
Annals of Mathematics and Artificial Intelligence **42:** 345–367, 2004.

A discrete-time quantized-state Hopfield neural network

Carlos Bousoño-Calzón and Sancho Salcedo-Sanz *

Department of Signal Theory and Communications, Universidad Carlos III de Madrid, Spain
E-mail: sancho@tsc.uc3m.es

A discrete-time quantized-state Hopfield neural network is analyzed with special emphasis in its convergence, complexity and scalability properties. This network can be considered as a generalization of the Hopfield neural network by Shrivastava et al. [27] into the interior of the unit hypercube. This extension allows its use in a larger set of combinatorial optimization problems and its properties make of it a good candidate to build hybrid algorithms along with other heuristics such as the evolutive algorithms. Finally, the network is illustrated in some instances of the linear assignment problem and the frequency assignment problem.

Keywords: quantized-state Hopfield neural network, stability, convergence, scalability, complexity

AMS subject classification: 68W40, 68Q17, 68Q25, 68R10

1. Introduction

Since the seminal papers by J.J. Hopfield et al. [6,15,16] Hopfield neural networks have been extensively applied to combinatorial optimization problems (see for instance [7,9] and [24]). Although the results are quite encouraging, some drawbacks have been perceived by the research community.

First, in spite of much work has been done to understand the stability of continuous neural networks and to select their parameters to obtain feasible solutions for optimizations problems [1,2,17,21], when a continuous system is simulated in a computer, the dynamics of the discretized model may be completely different to that of the continuous-time model.[1] The step size for the integration of the differential equations describing the continuous neural network, which is one of the most important parameters governing the dynamics of the discretized system, is usually selected by trial and error. Thus, no concluding results have been developed for the "simulated" systems which makes some important issues, such as the scalability of a neural network, difficult to control [25].

Second, the results obtained by these networks can be guaranteed to fulfil the constraints of a given optimization problem, but the neural networks are not so good at obtaining optimal solutions. This fact has fostered an interest in mixing these neural algorithms with other heuristics (for example genetic algorithms) to increase their performance [3,26,32].

* Corresponding author.

[1] For a rigorous discussion of this topic, see [11,30] and [10].

Along with the continuous-state Hopfield neural networks, the digital Hopfield networks, with neuron states in {0, 1}, have deserved an elaborated mathematical theory [29] which allows a better control of the aforementioned issues. In particular, Shrivastava et al. [27] have proposed a Hopfield neural network which provably converges to a solution in at most $2n$ steps, n being the number of neurons in the network. They show that there is a strong relation between Hopfield networks and the so called *vertex covering problem* (VCP), for which their neural algorithm has at least a comparable performance to classical algorithms. The VCP can be briefly stated as follows: Let $G = (V, E)$ be a graph[2] with a set of vertices V and a set of undirected edges E. A subset $X \subseteq V$ is called a *vertex covering* if every edge of G has at least one endpoint in X. Then the VCP is defined as finding a vertex cover X with minimum cardinality (i.e. size). In addition let $G' = (V', E')$ be an *induced subgraph* of G, that is, a graph where V' is a subset of V and E' is the subset of E consisting of those edges in E which have both endpoints in V'. We say that G' is *isolated* if $E' = \emptyset$. Additionally, if G' is isolated, it is a *Maximum Independent Set* (MIS) if every vertex in the difference set $V - V'$ is connected to at least one vertex in V'. The following relation between those vertices sets holds: $X = V - V'$ is an optimal solution to the VCP if and only if G' is the largest MIS of G.

This paper elaborates on an extension of Shrivastava's network from digital values {0, 1} to a finite number of values in [0,1]. This model is suggested by Matsuda [22, 23] which also demonstrates its convergence; however, it is also known that steepest descent networks may require exponential time [12]. Our analysis is centered in its complexity proving polynomial bounds in the number of states, and hopefully providing new insights into its dynamics which may allow a judicious design of hybrid algorithms.

In order to test the complexity properties of the quantized network in combinatorial optimization problems, we have selected the so-called "linear assignment problem" (LAP) [25]. This problem has well known solutions provided by traditional algorithms so the comparative performance of this neural network can be seized along with the study of its complexity. We have also applied the network to an instance of an NP-complete problem, the frequency assignment problem [9], in order to show how the complexity of our network is also bounded in this kind of problems.

The structure of the paper is the following: the quantized Hopfield neural network is described in section 2. In section 3, we discuss the convergence and scalability properties. We provide the illustration of its behavior solving some LAP and FAP instances in section 4.1. Finally, we close the paper by drawing some conclusions in section 5.

2. The neural network

The architecture of the Hopfield model is a graph with each node, x_i, representing a neuron for $i = 1, \ldots, n$, where n is the *order* of the network. Let $\mathbf{W}$ be the *adjacency matrix* for the graph consisting of symmetric components $w_{ij} = w_{ji} \in \{0, 1\}$. We

[2] For and introduction to graph concepts, see [14].

further assume that there are not self connections, that is $w_{ij} = 0$ if $i = j$. We will refer to this graph as *the neural network*.

For some $p = 1, 2, \ldots$, let $\Delta = 1/p$ and $x_i(t) \in S$ denote the state of the neuron x_i at iteration t, where $S = \{0, \Delta, 2\Delta, \ldots, p\Delta\}$. The vector $\mathbf{x}(t) = [x_1(t), \ldots, x_n(t)]^{\mathrm{T}}$ (where the superscript indicates transpose) is the state of the network at iteration t, with $t = 0, 1, 2, \ldots$. We will use "0" and "1" to denote the states where the neuron values are 0 and 1, respectively.

Each neuron is also associated a threshold; let $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_n]^{\mathrm{T}}$ where $\theta_i \in (0, 1)$ be the threshold vector: we will prove that the thresholds are forced to be in this interval in order to get feasible solutions for the MIS problem. Let us further assume that for every i, $\theta_i \notin S$, which facilitates the proof of the statements below.

The algorithm operates in serial iteration mode, which means that only one neuron is updated each iteration time while the rest are left unchanged. The change of state follows these rules:

R1. Select $\pi \in \wp$, where $\wp$ is the set of permutations of $\{1, \ldots, n\}$, i.e. π is a random ordering of the n neurons in the network.

R2. Select an initial state $\mathbf{x}(\mathbf{0}) \in S^n$.

R3. Update in a cyclical way the neurons, following ordering π: for each iteration $t = 0, 1, 2 \ldots$, neuron $i = \pi(k)$ (where $k = ((t \bmod n) + 1)$, and "mod" is the modulus function which gives the remainder after division by n) is updated as

$$x_i(t+1) = x_i(t) + \Delta \cdot \operatorname{sgn}\big(P_i(t)\big) = \begin{cases} x_i(t) + \Delta & \text{if } P_i(t) > 0, \\ x_i(t) - \Delta & \text{if } P_i(t) < 0 \end{cases} \tag{1}$$

here $P_i(t) = \theta_i - \sum_{j=1}^{n} w_{ij} x_j$, while the rest of the neurons are left unchanged.

R4. Finally, if as the result of R3 the state for x_i is outside the unit interval, it is corrected as follows:

$$x_i(t+1) = \begin{cases} 0 & \text{if } x_i(t+1) < 0 \text{ by R3}, \\ 1 & \text{if } x_i(t+1) > 1 \text{ by R3}. \end{cases} \tag{2}$$

In the rest of this section, we comment on the neural network definition.

Although it is possible to define matrix $\mathbf{W}$ with arbitrary positive values, we prefer to maintain its interpretation as the adjacency matrix for the neural network graph: this allows first to derive some results relating the structure of this graph and the convergence properties in next section, and second, to tackle the constraints of the optimization problem, essentially the MIS problem, while leaving the cost function for the optimization problem to be set by another algorithm in a hybrid system design, as expressed in the motivation of this work. The discretized space S for the neurons permits the coding of the cost functions in the initial state of the network as illustrated for the LAP in section 1.

The definition of a fixed step size, Δ, allows a controlled scalability of the network for such problems. With some constraints for the network parameters, we will prove

that the convergence time, measured as the number of complete cycles performed in the neural network, grows with the step size and not with n. To some degree, it is also possible to control the quality of the solutions obtained by the network; further improvements may be obtained by mixing the neural network with other global-search algorithms with a trade-off between the complexity of both parts.

The serial mode is selected because the convergence to a MIS is guaranteed; if rules R1 and R3 above are changed to a parallel mode iteration, in which all the neurons are modified at once, the attractor for the system may be a 2-cycle. To see this, proofs as those given in [29] or [5] can be followed, however, an alternative procedure will give some additional insight into the 2-cycles attractors:

Given a graph $G = (V, E)$ a *clique* is defined as a subset $V' \subset V$, such that for every $v_i, v_j \in V'$, there is an edge between them in E, we also consider the individual vertices as cliques [11]. If this concept is applied to the neural network, every pair of connected neurons is a clique.

Let us first assume a n-order neural network in which the whole set of neurons is a clique, obviously, any subset of neurons is also a clique in this case. We will prove that this neural network can oscillate in a 2-cycle.

Proposition 1. Given an n-order neural network in which the whole set of neurons is a clique, let k be a positive integer, such that $k < p$,

$$S_k = k\Delta \quad \text{and} \quad S_{k+1} = (k+1)\Delta, \tag{3}$$

$$\forall i, \quad \theta_i = \theta, \quad S_k < \frac{\theta}{n-1} < S_{k+1} \tag{4}$$

and

$$\mathbf{x}(0) = [S_k, S_k, \ldots, S_k]^{\mathrm{T}}; \tag{5}$$

then the neural network oscillates between the states S_k and S_{k+1}.

Proof. For $i = 1, \ldots, n$ and $t = 0$, $\mathbf{x}(0) = [S_k, S_k, \ldots, S_k]^{\mathrm{T}}$ and

$$P_i(0) = \theta_i - \sum_{j=1}^{n} w_{ij} x_j(0) = \theta - (n-1)k\Delta.$$

By (5), $P_i > 0$ so that

$$x_i(1) = k\Delta + \Delta = (k+1)\Delta = S_{k+1}$$

and

$$\mathbf{x}(1) = [S_{k+1}, S_{k+1}, \ldots, S_{k+1}]^{\mathrm{T}},$$

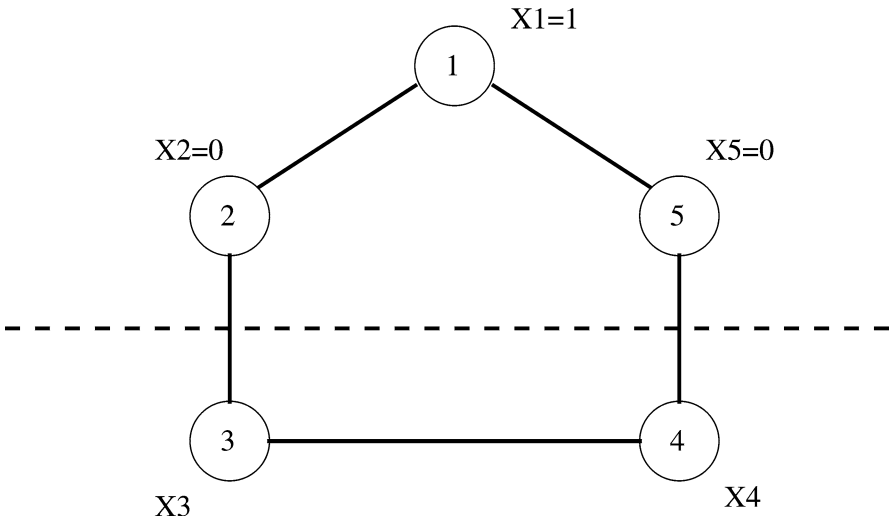


Figure 1. This neural network illustrates a possible oscillation of neurons x_3 and x_4 while x_1, x_2 and x_3 are in a stable state.

so that

$$P_i(1) = \theta_i - \sum_{j=1}^{n} w_{ij} x_j(1) = \theta - (n-1)(k+1)\Delta,$$

and by (5) $P_i < 0$, therefore

$$x_i(2) = (k+1)\Delta - \Delta = k\Delta = S_k$$

and

$$\mathbf{x}(2) = [S_k, S_k, \ldots, S_k]^{\mathrm{T}} = \mathbf{x}(0).$$

The proof is completed. □

However, there is a more interesting case when in a network some subgraph has converged, but there is still a clique oscillating. To give a simple example, consider the graph in figure 1: since the neurons x_2 and x_5 are in state 0, they do not affect to the clique consisting of $\{x_3, x_4\}$ to which proposition 1 can be applied with $n = 2$.

Finally, the definition of an order for the sequential update of the neurons, rule R1, is not necessary in order to prove some convergence properties, although it simplifies the process. It may be worth to point out that the explicit consideration of an ordering allows the formulation of alternative hybrid schemes in which the order of the cycle π, and not the initial state, is managed by the coupling algorithm, as usually done in literature.

3. Stability and convergence results

We prove in this section that the stable states for this network are those corresponding to the MIS problem and give some upper bounds on the number of iterations needed to obtain a solution for it.

We need the following definition: $\mathbf{x}(t)$ is a *stable state*[3] of the neural network if, for any positive integer $m \leqslant n$, $\mathbf{x}(t+m) = \mathbf{x}(t)$. Note that this definition of stable state assumes a cyclic updating of the neurons as defined in R3.

Lemma 2. Every stable state has to be a corner of the unit hypercube $\{0, 1\}^n$. Furthermore, if $x_i = 1$ then $P_i > 0$ and if $x_i = 0$, then $P_i < 0$ for a stable state.

Proof. In every cycle, all neurons are updated in turn and by the definition of the states and thresholds, P_i is always different from zero; so that for every neuron with a state different from "0" and "1" to be updated, the variation of the neuron is not null and not stationary. Therefore, a stationary state cannot be in the interior or faces of the unit-hypercube but only on its vertices.

Additionally, if $x_i = 1$ and $P_i < 0$ ($x_i = 0$ and $P_i > 0$), this value can decrease (increase), so it is not stationary. Therefore in order to be a stable state, if $x_i = 1$ then $P_i > 0$ and if $x_i = 0$, then $P_i < 0$. This completes the proof. □

Theorem 3. Stable states are MIS if and only if $\theta_i \in (0, 1)$ for every i.

Proof. First, suppose that $\theta_i \in (0, 1)$ and that $\mathbf{x}(t)$ is a stable state, thus, by lemma 2, all the neurons are in state "0" with $P_i < 0$ or in state "1" with $P_i > 0$. Let also V be the set of indices such that $x_i = 1$ if $i \in V$ and V' its complement. Additionally, let N_i denote the neighborhood of x_i, that is, the set of the neurons connected to x_i.

If $i \in V$, $P_i > 0$, then $\theta_i > \sum_{j \in N_i} x_j$ all the neurons in N_i must be in state "0", which means that V is isolated.

If $i \in V'$, $P_i < 0$, then $\theta_i < \sum_{j \in N_i} x_j$ and there is at least one $j \in N_i$ in state "1", which means that V is a MIS.

Second, suppose that the stable set is a MIS.

If $i \in V$, $P_i > 0 \Rightarrow \theta_i > \sum_{j \in N_i} x_j$, and since it is isolated, for every $j \in N_i$, $x_j = 0$ so that, $\theta_i > 0$.

If $i \in V'$, $P_i < 0 \Rightarrow \theta_i < \sum_{j \in N_i} x_j$, and since V is a MIS, there is one $j \in N_i$ such that $x_j = 1$ so that, $\theta_i < 1$.

This completes the proof. □

Note that for theorem 3 a fixed value for θ_i is supposed.

We give below some proofs about the dynamics of this network: proofs for theorems 4 and 6 follow [29] and [5], and essentially make use of a Lyapunov (or Energy) function for the system; we also prove theorem 8 following [27] (see also [29] and [22]).

[3] What we call *stable states* of the network are also known as *equilibrium points* in the literature.

Theorem 4. The quantized neural network in serial mode following R1–R4, with symmetric weights (see section 2), converges to a stable state.

Proof. (a) Let $E = 2\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta} - \mathbf{x}^{\mathrm{T}}\mathbf{W}\mathbf{x}$, we will prove that $E(t) \geqslant E(t+1)$, which makes E a Lyapunov function for the quantized neural network.

In order to simplify the notation, we will denote $E(t)$, $\mathbf{x}(t)$ and $x_i(t)$ by E, $\mathbf{x}$ and x_i, and $E(t+1)$, $\mathbf{x}(t+1)$ and $x_i(t+1)$ by E^+, $\mathbf{x}^+$ and x_i^+, respectively.

Let $\delta E = E - E^+$, $\delta\mathbf{x} = \mathbf{x} - \mathbf{x}^+$ and $\delta x_i = x_i - x_i^+$. We will show that $\delta E \geqslant 0$.[4]

$$\begin{aligned}
\delta E &= E - E^+ \\
&= 2\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta} - \mathbf{x}^{\mathrm{T}}\mathbf{W}\mathbf{x} - 2\mathbf{x}^{+\mathrm{T}}\boldsymbol{\theta} + \mathbf{x}^{+\mathrm{T}}\mathbf{W}\mathbf{x}^+ \\
&= 2\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta} - \mathbf{x}^{\mathrm{T}}\mathbf{W}\mathbf{x} - 2(\mathbf{x}-\delta\mathbf{x})^{\mathrm{T}}\boldsymbol{\theta} + (\mathbf{x}-\delta\mathbf{x})^{\mathrm{T}}\mathbf{W}(\mathbf{x}-\delta\mathbf{x}) \\
&= -\sum_{i,j}\delta x_i \delta x_j w_{ij} - \sum_{i,j}\delta x_i x_j w_{ij} - \sum_{i,j} x_i \delta x_j w_{ij} + 2\sum_i \delta x_i \theta_i .
\end{aligned}$$

Since the dynamics is serial, let us suppose that at iteration t the only neuron to be updated is i, so that, $\delta\mathbf{x} = [0,\ldots,0,\delta x_i,0,\ldots,0]^{\mathrm{T}}$, additionally we have considered $w_{ii} = 0$, and the equation above can be simplified to

$$\delta E = -\delta x_i \sum_j x_j w_{ij} - \delta x_i \sum_j x_j w_{ji} + 2\delta x_i \theta_i \tag{6}$$

and by the symmetry of $\mathbf{W}$, to

$$\delta E = -2\delta x_i \sum_j x_j w_{ij} + 2\delta x_i \theta_i = 2\delta x_i \left(\theta_i - \sum_j x_j w_{ij}\right) = 2\delta x_i P_i . \tag{7}$$

Note that $\delta x_i > 0$ if $P_i > 0$ and $\delta x_i < 0$ if $P_i < 0$ so that, $\delta x_i P_i > 0$ and therefore $\delta E \geqslant 0$. This completes proof of (a).

(b) In a stable state, $E(t) = E(t+1)$.

From the definition of θ_i, P_i is always non null, therefore, $\delta E = 0$ if and only if $\delta x_i = 0$, and the state where the energy is stopped is always stable, what proves (b).

In order to finish the proof we only need to show that the number of iterations to reach convergence is finite:

Let $\theta_i > 0$. Since $\max|x_i| = 1$, $|E| < 2\sum_i |\theta_i| + \sum_{i,j} w_{ij}$, the energy has a lower bound. In addition $P_i = \theta_i - \sum_j w_{ij}x_j$ with $\theta_i \notin S$, so that $\min|P_i| > 0$. We can even choose θ_i to be the average of two consecutive levels in S, having $\min|P_i| = \Delta/2$. In this last case, $\min|\delta E| = \Delta^2$; in general $\min|\delta E| = 2\Delta\min|P_i|$. Finally, since E has a lower bound and $\delta E \leqslant 0$, if $\min|\delta E| > 0$, there must be a finite number of iterations to reach the convergence to a stable state. □

[4] We have defined δE as $\delta E = E - E^+$ for it to be consistent with the dynamics of the network defined by $P_i(t)$ in section 2. Note that with the definitions above $\delta E \geqslant 0$ implies that $E(t) \geqslant E(t+1)$.

The next step is to provide an upper bound for this finite number of iterations to get a stable state. We first give a bound on lemma 5 based on the Energy function defined for the neural network following a similar statement provided in [29]. We also prove that this bound is related to the structure of the graph for the neural network in theorem 6.

Theorem 5. Let $\alpha = 2\sum_i |\theta_i| + \sum_{i,j} w_{ij} \geqslant \max|E|$, with $\theta_i > 0$, the maximum number of iterations the neural network needs to converge to a stable state is $2\alpha n/\Delta^2$.

Proof. Let m be the *effective* number of iterations to get convergence. By *effective* we mean those iterations in which $\delta E < 0$. Note that

$$m \cdot \min|\delta E| \leqslant 2\max|E|. \tag{8}$$

We can choose $\min|\delta E| = \Delta^2$ (see theorem 4) and, from the definition of α, we have that

$$m \leqslant \frac{2\max|E|}{\min|\delta E|} \leqslant \frac{2\alpha}{\Delta^2}. \tag{9}$$

In each cycle there is at least one *effective* iteration (otherwise, the network is already in a stable state), so that, in the worst case (one *effective* iteration per cycle) the number of iterations needed to get a stable state is m times n, and the lemma is proved. □

In order to find the influence of the structure of the network graph in the number of iterations needed to converge, let deg be the average number of edges per node in the network graph [14], then taking into account that $|\theta_i| \leqslant 1$, and noting that $\sum_{i,j} w_{ij}$ is twice the total number of edges in the network graph, it is easy to observe that

$$\alpha = \sum_i |\theta_i| + \sum_{i,j} w_{ij} \leqslant n + 2 \cdot \deg \cdot n = n(2 \cdot \deg + 1) \tag{10}$$

and using lemma 5, we arrive to

Theorem 6. The maximum number of iterations for the neural network to converge to a stable state is

$$2(2 \cdot \deg + 1)\left(\frac{n}{\Delta}\right)^2.$$

However, this bound is not tight: it can be noted in proof for lemma 5 that the assumption of only one neuron updated per cycle is a worst case scenario which may be quite unlikely. This assumption causes the number of iterations to be a quadratic function of n/Δ, however, if this assumption is relaxed, this function can be expected to be linear. Below in this section, we elaborate a subclass of the proposed quantized neural network which exhibits a much faster convergence and, in next section, we give some experimental results for the LAP in which the general class of quantized Hopfield net shows a linear behavior, as expected.

Lemma 7. Let $\forall i$, $\theta_i = \theta$, N_i be the neighborhood of x_i and t be the iteration to update neuron i; if $x_i(t) > \theta$ and $P_i(t) > 0$, x_i will converge to state "1" in less than n/Δ iterations.

Proof.

$$\forall k \in N_i, \quad P_k(t') = \theta - x_i(t') - \sum_{j \in N_k, j \neq i} x_j(t') < 0 \quad \text{for } t < t' < t + n,$$

which makes

$$P_i(t+n) = \theta - \sum_{j \in N_i} x_j(t+n) \geqslant \theta - \sum_{j \in N_i} x_j(t) > 0.$$

Therefore, since in every cycle all the neurons are updated, in at most $1/\Delta$ cycles x_i will converge to state "1" and, in at most $1/\Delta$ cycles, the neurons in N_i will converge to state "0". As there are n iterations per cycle, the lemma is proved. □

If the class of quantized neural networks is further constrained to have $\theta_i = \theta = \Delta/2$, $\forall i$, we have the following:

Theorem 8. The quantized neural network with $\theta_i = \theta = \Delta/2$, $\forall i$ will converge to a stable state in less than $2n/\Delta$ iterations.

Proof. For neuron i, $P_i(0) > 0$ or $P_i(0) < 0$.

(1) If $P_i(0) > 0$, we have that $x_i(1) > \theta$ and $P_i(1) > 0$, so that by lemma 7, this neuron and its neighborhood will converge in at most $1/\Delta$ cycles,

(2) Let us now consider that $P_i(0) < 0$; two cases are possible:

(a) There exist $j \in N_i$ with $P_j(0) > 0$. In this case, x_j will converge to 1 and its neighborhood, x_i included, to '0' in at most $1/\Delta$ cycles.

(b) For every neuron $j \in N_i$, $P_j(0) < 0$. In this other case,

(b.1) either there is $t \in [0, n/\Delta]$ and $j \in N_i$ such that $P_j(t) > 0$, or

(b.2) for every $t \in [0, n/\Delta]$ and every $j \in N_i$, $P_j(t) < 0$.

(b.1) implies that x_j will converge to '1', and x_i to '0' in another n/Δ steps at most, and (b.2) implies that after $1/\Delta$ cycles being $P_j < 0$ for every $x_j \in N_i$, all of them have gone to '0' and $P_i(n/\Delta) = \theta - \sum_{j \in N_i} x_j > 0$ so it will take some more $1/\Delta$ cycles, at most, for x_i to converge to '1'.

Thus, complementary cases (b.1) and (b.2) give $2/\Delta$ cycles at most to converge, and the proof is completed. □

Note that for $\Delta = 1$, this particular class of neural network is equivalent to Shrivastava et al.'s neural network [27], and theorem 8 gives a maximum number of iterations to converge to the stable state of $2n$: the same result as reported in [27].[5] Therefore, we

[5] Note however that Shrivastava et al.'s neural network does not assume $\Theta_i = 1/2$, as theorem 8 does.

consider our quantized-state neural network a generalization of that network.

4. Simulations and results

This section shows the performance of the discrete Hopfield network proposed by means of simulations on two problems: The Linear Assignment Problem (LAP) and the Frequency Assignment Problem (FAP). We use the simulations in the LAP problem for illustrate the main theoretical results obtained in the previous sections. The application of the discrete Hopfield network to an NP-complete combinatorial optimization problem is shown by means of the FAP problem.

4.1. Application of the discrete network to the Linear Assignment Problem

In this section, we first introduce LAP and show this problem can be tackled if the neural network for the MIS is allowed to have initial values inside the unit hypercube. We then illustrate the scalability behavior of the quantized network for this problem.

4.1.1. The Linear Assignment Problem

The LAP can be formulated as follows. Let M be a $k \times l$ array of numbers which, without loss of generality, can be assumed to belong to the interval [0, 1]. A solution for the LAP is the selection of one and only one entry for every row and every column of M in such a way that the sum of the selected entries be the maximum among every possible selection.

In order to apply the neural network to this problem, each entry of M is associated to a neuron in the network, hence the order of the neural network is $n = k \times l$. Since the constraints for this problem consist of allowing one and only one entry per row and one and only one entry per column, $\mathbf{W}$ is an $n \times n$ matrix with $w_{ij} = w_{ji} = 1$ if the entries of M associated to neurons x_i and x_j are in the same row or in the same column. Every feasible solution to the LAP can be interpreted as a MIS for the neural network graph. Additionally, we define the weight of a MIS as the sum of the entries of M associated to this MIS. The LAP can then be stated as the search for a maximal weight MIS. We will refer to this weight as the *cost* or *cost function* for the LAP.

To gain some insight into the properties and applicability of the neural network to find a maximal MIS, we include in here the following theorem by Wolf et al. [33]:

Theorem 9. If the LAP is encoded as the initial state of the neural network, the convergence to the euclidean-nearest feasible MIS constitutes optimal performance. For convenience, we include a proof of this theorem in the appendix.

From theorem 9, it is easy to see that in order to find an optimal solution for the LAP, a neural network for this MIS can be used by coding the entries of M as the (quantized) initial state of the assigned neurons in the network and allowing this neural network to converge to its nearest stable state. Of course, the last requirement does

not hold but it can be considered to be a heuristics. In this paper we evaluate, through the Monte Carlo method, the performance of the neural network alone and study its scalability for the LAP as well as the quality of solutions for different values of the structure parameters.

4.1.2. Experiments on the Linear Assignment Problem

We present below the results of two experiments. In the first one, we compare the performance of the quantized neural network for $\theta_i = \theta = \Delta/2,\ \forall i$ (we call this $\Delta/2$-Hopfield) with Munkre's optimum algorithm [18]. In the second experiment we study the quantized neural network with $\theta_i = \theta = 0.5,\ \forall i$ (0.5-Hopfield) to show that the scalability properties remaining similar to the neural network of the first experiment, but the quality of the results is somehow improved. These results corroborate our belief that the bound calculated by using Energy function arguments in theorem 6 is quite pessimistic, and indicate that thresholds closer to "1" state give better results than thresholds closer to "0" state.

For both experiments, we average results for 100 random problem instances (i.e., for 100 different initial values) and repeat the calculations for two different array sizes: 10×10 and 50×50. The performance of the neural network and the average number of cycles for this network to converge are reported jointly for both experiments to allow an easier comparison. Note that the number of cycles instead of iterations are reported in the following.

4.1.3. Results on the Linear Assignment Problem

Figure 2 shows the quality of the solutions obtained for the 10×10 LAP with different quantization steps sizes of the neural networks in both experiments; the mean optimum values are given for comparison. As already mentioned, the neural network alone is not competitive with the optimum algorithm solutions, however, there is a reasonable improvement with respect to the 2-states neural network whose values can be observed in the graph for $\Delta = 1$. The change of the threshold for the neural network allows to increase its performance substantially. Since the aim of this paper is not the assignment problem, no further attempt has been made to find an optimal value for the threshold, although the rule suggested by this experiments is to put the threshold close to state "1". It is also observed that there is not further improvement for values of $1/\Delta$ greater than about 25, for $\theta = \Delta/2$ and about 40 for $\theta = 0.5$, which means that the maximum performance of this scheme is reached for roughly these values.

Figure 3 gives the number of cycles for the neural networks in both experiments to reach a stable state for 10×10 problem size. From theorem 8 it is known that for $\theta = \Delta/2$, the maximum number of cycles is $2/\Delta$. It can be observed that the actual number of cycles in simulations is about $1.5/\Delta$ for this threshold. For $\theta = 0.5$, the upper bound of theorem 8 does not hold, however, the number of cycles is slightly larger for this problem size.

The results for $N_M = 50 \times 50$ are given in figures 4 and 5, respectively. In this case, the improvement in the quality is better with respect to the digital neural network,

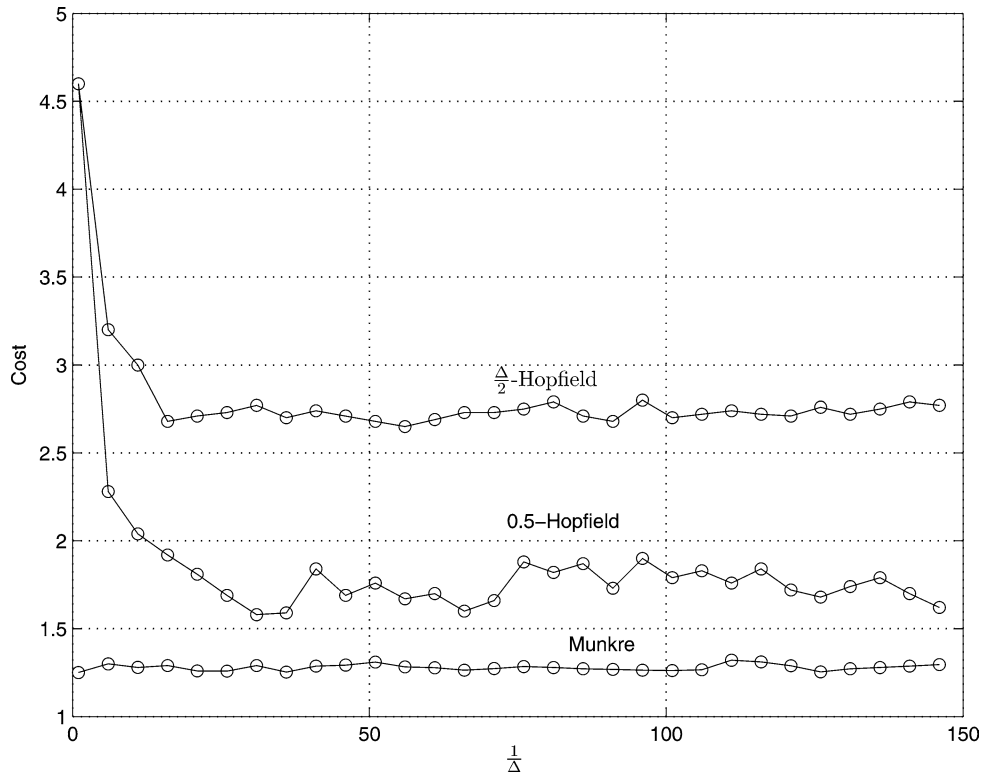


Figure 2. Cost obtained by the neural networks in experiment 1 ($\Delta/2$-Hopfield) and experiment 2 (0.5-Hopfield), both for $N_M = 10 \times 10$. Munkre's optimum is given for comparison.

but the neural network does worse with respect to the Munkre's optimum. Interestingly, the value of $1/\Delta$ to obtain a good result is about 50, which is only twice the number for $N_M = 10 \times 10$ and $\theta = \Delta/2$, and even less for $\theta = 0.5$ when the neural network has 25 times more neurons. The number of cycles to converge grows for $\theta = \Delta/2$ as $1.5/\Delta$ which is the same figure that for $N_M = 10 \times 10$; for $\theta = 0.5$, the number of cycles is growing above the value given by theorem 8, but the important fact to note is that this growth is linear with $1/\Delta$ and not quadratic, as theorem 6 bounds. Finally, note that the improvements derived from the continuous state neural network in the LAP, for instance those by Wolf et al. in their paper, can be easily applied to this quantized network.

4.2. Application of the quantized network to an instance of the Frequency Assignment Problem

Frequency Assignment Problem (FAP) is a well-known NP-complete problem [9], which has been the focus of multiple works involving artificial intelligence tools such as Hopfield neural networks [8], genetic algorithms [20], or other heuristics [13,31].

The FAP in a mobile communications network can be stated as follows: given a mobile communication network formed by N cells, a set of M available frequencies and

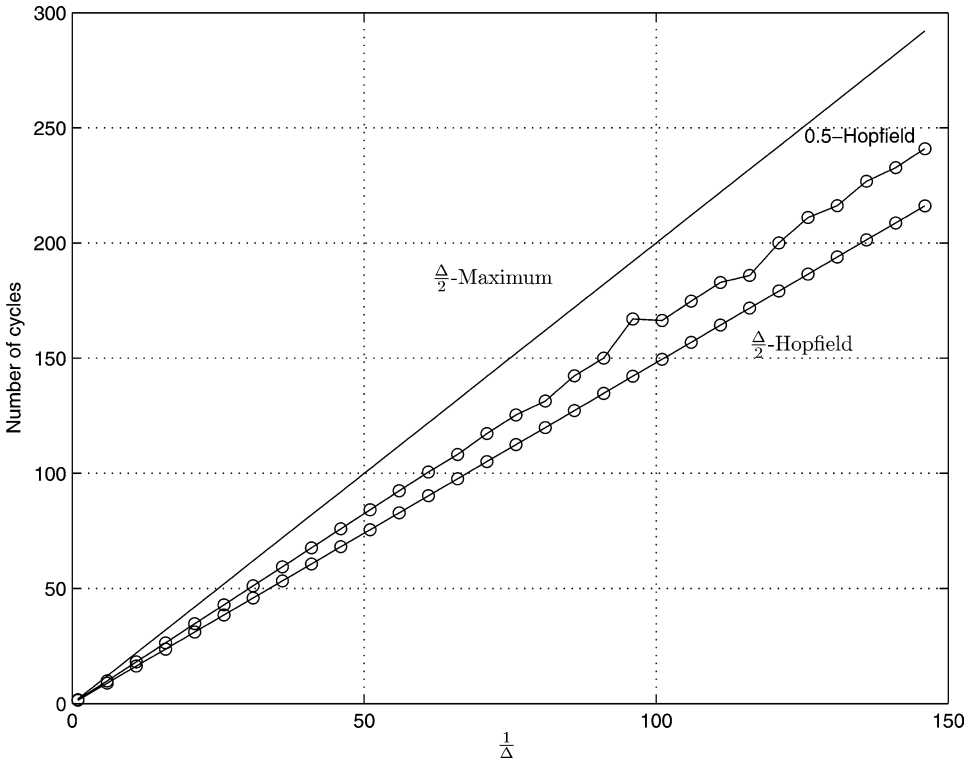


Figure 3. Number of cycles to obtain a stable solution for the neural networks in experiment 1 ($\Delta/2$-Hopfield) and experiment 2 (0.5-Hopfield), both for $N_M = 10 \times 10$. The bound for $\theta = \Delta/2$ given by theorem 8 is also included for reference.

a vector of frequencies requested of length N, $\mathbf{v}$ (v_i the number of frequencies requested by cell i), achieve an assignment of frequencies to every cell, in such a way that the system is free from interferences.

The minimum distance between frequencies in the system in order to avoid interferences is given by an integer matrix D. The solution of the problem can be represented as an $N \times M$ binary matrix X, where the rows represents the cells and the columns represents the available frequencies. An element $x_{ij} = 1$ means that the frequency j has been assigned to the cell i.

Mathematically the problem can be defined as: Find an assignment X, which maximizes

$$\sum_{i=1}^{N} \sum_{j=1}^{M} x_{ij} \tag{11}$$

and such that if $x_{ik} = 1$ and $x_{jl} = 1$:

$$|k - l| \geqslant d_{ij} \tag{12}$$

with the exact number of requested frequencies v_i assigned to every cell.

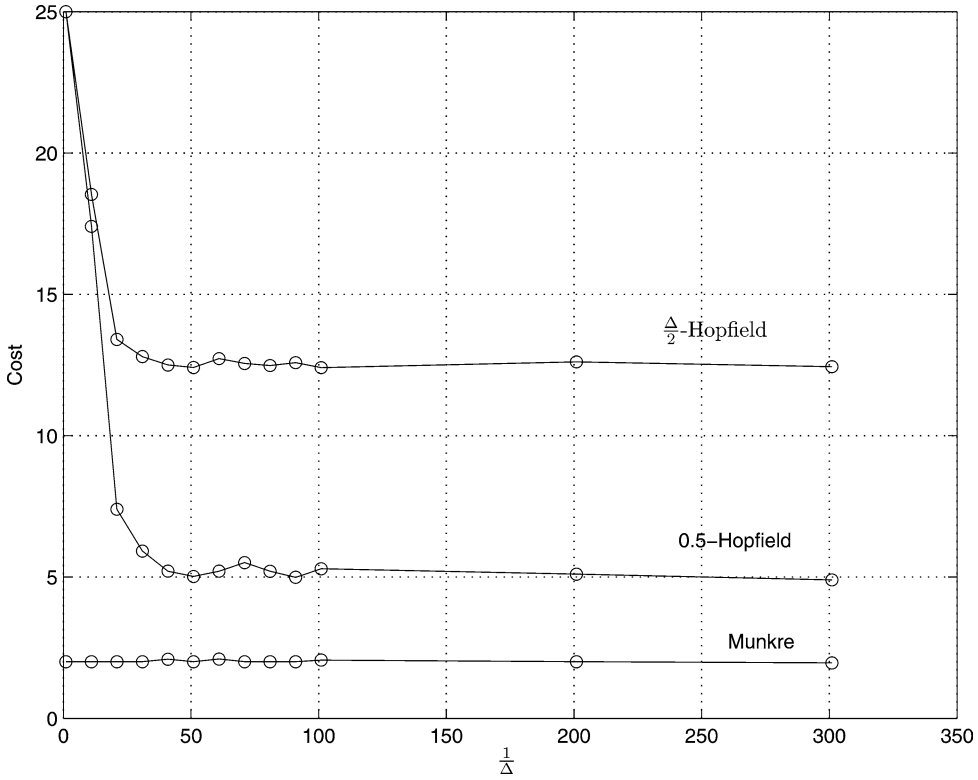


Figure 4. Cost obtained by the neural networks in experiment 1 ($\Delta/2$-Hopfield) and experiment 2 (0.5-Hopfield), both for $N_M = 50 \times 50$. Munkre's optimum is given for comparison.

4.2.1. Results in the Frequency Assignment Problem

We have tackled an instance of the FAP given in [9] as Problem #3. The problem consists of a mobile communication network formed by 25 cells, where 73 frequencies are available to be used. The compatibility matrix of the problem as well as the vector of frequencies required by each cell is given in reference [9]. The optimum assignment for this problem involves 167 assigned frequencies.

The quantized neural network presented in this paper is able to solve the constraint given by equation (12), however, it cannot manage the constraint of obtaining the exact number of frequencies requested by each cell. In order to solve this, a first approach can be carried out by means of the recursive application of the proposed network. In a first step, the network is initializated at random, and it is run until a solution fulfilling equation (12) is obtained. After removing at random the spare frequencies of every cell, the neural network is run again using the previous assignment as initial point.

This heuristic can be outlined in pseudocode as follows:

0. Initialize the quantized neural network, at random.

   $M = 1$;

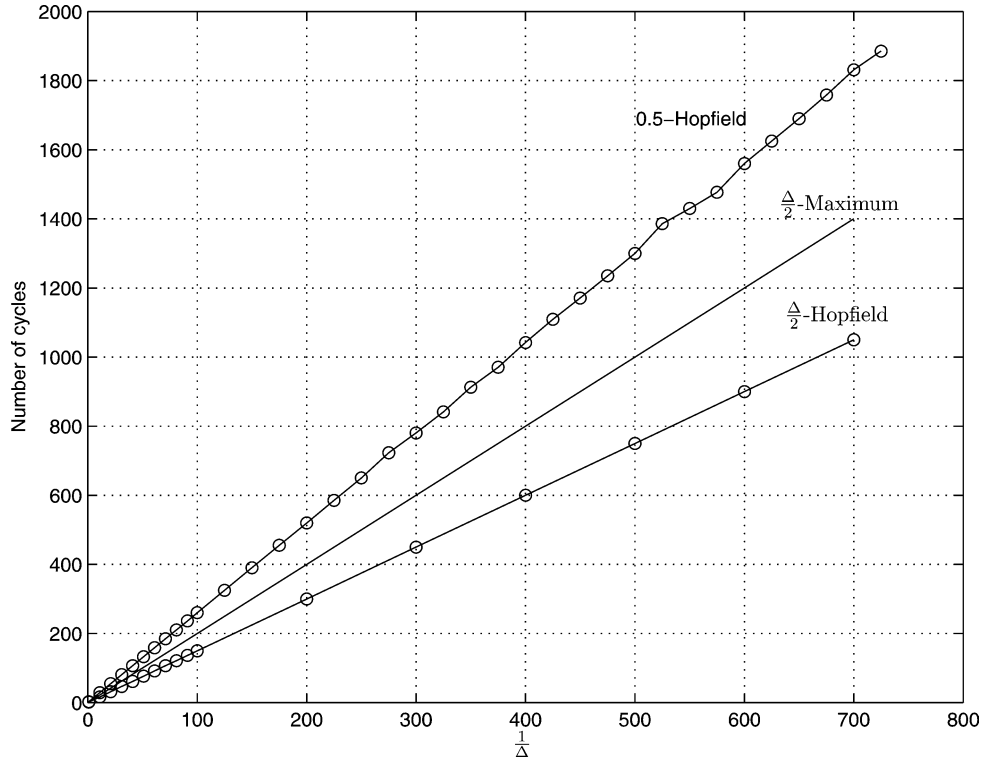


Figure 5. Number of cycles to obtain a stable solution for the neural networks in experiment 1 ($\Delta/2$-Hopfield) and experiment 2 (0.5-Hopfield), both for $N_M = 50 \times 50$. The bound for $\theta = \Delta/2$ given by theorem 8 is also included for reference.

1. Run the quantized neural network without the requested frequencies constraint.
2. **for** (each cell)

   if there are more assigned frequencies than requested, remove the spare frequencies at random.

   endfor
3. Use the obtained assignment as the initial state for another run of the network.
4. $M = M + 1$, go to 1.

Note that since the quantized neural network is bounded in complexity by the bound in theorem 8 ($2/\Delta$ cycles for convergence), the maximum number of cycles for convergence using the proposed heuristic is $2M/\Delta$.

Figures 6 and 7 show the performance of the quantized neural network in the tackled instance of the FAP, 2 and 5 states, respectively. Each point has been calculated by averaging the results obtained in 200 runs of the the network with each value of M. If we stop the heuristic in $M = 1$, it is easy to see that the network of 5 states achieves a better solution than the network of 2 states (roughly 128 assigned frequencies against 111).

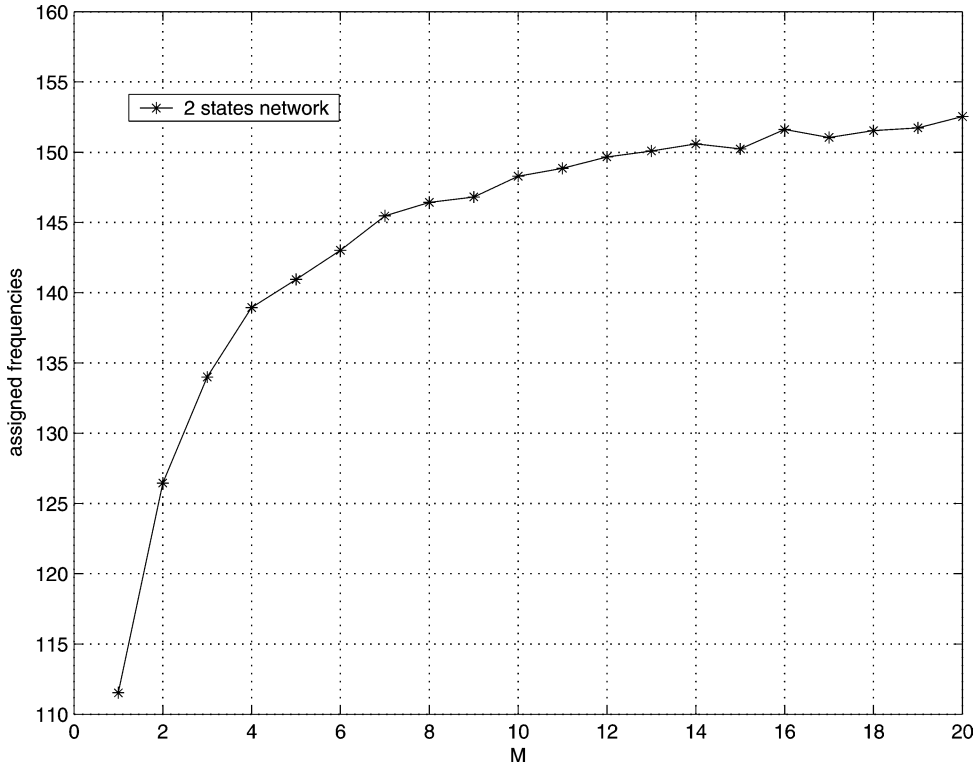


Figure 6. Number of assigned frequencies for each value of M. Two states network ($\Delta = 1$).

When the value of M grows the number of assigned frequencies is slightly better using the 5 states network, but differences are small. The best assignment achieved by the 2 states network allocated 161 frequencies, out of 167 possibles. The best assignment achieved by the 5 states network allocated 164 out of 167.

Figure 8 shows the number of cycles needed by 5 states and 2 states network for converging to a stable solution. Note that, following theorem 8, the maximum number of cycles for the 2 states network is 2, whereas the maximun number of cycles for the 5 states network is 8. In both cases the heuristic used introduces a linear increasing of the number of cycles with parameter M. In order to have a reliable comparison measure of the computational cost, we can take the Hopfield network approach proposed in [9] for this instance of the FAP, which converges in average in 294 cycles. This result shows that our approach is appropriate for controlling the computational cost maintaining the performance of the network in acceptable levels.

The mixing of the quantized Hopfield network with global search algorithms such as Genetic Algorithms or Simulated Annealing is expected to improve the quality of the solutions obtained.

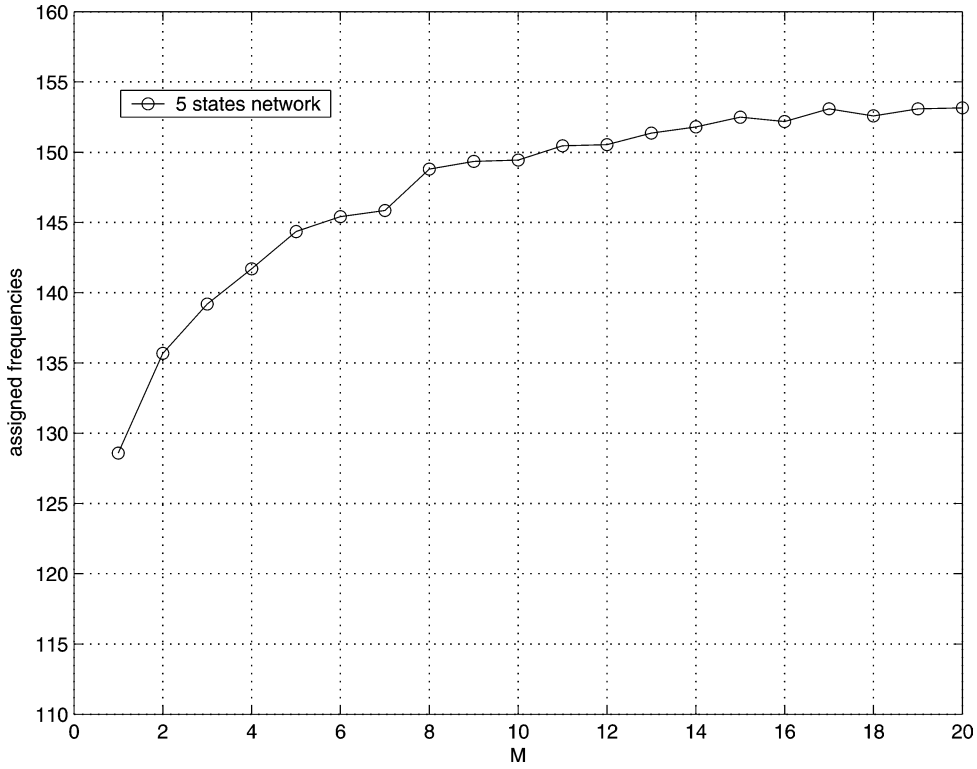


Figure 7. Number of assigned frequencies for each value of M. Five states network ($\Delta = 4$).

4.3. Heuristics for the FAP based on the quantized Hopfield network

The recursive heuristic using the quantized network presented in the previous section provides feasible solutions for the FAP, in terms of equation (12). However these solutions can be considered low quality solutions, since the average number of assigned frequencies is about 155 out of 167. In order to improve the quality of solutions obtained, different heuristics based on the quantized Hopfield network can be proposed.[6]

We present here one of these heuristics, which operates by means of adding a new rule in the network dynamics. Thus, the quantized Hopfield network dynamics described in section 2 by means of four rules (R1–R4) includes now a new rule R5:

R5. Heuristic of requested frequencies: if $\sum_{i \in \text{cell } k} x_i(t) > v_k$ then $x_j = 0$, $j \in$ cell k, with j the neuron being updated.

Note that rule R5 eliminates the spare frequencies in a given cell, if existing. This new rule does not affect to the convergence of the network, but may slightly modify its computational cost, as we will show following.

[6] The majority of these heuristics try to eliminate extra assigned frequencies to a given cell.

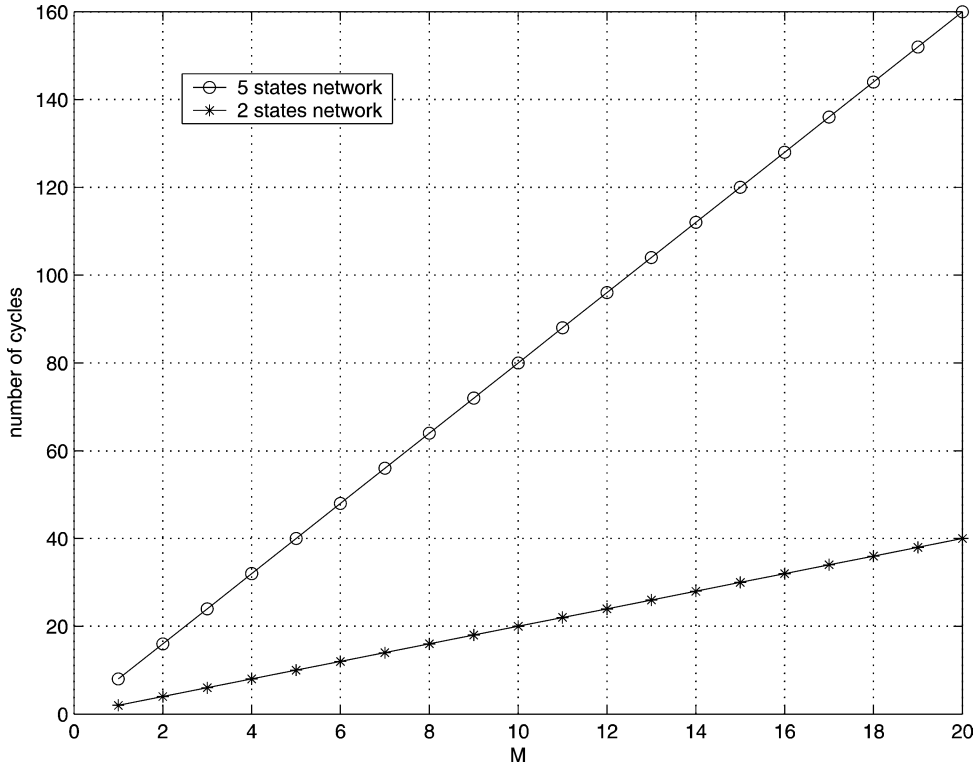
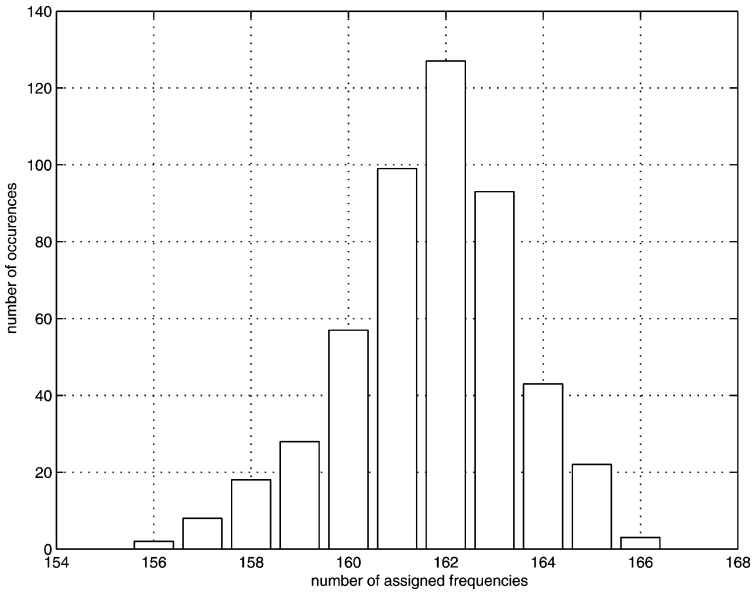


Figure 8. Number of cycles needed for convergence for each value of M. Two and five states networks.

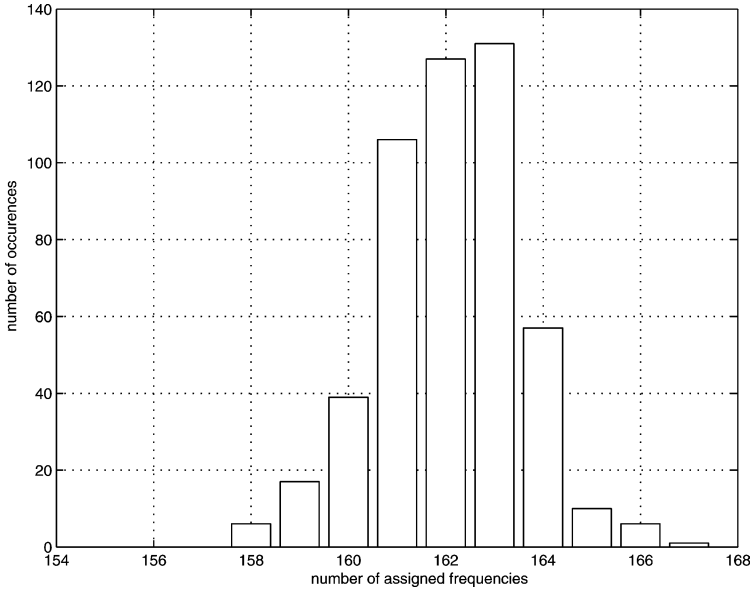
In order to show the performance of this heuristic we tackle the same instance of the FAP as in the previous section, with 25 cells and 73 available frequencies. Again, the quantized Hopfield network with $\Delta = 1$ (two states $S_2 = \{0, 1\}$) and $\Delta = 4$ (five states $S_5 = \{0, 0.25, 0.5, 0.75, 1\}$) are compared in performance and computational cost. As in the previous section, we also compare the computational cost of these network with the computational cost of a *gradual* Hopfield network approach given in [9] for the considered FAP instance.

Figures 9(a) and (b) shows the histograms of the solutions obtained by quantized Hopfield network including rule R5, using $\Delta = 1$ and $\Delta = 4$, respectively, in 500 network runs. The optimum assignment in this instance would include 167 assigned frequencies. Note that the network with $\Delta = 4$ obtains better results than the network with $\Delta = 1$, even obtaining the optimal assignment. In addition, using the network with $\Delta = 4$ low quality solutions assigning 156 and 157 frequencies are avoided.

On the other hand, figure 10 shows the computational cost of the quantized networks with $\Delta = 1$ (a) and $\Delta = 4$ (b). The network with $\Delta = 1$ converges much faster than the network with $\Delta = 4$ (4 cycles against 22 in average), as expected. Note that these results shown that the heuristic included in the network dynamics as rule R5 for the FAP modifies the convergence properties of the quantized network. Following theo-

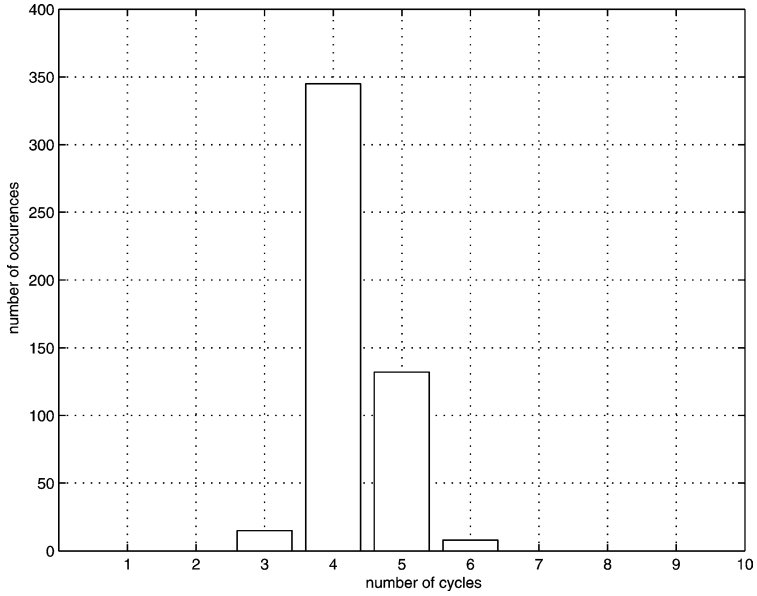


(a)

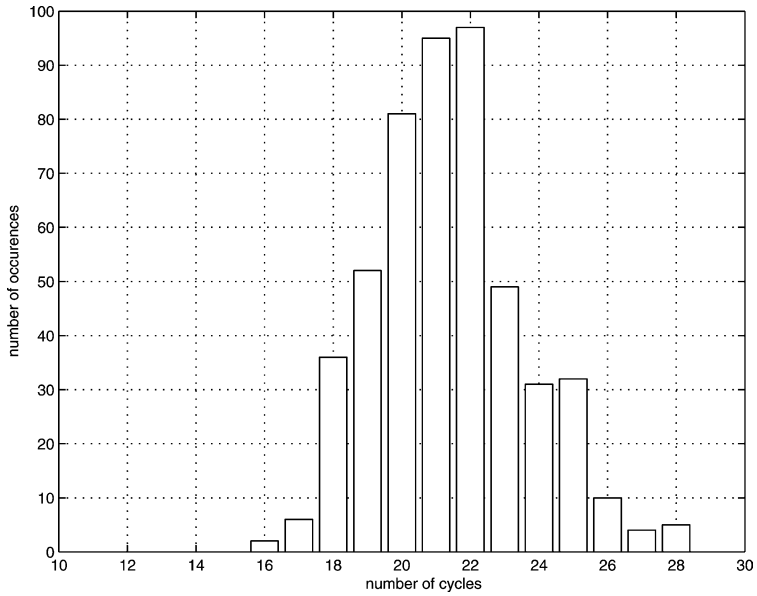


(b)

Figure 9. Comparison of the number of assigned frequencies. (a) Network of two states ($\Delta = 1$); (b) Network of five states ($\Delta = 4$).



(a)



(b)

Figure 10. Comparison of the number of cycles needed for convergency. (a) Network with $\Delta = 1$; (b) Network with $\Delta = 4$.

rem 8 the maximum number of cycles for convergence is bounded by $2/\Delta$, however the obtained results on the FAP show that this limit is not fulfilled by the network with the new rule R5, and the number of cycles until convergence is slightly larger, about twice the upper bound given by theorem 8.

In order to have a reliable comparison measure of the computational cost, we can take the Hopfield network approach proposed in [9] for this instance of the FAP, which converges in average in about 280 cycles (obtaining slightly higher quality solutions in terms of assigned frequencies, see [9]). This result shows that our approach is appropriate for controlling the computational cost maintaining the performance of the network in acceptable levels.

5. Conclusions

This paper elaborates on the complexity of a quantized states-discrete time Hopfield neural network. This model extends the applicability of Shrivastava's by allowing inner states to the unit hypercube (see also Matsuda [22]). Quadratic complexity bounds are provided through the study of the Lyapunov function. This bounds however turn out to be quite pessimistic as an example in an instance of the linear assignment problem has shown. Furthermore, it is possible to formulate a constrained submodel which provably scales linearly in the number of states.

Acknowledgement

The authors would like to thank the anonymous reviewers for their useful comments and constructive hints.

Appendix: Proof of theorem 9

Let $M = \{m_{ij}\}$, the function maximized for the LAP can be written as $f = \sum_{i,j\in A} m_{ij}$, where A is the set of pairs (i, j) corresponding a feasible solution. The euclidean distance, d, from the initial state to any feasible solutions is given by

$$d^2 = \sum_{i,j\in A} (m_{ij} - 1)^2 + \sum_{i,j\in B} m_{ij}^2 = N_A + \sum_{i,j} m_{ij}^2 - 2 \sum_{i,j\in A} m_{ij}$$

where B is the complementary of A and N_A is the number of elements in A.

Note that since the term $N_A + \sum_{i,j} m_{ij}^2$ is constant, to minimize d is equivalent to maximize f. This completes the proof.

References

[1] S. Abe, Global convergence and suppression of spurious states of the Hopfield Neural Networks, IEEE Trans. Circuits Syst. 40(4) (1993) 246–257.

[2] S.V.B. Aiyer, M. Niranjan and F. Fallside, A theoretical investigation into the performance of the Hopfield model, IEEE Trans. Neural Networks 1(2) (1990) 204–216.
[3] J. Balicki, A. Stateczny and B. Zak, Genetic algorithms and Hopfield neural networks for solving combinatorial optimization problems, Appl. Math. Comput. Sci. 7(3) (1997) 567–592.
[4] F. Barahona, On the computational complexity of Ising spin glass models, J. Phys. A: Math. General 15(10) (1982) 815–826.
[5] J. Bruck and J.W. Goodman, A generalized convergence thorem for neural networks, IEEE Trans. Inform. Theory 34(5) (1988) 1089–1092.
[6] M.A. Cohen and S. Grossberg, Absolute stability of global pattern formation and parallel memory storage by competitive neural networks, IEEE Trans. Systems, Man Cybernetic 13(5) (1983).
[7] F. Fabris and G.D. Riccia, An application of the Hopfield model to huffman codes, IEEE Trans. Inform. Theory 39(3) (1993) 1071–1076.
[8] N. Funabiki, N. Okutani and S. Nishikawa, A three-stage heuristic combined neural-network algorithm for channel assignment in cellular mobile systems, IEEE Trans. Vehicular Technol. 49(2) (2000) 397–403.
[9] N. Funabiki and Y. Takefuji, A neural network parallel algorithm for channel assignment problems in cellular radio networks, IEEE Trans. Vehicular Technol. 41(4) (1992) 430–437.
[10] S. Geman and D. Geman, Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images, IEEE Trans. Pattern Anal. Machine Intell. 6(6) (1984) 721–741.
[11] R.M. Golden, *Mathematical Methods for Neural Network Analysis and Design* (MIT Press, Cambridge, MA, 1996).
[12] A. Haken and M. Luby, Steepest descent can take exponential time for symmetric connectionist networks, Complex Systems 2(2) (1988) 191–196.
[13] J.K. Hao, R. Dorne and P. Galinier, Tabu search for frequency assignment in mobile radio networks, J. Heuristics 4(1) (1998) 47–62.
[14] N. Hartsfield and G. Ringel, *Pearls in Graph Theory. A Comprehensive Introduction* (Academic Press, San Diego, CA, 1994).
[15] J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proc. Natl. Acad. Sci. USA Biophys. 81 (1984) 3088–3092.
[16] J.J. Hopfield, Neurons with graded response have collective computational properties like those of two-states neurons, Proc. Natl. Acad. Sci. USA Biophys. 79 (1982) 2554–2558.
[17] J.C. Juang, Stability analysis of Hopfield-type neural networks, IEEE Trans. Neural Networks 10(6) (1999) 1366–1374.
[18] H.W. Khun, The hungarian method for the assignment problem, Naval Research Logistics Quarterly 2(83) (1955).
[19] P. Koiran, Dynamics of discrete time, continuous state Hopfield networks, Neural Comput. 6(3) (1994) 459–468.
[20] W.K. Lai and G.C. Coghill, Channel assignment through evolutionary optimization, IEEE Trans. Vehicular Technol. 45(1) (1996) 91–96.
[21] S. Matsuda, "Optimal" Hopfield network for combinatorial optimization with linear cost function, IEEE Trans. Neural Networks 9(6) (1998) 1319–1330.
[22] S. Matsuda, Quantized Hopfield networks for integer programming, Systems Comput. Japan 6 (1999) 1–12.
[23] S. Matsuda, Theoretical analysis of quantized Hopfield network for integer programming, in: *Proc. of the IEEE International Joint Conference on Neural Networks* (1999) pp. 568–571.
[24] Neurocomputing 8, Special Issue on Optimization and Combinatorics (1995).
[25] I. Parberry, Scalability of a neural network for the knight's tour problem, Neurocomputing 12 (1996) 19–34.
[26] S. Salcedo-Sanz, C. Bousoño-Calzón and A.R. Figueiras-Vidal, A mixed neural-genetic algorithm for the broadcast scheduling problem, IEEE Trans. Wireless Commun. 2(2) (2003) 277–283.

[27] Y. Shrivastava, S. Dasgupta and S.M. Reddy, Guaranteed convergence in a class of Hopfield networks, IEEE Trans. Neural Networks 3(6) (1992) 951–961.
[28] J. Sima, P. Orponen and T. Antti-Poika, On the computational complexity of binary and analog symmetric Hopfield nets, Neural Comput. 12(12) (2000) 2965–2989.
[29] K.Y. Siu, V. Roychowdhury and T. Kailath, *Discrete Neural Computation. A Theoretical Foundation* (Prentice Hall, Englewood Cliffs, NJ, 1995).
[30] A.M. Stuart and A.R. Humphries, *Dynamical Systems and Numerical Analysis* (Cambridge University Press, Cambridge, UK, 1998).
[31] C. Voudouris and E.P.K. Tsang, Solving the radio link frequency assignment problem using guided local search proceedings, in: *Proc. of NATO Symposium on Radio Length Frequency Assignment, Sharing and Conservation Systems (Aerospace)*, Aalborg, Denmark (October 1998).
[32] Y. Watanabe, N. Mizuguchi and Y. Fujii, Solving optimization problems by using a Hopfield neural network and genetic algorithm combination, Systems Comput. Japan 29(10) (1998) 68–74.
[33] W.J. Wolf, Inhibitory grids and the assignment problem, IEEE Trans. Neural Networks 4(2) (1993) 319–331.