter u_0.

Therefore, the derivative of any state component with respect to the time

$$\frac{\mathrm{d}v_i(t)}{\mathrm{d}t} = \frac{2}{u_0} v_i(t)(1 - v_i(t)) \frac{\mathrm{d}u_i(t)}{\mathrm{d}t}$$

is inversely proportional to the gain parameter u_0.

Consequently, in each iteration, the new state variable

$$v(t + \Delta t) = v(t) + \Delta t \frac{\mathrm{d}v_i(t)}{\mathrm{d}t}$$

does not depend on u_0 if the integration step Δt is directly proportional to u_0.

The following values are allowed to the integration step:

- $\Delta t = S_1/S_2$ if $S_2 > 0$.
  In this case,

$$\Delta t = \frac{S_1}{S_2} = \frac{\sum_i (\mathrm{d}v_i(t)/\mathrm{d}t)\, \mathrm{d}u_i(t)/\mathrm{d}t}{-\sum_i \sum_j (\mathrm{d}v_i(t)/\mathrm{d}t)\, T_{i,j}\, \mathrm{d}v_j(t)/\mathrm{d}t}$$

$$= \frac{u_0}{2} \frac{\sum_i v_i(t)(1 - v_i(t))(\mathrm{d}u_i(t)/\mathrm{d}t)^2}{-\sum_i \sum_j v_i(t)(1 - v_i(t))(\mathrm{d}u_i(t)/\mathrm{d}t)\, T_{i,j} v_j(t)(1 - v_j(t))\, \mathrm{d}u_j(t)/\mathrm{d}t}.$$

- $\Delta t = \max\{\delta \geqslant 0 / v_i(t) + \delta\, \mathrm{d}v_i(t)/\mathrm{d}t \in [0, 1]\ \forall i \in \{1, \ldots, n\}\}$.
  Let i^0 be the index verifying $v_{i^0}(t) + \Delta t\, \mathrm{d}v_{i^0}(t)/\mathrm{d}t = 1$ (the other value 0 is analogously proven). Then,

$$\Delta t = \frac{1 - v_{i^0}(t)}{\mathrm{d}v_{i^0}(t)/\mathrm{d}t} = \frac{u_0}{2 v_{i^0}(t)\, \mathrm{d}u_{i^0}(t)/\mathrm{d}t}.$$

- $\Delta t = \min\{\delta \geqslant 0 / |u_i(t) + \delta\, \mathrm{d}u_i(t)/\mathrm{d}t| = u^{\varepsilon}\ \forall i \in \{1, \ldots, n\}\}$ with $\mathbf{v} \in H_{\mathrm{C}}$.
  Let i^0 be the index verifying $u_{i^0}(t) + \Delta t\, \mathrm{d}u_{i^0}(t)/\mathrm{d}t = u^{\varepsilon}$ (the other value $-u^{\varepsilon}$ is analogously). Then

$$\Delta t = \frac{u^{\varepsilon} - u_{i^0}(t)}{\mathrm{d}u_{i^0}(t)/\mathrm{d}t} = \frac{u_0}{2} \frac{\ln(\varepsilon/(1 - \varepsilon)) - \ln(v_{i^0}(t)/(1 - v_{i^0}(t)))}{\mathrm{d}u_{i^0}(t)/\mathrm{d}t}$$

For all cases, Δt is directly proportional to u_0. □

Remark 5.1. With the variable integration step introduced in this paper, it can be analogously proven that the state variable trajectory does not depend on scale changes of the CHN weights.

6. The algorithm

The scheme of the algorithm which obtains an equilibrium point for the CHN is depicted in Fig. 2. In spite of the theoretical results of Section 5, the gain parameter u_0 will be included in the mathematical expressions in order to verify this property computationally.

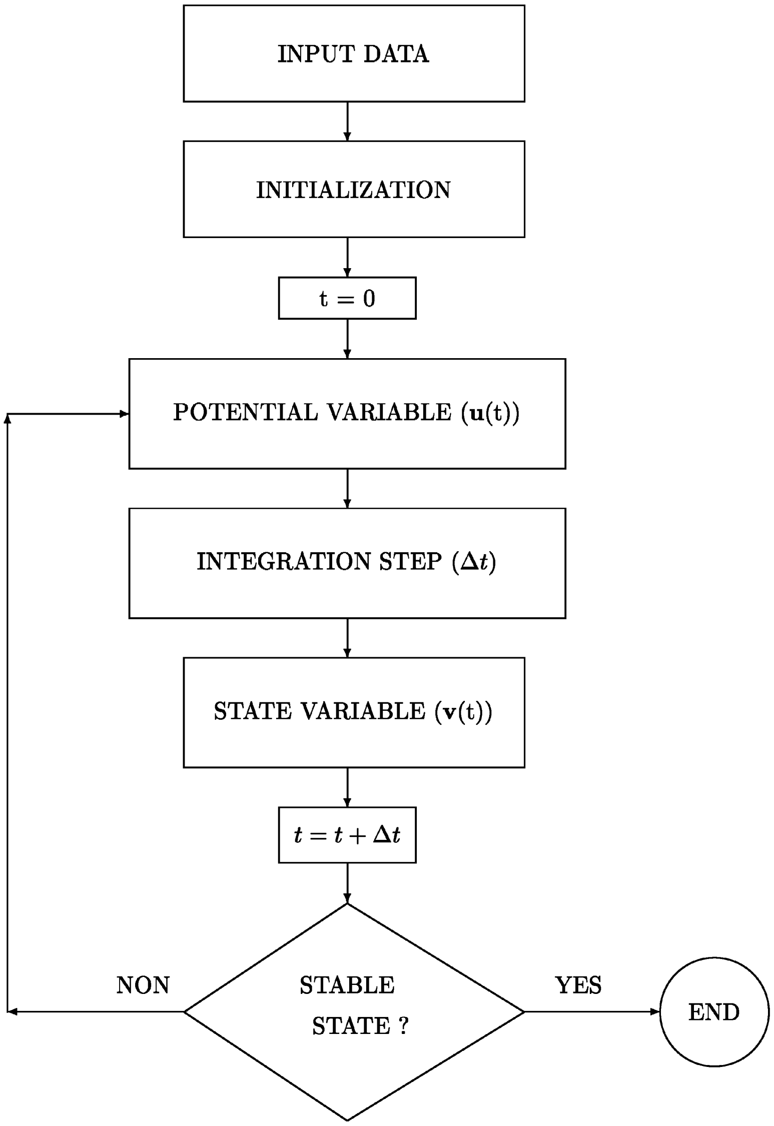

Fig. 2. Control scheme of the algorithm.

A new parameter *R_ITER* is introduced, which represents the number of first iterations for which the integration step is reduced by factor q.

Remark 6.1. Although a synchronous updating procedure is used, it is not difficult to modify the algorithm in order to implement an asynchronous updating one: the time step would be computed from the chosen neuron. The asynchronous updating procedure would be more simple but the overall CPU-time will be increased.

```
Enter: $n$, $\mathbf{T}$, $\mathbf{i}^b$                                  \* input data for the $CHN$ *\
Enter: $u_0$, $\varepsilon$, $q$, $R\_ITER$                            \* initialization parameters *\
Compute $\mathbf{v}^0$,   $u^\varepsilon = \frac{u_0}{2}\ln\left(\frac{\varepsilon}{1-\varepsilon}\right)$
$t = 0$, $iter = 0$
do                                                        \* $CHN$ simulation *\
    $\Delta t = 10^{100}$                                          \* initial value *\
    do  $i = 1, \ldots, n$
        $\frac{du_i(t)}{dt} = \sum_j T_{i,j} v_j(t) + i_i^b$                  \* potential variation *\
        if ($v_i(t) \in \{0,1\}$) then                  \* $\Delta t$ computation (border value) *\
            if (($v_i(t) = 0$ and $\frac{du_i(t)}{dt} > 0$) or ($v_i(t) = 1$ and $\frac{du_i(t)}{dt} < 0$)) then \* unstable *\
                $\Delta t = \min\left\{\Delta t, \frac{|u_i(t)| - u^\varepsilon}{\left|\frac{du_i(t)}{dt}\right|}\right\}$
        else                                   \* $\Delta t$ computation (interior value) *\
            $\frac{dv_i(t)}{dt} = \frac{2}{u_0} v_i(t)(1 - v_i(t)) \frac{du_i(t)}{dt}$
            if ($\frac{dv_i(t)}{dt} < 0$) then      $\Delta t = \min\left\{\Delta t, \frac{-v_i(t)}{\frac{dv_i(t)}{dt}}\right\}$
            if ($\frac{dv_i(t)}{dt} > 0$) then      $\Delta t = \min\left\{\Delta t, \frac{1 - v_i(t)}{\frac{dv_i(t)}{dt}}\right\}$
        endif
    enddo
    $S_1 = \sum_i \frac{dv_i(t)}{dt}\frac{du_i(t)}{dt}$        $S_2 = -\sum_i \sum_j \frac{dv_i(t)}{dt} T_{i,j} \frac{dv_j(t)}{dt}$
    $sw\_optimal = 0$
    if ($S_2 > 0$) then                                   \* optimal $\Delta t$ computation *\
        if $\Delta t < \frac{S_1}{S_2}$ then
            $\Delta t = \frac{S_1}{S_2}$
            $sw\_optimal = 1$
        endif
    endif
    if ($iter < R\_ITER$ and $sw\_optimal = 0$) then
        $\Delta t = q\ \Delta t$                                  \* reduction integration step *\
    endif
...
```

```
...
  do  $i = 1, \ldots, n$                                          \* states update *\
    if $(v_i(t) \in \{0,1\})$ then                                 \* border value *\
      $u_i(t+\Delta t) = u_i(t) + \frac{du_i(t)}{dt}\Delta t$
      $v_i(t+\Delta t) = \frac{1}{2}\left(1 + \tanh\left(\frac{u_i(t+\Delta t)}{u_0}\right)\right)$
    else                                                           \* interior value *\
      if $(v_i(t) + \frac{dv_i(t)}{dt}\Delta t \in [\varepsilon, 1-\varepsilon])$ then
        $v_i(t+\Delta t) = v_i(t) + \frac{dv_i(t)}{dt}\Delta t$
      else
        $u_i(t) = \frac{u_0}{2}\ln\left(\frac{v_i(t)}{1-v_i(t)}\right)$
        $u_i(t+\Delta t) = u_i(t) + \frac{du_i(t)}{dt}\Delta t$
        $v_i(t+\Delta t) = \begin{cases} 0 & \text{if} \quad v_i(t) + \frac{dv_i(t)}{dt}\Delta t < \varepsilon \\ 1 & \text{if} \quad v_i(t) + \frac{dv_i(t)}{dt}\Delta t > 1-\varepsilon \end{cases}$
      endif
    endif
  enddo
  $t = t + \Delta t$                                              \* time evolution *\
  $iter = iter + 1$
while $\begin{cases} \max_{i\in\{1,\ldots,n\}}\{|v_i(t) - v_i(t+\Delta t)|\} \geq \varepsilon^{1.5} & \text{border unstable case} \\ \max_{i\in\{1,\ldots,n\}}\{|v_i(t) - v_i(t+\Delta t)|\} \geq \varepsilon & \text{otherwise} \end{cases}$
```

Pseudocode *CHN equilibrium point*

7. Computational experiences

In order to test the algorithm introduced in this paper, some computer programs have been designed. These programs, which have been coded in *Borland C++Builder*3, compute one equilibrium point of the CHN by the Euler method and by the proposed algorithm. The programs were run in a compatible IBM PC, Pentium IV, 2533 MHz and 512 MB of RAM.

The theoretical properties of energy function decrease and convergence were verified through a first set of computational experiences based on 200 random CHN of size $n = 500$. The strength of the connections between these 500 neurons and the offset biases were randomly generated in the following way:

$$T_i = U[-10, 0], \quad i_i^{\mathrm{b}} = U[0, 5], \quad \forall i \in \{1, \ldots, n\},$$

where $U[a, b]$ is the uniform distribution in the interval $[a, b]$. The initial states were also randomly generated

$$v_i = U[0, 1] \quad \forall i \in \{1, \ldots, n\}.$$

For each one of these 200 CHN, the same results were obtained by the proposed algorithm setting the gain parameter $u_0 = 0.02$ and 20. In this way, Proposition 5.1 has been validated.

A second set of computational experiences was designed in order to compare the simulated trajectories by the Euler method and the proposed algorithm. The Euler method was run with a value $u_0 = 0.02$ and an integration step $\Delta t = 10^{-6}$ for the first 100 iterations and $\Delta t = 10^{-4}$ for the remainder. The proposed method was run with $R\_ITER = 0$. In both methods, the stopping rule 15

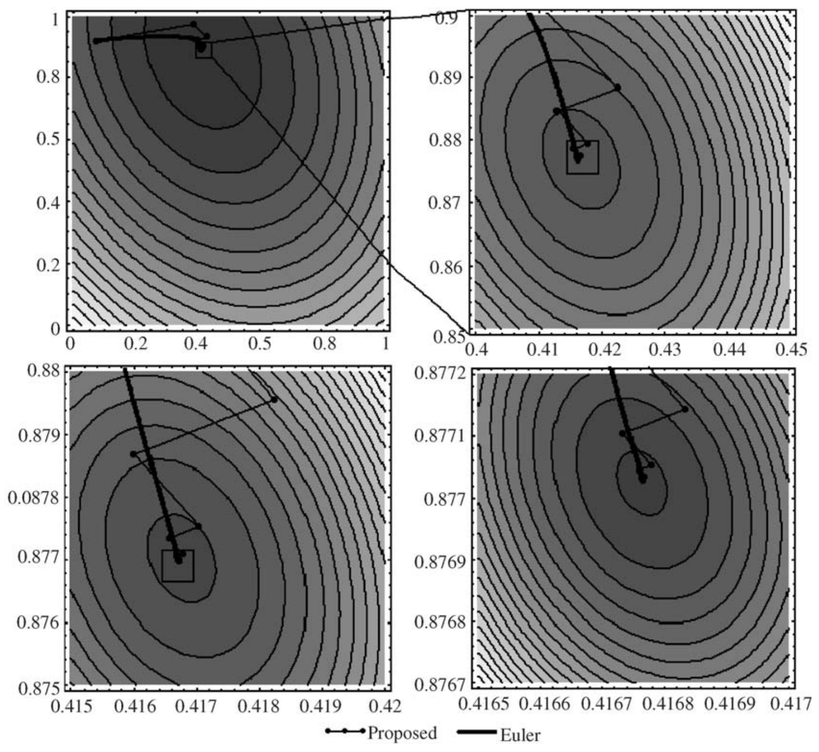

Fig. 3. Trajectories of the Euler and proposed methods for a two neurons CHN.

was used with $\varepsilon = 10^{-8}$. Fig. 3 shows, with different approximation levels, the trajectories of a two neurons CHN simulated by the Euler and the proposed method.

As can be seen from this figure, the Euler method evolves slowly so that its trajectory looks like a continuum line; however, the trajectory based on the proposed method shows large jumps between consecutive points around the above trajectory.

In this way, less iterations are needed so that some equilibrium point is attained. However, and in order to avoid jumping off to a different atractor basin, the reduction parameter q can be useful in the first iterations of the proposed method.

The Euler method shows a more accurate trajectory of the system variable only if the step integration Δt and the gain parameter u_0 have been properly chosen; otherwise, the Euler method shows erratic trajectories in the Hamming hypercube. On the other hand, the proposed algorithm is robust with respect to these initialization parameters.

This fact was verified by some additional computational experiences with CHN of size 500: changing the network parameters

$$T'_{i,j} = 20T_{i,j}, \quad i_i^{\mathrm{b}'} = 20\, i_i^{\mathrm{b}},$$

the Euler method never performed properly, but the same results were obtained with the proposed method, showing the robustness of this method with respect to the initializing parameters.

Table 1
Number of iterations and CPU time

	Average number of iterations with CHN size			Average CPU time with CHN size		
	100	500	1000	100	500	1000
Euler method	8451.37	10012.22	10744.73	1.29	35.9	148.33
P.A. (R_ITER = 0)	196.96	309.17	303.11	0.07	2.51	9.75
P.A. (R_ITER = 50)	242.67	337.14	380.89	0.09	2.78	12.11
P.A. (R_ITER = 100)	290.54	387.99	419.66	0.11	3.18	13.43
P.A. (R_ITER = 150)	335.79	422.17	446.58	0.11	3.41	14.24

Table 2
Relative performance of the proposed algorithm

	Relative CPU time with CHN size			Relative energy with CHN size		
	100	500	1000	100	500	1000
P.A. (R_ITER = 0)	0.05	0.07	0.07	1.03	1.06	1.06
P.A. (R_ITER = 50)	0.07	0.08	0.08	1.01	1.02	1.04
P.A. (R_ITER = 100)	0.08	0.09	0.09	1.00	1.01	1.00
P.A. (R_ITER = 150)	0.09	0.09	0.10	1.00	1.01	1.01

Finally, a third set of computational experiences was made in order to compare the efficiency of the proposed algorithm with respect to the Euler method.

Three variants of the proposed algorithm have been tested; they are identified by a fixed reduction integration step $q = 0.1$ and the bounds $R\_ITER = 50, 100, 150$ for the number of iterations for which this reduction is applied.

Three sizes $n = 100, 500, 1000$ were considered for the random CHN. For each one of these sizes, a sample of 200 CHN was generated using their random strength weights and random initial points.

From these samples, two output variables are obtained for the Euler method and the three variants of the proposed algorithm: the average number of iterations needed to obtain an equilibrium point and the average CPU time, in seconds; these statistics are depicted in Table 1.

Clearly, the proposed algorithm performs much better than the Euler method with respect to the number of iterations and, consequently, in the CPU time spent in obtaining an equilibrium point; indeed, it is about 12 times faster than the Euler algorithm.

However, it is necessary to compare the quality of the solution to select a CHN equilibrium points algorithm. This measure will be the energy function of the solution.

Because of the randomness of the sample, two different CHN of the same size can show very different energy function scales, the output variable will not be the absolute energy average values attained by the algorithms, but the average ratio of the energy value taken by the different variants of the proposed algorithm to the energy value taken by the Euler method. In the same way, the CPU-time will also be transformed.

In Table 2 are depicted these two output variables. The smaller the variables, the better the proposed algorithm.

The relative CPU-time 0.05 for the proposed algorithm with $R\_ITER = 0$ shows that the proposed algorithm is 20 times faster than the Euler algorithm for a CHN of size $n = 100$. It can be seen from the table that the CPU-time increases linearly with respect to the parameter R_ITER.

On the other hand, taking into account that a relative energy greater than 1.00 indicates a greater value for the proposed algorithm with respect to the Euler algorithm, the better performance for the proposed algorithm is associated with the greater values for the parameter R_ITER.

As a trade-off between these two opposite considerations, the parameter value $R\_ITER = 100$ seems to be the most appropriate, since the same energy values are obtained as those obtained by the Euler method, but with a drastic reduction of the CPU-time.

Moreover, these excellent computational properties of the proposed algorithm have been shown in some applications of this neural approach in the sense that very large neural networks can be handled. This is the case of the combinatorial optimization problem already mentioned of the TSP: the solution of instances with sizes of up to 1000 cities, implying neural networks up to one million nodes, have been obtained with the proposed algorithm, see Talaván and Yáñez [8].

8. Summary and conclusions

A new algorithm which obtains one equilibrium point for a CHN is introduced in this paper. This algorithm has been compared with the Euler method. Although the Euler method is better at simulating the overall trajectory of the CHN, the proposed method is faster, because it is designed as a procedure to obtain one equilibrium point for the CHN. Moreover, the proposed method is more robust than the Euler method with respect to the initialization parameters.

The equilibrium point of the appropriate CHN can be used to represent a solution of some classification and combinatorial optimization problems, and taking into account that the size of those neural networks increases with respect to the size of these problems, a very efficient computational procedure is needed. The proposed algorithm verifies all of these requirements.

As was pointed above, taking into account that the energy function is non-increasing, the proposed algorithm always converges to a local minimum, it works as a steepest descent method. Some future research is needed in order to avoid the problem of trapping in local minima; in this way, see Remark 3.1, this algorithm could be combined with some simulated annealing scheme.

Consequently, the proposed algorithm is a very useful computational procedure to solve some classification and combinatorial optimization problems through the neural network approach.

Acknowledgements

We thank the anonymous referee for his interesting suggestions. This research has been supported by DGICYT (National Grant number BFM2002-00281).

References

[1] Aiyer SVB, Niranjan M, Fallside F. A theoretical investigation into the performance of the Hopfield model. IEEE Transactions on Neural Networks 1990;1(2):204–15.

[2] Hopfield JJ. Neurons with graded response have collective computational properties like those of two-states neurons. Proceedings of the National Academy of Sciences USA 1984;81:3088–92.
[3] Ghosh A, Pal NR, Pal SK. Object background classification using Hopfield type neural networks. International Journal of Pattern Recognition and Artificial Intelligence 1992;6:989–1008.
[4] Nasrabadi NM, Choo CY. Hopfield network for stereo vision correspondence. IEEE Transactions on Neural Network 1992;3(1):5–13.
[5] Wasserman PD. Neural computing. Theory and practice. New York: Van Nostrand Reinhold; 1989.
[6] Wu JK. Neural networks and simulation methods. New York: Marcel Dekker; 1994.
[7] Hopfield JJ, Tank DW. Neural computation of decisions in optimization problems. Biological Cybernetics 1985;52: 1–25.
[8] Talaván PM, Yáñez J. Parameter setting of the Hopfield network applied to TSP. Neural Networks 2002;15(3): 363–73.
[9] Talaván PM. El modelo de Hopfield aplicado a problemas de optimización combinatoria. PhD dissertation. Universidad Complutense de Madrid, Spain; 2003.
[10] Demidowitsch BP, Maron IA, Schuwalowa ES. Métodos numéricos de Análisis. Madrid: Paraninfo; 1980.
[11] Bagherzadeh N, Kerola T, Leddy B, Brice R. On parallel execution of the traveling salesman problem on a neural network model. Proceedings of the IEEE International Conference on Neural Networks, San Diego, USA, vol. III, 1987. p. 317–24.
[12] Brandt RD, Wang Y, Laub AJ, Mitra SK. Alternative networks for solving the traveling salesman problem and the list-matching problem. Proceedings of the International Conference on Neural Network (ICNN'88), San Diego, USA, vol. II, 1988. p. 333–40.
[13] Abe S. Global convergence and suppression of spurious states of the Hopfield neural networks. IEEE Transactions on Circuits and Systems 1993;40(4):246–57.
[14] Wang L, Smith K. On chaotic simulated annealing. IEEE Transactions on Neural Network 1998;9(4):716–8.