Modelling competitive Hopfield networks for the maximum clique problem

G. Galán-Marín\textsuperscript{a}, *, E. Mérida-Casermeiro\textsuperscript{b}, J. Muñoz-Pérez\textsuperscript{c}

\textsuperscript{a}Department of Electronics and Electromechanical Engineering, University of Extremadura, Escuela de Ingenierías Industriales, Avda. de Elvas s/n, 06071 Badajoz, Spain
\textsuperscript{b}Department of Applied Mathematics, University of Málaga, Málaga, Spain
\textsuperscript{c}Department of Computer Science, University of Málaga, Málaga, Spain

Received 1 February 2001; received in revised form 1 January 2002

Abstract

The maximum clique problem (MCP) is a classic graph optimization problem with many real-world applications. This problem is NP-complete and computationally intractable even to approximate with certain absolute performance bounds. In this paper, we present the design of a new discrete competitive Hopfield neural network for finding near-optimum solutions for the MCP. Other competitive Hopfield-style networks have been proposed to solve the MCP. However, recent results have shown that these models can sometimes lead to inaccurate results and oscillatory behaviors in the convergence process. Thus, the network sometimes does not converge to cliques of the considered graph, where this error usually increases with the size and the density of the graph. In contrast, we propose in this paper appropriate dynamics for a binary competitive Hopfield network in order to always generate local/global minimum points corresponding to maximal maximum cliques of the considered graph. Moreover, an optimal modelling of the network is developed, resulting in a fast two-level winner-take-all scheme. Extensive simulation runs show that our network performs better than the other competitive neural approaches in terms of the solution quality and the computation time.

Scope and purpose

The MCP is a classic one in computer science and in graph theory, and is known to be NP-complete. Although many algorithms have been proposed for the MCP, usually the exponentially increasing computation time prohibits us from solving large-scale problems. The MCP is related to many real-life problems. These include classification theory, coding theory, project selection, fault tolerance, computer vision, signal transmission theory, VLSI circuit design, information retrieval, biological systems and economics. Hence, simple algorithms which yield acceptable solutions sufficiently fast are quite important for such related practical problems.

* Corresponding author. Tel.: +34-92-428-9600; fax: +34-92-428-9601.
E-mail addresses: gloriagm@unex.es (G. Galán-Marín), merida@ctima.uma.es (E. Mérida-Casermeiro), munozp@lcc.uma.es (J. Muñoz-Pérez).
A very powerful neural approach for the MCP has been presented by Lee and Takefuji, which is based on a competitive Hopfield-type network called maximum neural network. Lee and Takefuji showed that the maximum neural network solves large-scale MCPs in a reasonable computation time where the best-known algorithms cannot solve it. However, some authors have discussed the convergence properties of Takefuji’s model. According to these works, our simulation results in the MCP show that the percentage of randomly generated initial states that do not converge to a maximal clique of the graph oscillates between 8% and 62%, where this error increases with the size and the density of the graph.

In contrast, we present in this paper a competitive Hopfield model that always guarantees convergence to a maximum/maximal clique of the graph. Experimental results on the MCP show that our network is computationally superior to the parallel maximum neural network in terms of the solution quality and in terms of the computation time on a conventional sequential machine. Moreover, this superiority increases with the size and the density of the graph. © 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Maximum clique; Combinatorial optimization; Discrete Hopfield model; Maximum neural network; Algorithm

1. Introduction

In graph theory a clique is a completely connected subgraph and the maximum clique problem (MCP) is that of finding, for a given graph, the size of the largest clique. We should distinguish a maximum clique from a maximal clique. A maximal clique is a clique that is not a subset of any other clique. Then, a maximum clique is a maximal clique that has the maximum cardinality (number of vertices). The MCP is a well-studied NP-complete problem [1] and Pardalos and Xue [2] recently published a review with 260 references. This problem is computationally intractable even to approximate with certain absolute performance bounds [3,4].

Several practical applications of the MCP and related problems have been reported, such as classification theory, coding theory, project selection, fault tolerance, computer vision, signal transmission theory, VLSI circuit design, information retrieval, biological systems and economics (see [2,18], and the related references therein). Moreover, a number of graph theoretic problems can be transformed into the MCP. Hence, it is of interest to develop methods for finding the exact and also approximate solutions to it.

Many algorithms for the MCP have been proposed [5–9]. Among them, the branch-and-bound method has been the most intensively studied, but the exponentially increasing computation time prohibits us from solving large-scale problems. Pardalos and Phillips [6] formulated the MCP as a linearly constrained indefinite quadratic global optimization problem. However, this algorithm is only able to solve problems with < 75 vertices. Carraghan and Pardalos [7] proposed an algorithm based on partial enumeration. Although this algorithm can deal with up to 3000-vertex graph problems, it requires a prohibitively long computation time even for middle sized graph problems.

One possible and very promising approach to combinatorial optimization problems is to apply artificial neural networks. The first neural network for combinatorial optimization problems was the analog Hopfield model introduced by Hopfield and Tank [10] in 1985. In the Hopfield neural network, the gradient descent method seeks the local minimum of a given Liapunov energy function $E$, which represents the objective function and the constraints of the optimization problem.
Basically, the Hopfield model has two versions: analog [11] and discrete [12]. For the analog version, neurons have continuous values within (0,1), and for the discrete version the neurons have only two possible values, 0 or 1. Of the two versions, the analog is superior to the discrete in terms of the local minima problem, because of its smoother energy surface. Hence, the most famous neural approach to combinatorial optimization problems is the analog Hopfield network. However, owing to its continuous change in the state variable, its convergence is slow and this network does not always converge to a valid state [13]. Over the past 15 years, a significant amount of work has been reported on improving the analog Hopfield network [14–16]. However, usually these types of neural solutions cannot be still compared in terms of quality to those obtained using other optimization techniques.

A very powerful neural approach for combinatorial optimization problems has been presented by Takefuji and co-workers [17,18]. They have shown in some combinatorial optimization problems [19–23] that their neural algorithms perform better than the best-known algorithms. Takefuji et al. [17,24,25] found discrete neurons computationally more efficient than continuous neurons. Hence, they usually apply the equations of the analog Hopfield model to binary neurons. In 1992, Funabiki et al. [23] proposed a parallel Hopfield neural network based on the binary McCulloch–Pitts neuron model [26] to solve the MCP. Their simulation results showed that the computation time was shorter than that of better-known algorithms, while the solution quality was similar.

In 1996, Lee and Takefuji [18] proposed a different parallel neural network for solving the MCP, which was based on a competitive Hopfield-type network called maximum neural network [27]. The significant advantage of the maximum neural network in the MCP is that it does not demand fine-tuning of parameters, as most Hopfield neural networks do. The maximum neural network has been shown to provide powerful approaches for combinatorial optimization problems [27,19–22] and for polygonal approximation [28]. This network is composed of groups of neurons and a competitive winner-take-all rule is imposed for updating the neurons. Thus, the neuron with the maximum input per group is the only one that has nonzero output. Lee and Takefuji showed that the maximum neural network solves large-scale MCPs in a reasonable computation time, where in the best-known algorithms cannot be solved. They compared this neural algorithm with the algorithm of Pardalos and Phillips [6] and with the implicit enumerative algorithm of Carraghan and Pardalos [7], showing that their neural algorithm was much faster with very similar solution quality.

However, Tateishi and Tamura [29] and Wang [30] have discussed the convergence properties of Takefuji’s model, showing that it does not always guarantee the descent of the energy function. According to these works, our experimental results in the MCP show that many times the parallel maximum neural network does not converge to a local minimum of the energy function. Hence, reaching a maximal clique of the graph is not guaranteed. Our simulation results indicate that the percentage of randomly generated initial states that do not converge to a clique of the graph oscillates between 8% and 62%, where this error increases with the size and the density of the graph.

In contrast, we present in this paper a competitive Hopfield model that always guarantees convergence to a global/local minimum of the energy function, that is, a maximum/maximal clique of the graph. Although the structure of this new model is inspired from the maximum neural network, its dynamics are completely different. Instead of deriving them from the analog Hopfield model, as in Takefuji’s model, our dynamics are based on the discrete Hopfield model. The discrete Hopfield network is not usually found in the literature as a solver of combinatorial optimization problems, but as a content-addressable memory. Nevertheless, our recent models based on the discrete sequential
Hopfield model [31–33] have been shown to provide powerful approaches for combinatorial optimization. However, since the operation of this sequential model is based on the notion of a single update, it usually degrades the performance for very large size instances. In contrast, we present in this paper the design of a discrete Hopfield model based on the notion of group update introduced by Takefuji et al. in the maximum neural network. The result is a new network, namely the optimal competitive hopfield model (OCHOM), with a high speed of convergence even for large-scale problems. Our extensive simulation runs in the MCP show that the OCHOM is computationally superior to the parallel maximum neural network in terms of the solution quality and in terms of the computation time on a conventional sequential machine. Moreover, this superiority increases with the size and the density of the graph.

This paper is organized as follows. Section 2 presents the applied formulation of the MCP. Section 3 describes the parallel maximum network proposed by Lee and Takefuji for the MCP, and briefly discusses its behavior. Section 4 presents the proposed competitive Hopfield model and Section 5 applies it to the MCP. Finally, Section 6 evaluates the performance of the proposed OCHOM networks in solving the MCP compared with the maximum neural network. Conclusions are given in Section 7.

2. Formulation of maximum clique problem

Let $G = (V, E)$ be an arbitrary undirected graph, where $V = \{1, \ldots, n\}$ is the vertex set of $G$ and $E \subseteq V \times V$ is the edge set of $G$. $A = (a_{ij})_{n \times n}$ is the adjacency matrix of $G$, where $a_{ij} = 1$ if $(i, j) \in E$ and $a_{ij} = 0$ if $(i, j) \notin E$. The complement graph of $G = (V, E)$ is the graph $\tilde{G} = (V, \tilde{E})$, where

$$\tilde{E} = \{(i, j) | i, j \in V, i \neq j \text{ and } (i, j) \notin E\}.$$

Given a subset $S \subseteq V$, we call $G(S) = (S, E \cap S \times S)$ the subgraph induced by $S$. A graph $G = (V, E)$ is complete if all its vertices are pairwise adjacent, that is, $\forall i, j \in V, (i, j) \in E$. A clique $C$ is a subset of $V$ such that $G(C)$ is complete. The MCP requires a clique that has the maximum cardinality.

Pardalos and Rodger [8,9] showed that the MCP is equivalent to minimizing the following unconstrained quadratic zero-one programming problem

$$f(x) = X^t(A_{\tilde{G}} - I)X = -\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}x_i x_j,$$

where $x_i \in \{0, 1\}$ and $A_{\tilde{G}} = (\tilde{a}_{ij})$ is the adjacency matrix of the complement of $G$. The equivalence between quadratic 0–1 optimization and the max-cut problem has been pointed out by Hammer [34]. Barahona et al. [35] showed how an unconstrained quadratic zero-one programming problem is equivalent to minimizing the weight summation over the same partition in a newly constructed graph $G_M$ with vertex set $V_M = \{0, 1, 2, \ldots, n\}$. By applying these results, Lee and Takefuji [18] finally formulate the MCP problem as the global minimization of the function

$$E = \sum_{x=0}^{n} \sum_{y=0}^{n} \sum_{i=1}^{2} I_{x,y} v_i v_j,$$  \hspace{1cm} (1)
subject to the constraints $\sum_{i=1}^{2} v_{xi} = 1$ for $x = 0, 1, \ldots, n$, where $v_{xi} \in \{0, 1\}$, and $v_{01} = 1$. The weight matrix is defined by

$$t_{0i} = t_{i0} = \frac{1}{4} \left( \sum_{j=1}^{n} a_{ij} - 1 \right), \quad i = 1, \ldots, n,$$

$$t_{ii} = 0; \quad t_{ij} = t_{ji} = \frac{1}{4} a_{ij} \quad \forall i \neq j, \quad i, j = 1, \ldots, n. \quad (2)$$

In this formulation, the first step in solving the MCP is to construct a graph $G_M$ by adding a vertex 0 to $\bar{G}$ (the complement graph of $G$). Fig. 1 shows a graph $G$ with 10 vertices and 34 edges taken from [18] and the graph $G_M$. Three maximum cliques of $G$ are shown in Fig. 2. The weight matrix
of $G_M$ is given by (2), where $t_{ij}$ represents the weight of an edge between vertices $i, j$. The second step is to minimize the summation of the weights whose vertices belong to the same partition set, where $V_M$ is partitioned into

$$V^+ = \{x/v_{x1} = 1, v_{x2} = 0\}$$

corresponding to the vertices in the clique and

$$V^- = \{x/v_{x1} = 0, v_{x2} = 1\}$$

corresponding to the vertices not included in the clique.

In this paper, we will apply this formulation of the MCP, proposed by Lee and Takefuji [18]. The problem can be represented and solved using an $(n + 1) \times 2$ binary neural network, where $n$ is the number of vertices of the graph. The binary output of the $x$th neuron $v_{x1} = 1$ means the $x$ vertex belongs to the partition $V^+$ (the clique) and $v_{x1} = 0$, that is, $v_{x2} = 1$, means the $x$ vertex belongs to the partition $V^-$ (not included in the clique). Observe that we must impose $v_{01} = 1, v_{02} = 0$, that is, the 0 vertex is always assigned to the partition 1. The total number of vertices in the final solution is given by $\sum_{x=1}^{n} v_{x1}$.

The advantage of this formulation of the MCP applied to the maximum neural network or to the OCHOM is that in both networks one and only one neuron per group is activated. Then, if it is considered that every group of the model is a row of the network, the constraint $\sum_{i=1}^{2} v_{xi} = 1$ is always automatically satisfied. In this way, the MCP reduces to obtain a global minimum of the energy function (1) with none of the parameters affecting the global minimum search.

3. The parallel maximum network for MCP

The maximum neural network was introduced by Takefuji et al. [27]; it presents the following advantages in solving the MCP: (1) coefficient-parameter tuning is not needed while conventional neural networks might suffer from it and (2) the equilibrium state is clearly defined in order to terminate the algorithm while the existing neural networks do not usually have the clear definition. Moreover, this neural algorithm for large-size problems outperforms the best-known algorithms in terms of computation time with much the same solution quality. In these cases, the conventional branch-and-bound method cannot be used due to the exponentially increasing computation time.

The maximum neural network consists of $n$ groups of neurons where each group is composed of $m$ binary neurons. The input–output function of the $i$th neuron in the $x$th group is defined by

$$v_{xi} = \begin{cases} 1 & \text{if } u_{xi} = \max\{u_{x1}, \ldots, u_{xm}\}, \\ 0 & \text{otherwise}, \end{cases}$$ (3)

where $v_{xi}$ and $u_{xi}$ are the output and the input of the $x$th neuron, respectively. Takefuji et al. [18,17,27] considered that the change of the input of the $x$th neuron was given by the equation of the analog Hopfield model

$$\frac{du_{xi}}{dt} = -\frac{\partial E(v_{11}, \ldots, v_{x1}, \ldots, v_{nm})}{\partial v_{xi}},$$ (4)
where $E$ is the energy function of the network. In the MCP, we have from (1) that this equation reduces to

$$\frac{du_{xi}}{dt} = -\sum_{y=0}^{n} t_{xy}v_{yi}.$$ 

However, in practice, they approximate by the first-order Euler method in the form

$$\Delta u_{xi}(t) = -\sum_{y=0}^{n} t_{xy}v_{yi}(t).$$

The following procedure describes the algorithm proposed by Lee and Takefuji for solving the MCP based on the described parallel maximum network.

1. Set $t = 0$ and $v_{01} = 1, v_{02} = 0$.
2. Assign a random number $\in (-1, 1)$ to the initial values of $u_{xi}$, for $x = 1$ to $n$, and $i = 1$ to $2$.
3. Evaluate the values of $v_{xi}$, for $x = 1$ to $n$, and $i = 1$ to $2$ by the function

$$v_{xi}(t) = \begin{cases} 1 & \text{if } u_{xi}(t) = \max\{u_{x1}(t), u_{x2}(t)\}, \\ 0 & \text{otherwise} \end{cases}$$

4. Compute $\Delta u_{xi}(t)$, for $x = 1$ to $n$, and $i = 1$ to $2$: $\Delta u_{xi}(t) = -\sum_{y=0}^{n} t_{xy}v_{yi}(t)$.
5. Compute $u_{xi}$ for $x = 1$ to $n$, and $i = 1$ to $2$: $u_{xi}(t + 1) = u_{xi}(t) + \Delta u_{xi}(t)$.
6. Increment $t$ by 1.
7. If the state of the network reaches an equilibrium state then stop, else go to step 3.

An equilibrium state of the network is defined as

$$x \in V^+ \iff \Delta u_{x1} \geq \Delta u_{x2}; \quad x \in V^- \iff \Delta u_{x2} \geq \Delta u_{x1}$$

in every group $x \in \{1, 2, \ldots, n\}$, where we always impose $0 \in V^+$.

In [18, 17, 27] Lee and Takefuji present theorems that guarantee the local minimum convergence of the maximum neural network. However, Tateishi and Tamura [29] discussed the convergence properties of the updating rule (4), applied by Takefuji and co-workers in all their works. They showed how, in the case of binary neurons, (4) does not always guarantee the descent of the energy function. Tateishi and Tamura expected that (4) would guarantee the decrease if continuous neurons were used. However, Wang [30] has recently shown that, even for continuous neurons, sometimes the energy is increased if we apply (4) for discrete time. According to these works, our simulation results for the parallel maximum neural network in the MCP show that the energy is not guaranteed to decrease and that the network sometimes oscillates between states not corresponding to cliques of the graph. Fig. 3 shows the percentage of initial states that do not converge to a clique of the graph. This average error, obtained in our simulation runs for the sizes $n = 50, 100$ and $500$, is represented for the nine densities from 0.1 to 0.9. In each size and density, 10 different random graphs were solved, and for every graph 10 simulation runs from different random initial states were performed. It is observed in Fig. 3 that the error in the parallel maximum neural network usually increases with the size and the density of the graph. Thus, the largest average error is obtained for the density 0.9, that is, 35%, 47% and 64% for the sizes $n = 50, 100$ and $500$, respectively.
An important difference between the parallel maximum neural network and the competitive Hopfield model that we propose for the MCP is the computation mode. In the OCHOM, we consider that only one group of two neurons is considered for updating at every time. On the other hand, in the parallel maximum network, all the groups of neurons are simultaneously updated. Therefore, this network may require less computation time implemented on a parallel machine. However, this completely synchronous mode does not guarantee the energy decrease and sometimes generates inaccurate results and oscillatory behaviors in the convergence process. Hence, the equilibrium state is not well defined. Next, we present a simple example to show this oscillatory behavior in the parallel maximum neural network for the MCP.

Let us consider the very simple graph in Fig. 4. The first step is to construct the graph $G_M$ and the weight matrix

$$T = (t_{ij}) = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{4} & 0 \\
0 & 0 & 0 & 0 & \frac{1}{4} \\
0 & \frac{1}{4} & 0 & 0 & 0 \\
0 & 0 & \frac{1}{4} & 0 & 0
\end{pmatrix}.$$
The second step is to assign random numbers $\in (-1, 1)$ to $u_{xi}(0)$, for instance

$$U(0) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} \end{pmatrix}.$$  

By the maximum function (3) we obtain

$$V(0) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix},$$

and then we have

$$U(1) = U(0) + \Delta U(0) = U(0) + TV(0) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} \end{pmatrix}.$$  

By applying the maximum function again, we obtain the new state of the network

$$V(1) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$  

If we continue applying the algorithm, then the network state $V(2)$ is again the state $V(0)$. Therefore, the network oscillates between the states $V(1)$ and $V(0)$, none of them corresponding to a clique, and never reaches an equilibrium state.

### 4. The optimal competitive Hopfield model

In this section, we present the design of a new maximum-type neural network with completely different dynamics that are inspired from the discrete Hopfield model. The proposed network $M$ consists of a single layer of $N$ binary interconnected neurons or processing elements. Let us now consider that the network is partitioned into $n$ disjoint groups, where each group is composed of $m$ neurons, such that $N = n \times m$. The output state of the $i$th neuron in the $x$th group is denoted
by \( v_{xi}(k) \in \{0,1\} \), its input by \( u_{xi}(k) \) and its bias by \( \theta_{xi} \), for \( x = 1, \ldots , n \), \( i = 1, \ldots , m \), where \( k \) denotes discrete time. The interconnection strength between neurons \( xi \) and \( yj \) is denoted by \( \omega_{xi,yj} \), for \( x, y = 1, \ldots , n \), \( i, j = 1, \ldots , m \), where symmetric weights \( \omega_{xi,yj} = \omega_{yj,xi} \) are considered. Observe that we allow arbitrary values of the self-connections \( \omega_{xi,xi} \) for each neuron.

The neural network is characterized by Hopfield’s energy function

\[
E(k) = - \frac{1}{2} \sum_{x=1}^{n} \sum_{i=1}^{m} \sum_{y=1}^{n} \sum_{j=1}^{m} \omega_{xi,yj} v_{xi} v_{yj} + \sum_{x=1}^{n} \sum_{i=1}^{m} \theta_{xi} v_{xi} \tag{5}
\]

and the inputs of the neurons are computed by Hopfield’s updating rule

\[
u_{xi}(k) = \sum_{y=1}^{n} \sum_{j=1}^{m} \omega_{xi,yj} v_{yj}(k) - \theta_{xi} \tag{6}\]

Since the network is partitioned into disjoint groups of neurons we can introduce the notion of group update. Thus, instead of selecting a single neuron for update as in the standard discrete Hopfield model, we select a complete group of neurons. Based on this notion, we now propose an original definition of a neural network model, namely the competitive Hopfield model (CHOM), and its optimal dynamics in order to guarantee and maximize the energy descent on every time step \( k \).

**Definition 1** (CHOM). Let \( M \) be a binary \((1/0)\) neural network characterized by an energy function (5) where the inputs of the neurons are computed by (6). If the network is partitioned into disjoint groups of neurons, then we shall say that \( M \) is a CHOM if one and only if one neuron per group has 1 as its output at every time \( k \).

**Theorem 1** (Energy change in the CHOM). Let \( M \) be a CHOM in which only one group \( x \), for \( x = 1, \ldots , n \), is selected for updating at time \( k \). Let \( xo \) be the neuron in group \( x \) with the output 1 at time \( k \) and \( xc \) the candidate neuron in group \( x \) that will have the output 1 at time \( k+1 \). Then, the energy difference resulting is

\[
\Delta E_x(k) = E(k+1) - E(k) = u_{xo}(k) - u_{xc}(k) + K_{xo,xc}, \tag{7}
\]

where

\[K_{xo,xc} = -\frac{1}{2}(\omega_{xo,xo} + \omega_{xc,xc} - 2\omega_{xc,xo})\].

**Proof.** It is easily shown that the energy difference for any change of the state of any neurons of the network is

\[
\Delta E = -\sum_{x=1}^{n} \sum_{i=1}^{m} \Delta v_{xi} \left[ -\theta_{xi} + \sum_{y=1}^{n} \sum_{j=1}^{m} \omega_{xi,yj} \left( v_{yj} + \frac{1}{2} \Delta v_{yj} \right) \right].
\]

If we consider discrete-time dynamics, then

\[
\Delta E(k) = E(k+1) - E(k), \quad \Delta v_{xi}(k) = v_{xi}(k+1) - v_{xi}(k),
\]

\[
\Delta E(k) = u_{xo}(k) - u_{xc}(k) + K_{xo,xc}.
\]
where \( k \) denotes discrete time. Consider now that only the states of the neurons in the group \( x \) are altered. Then, since the inputs of the neurons are computed by Hopfield’s updating rule (6), the energy change can be expressed as

\[
\Delta E(k) = \Delta E_x(k) = - \sum_{i=1}^{m} \Delta v_{x,i} \left[ u_{x,i}(k) + \sum_{y=1}^{n} \sum_{j=1}^{m} \frac{\omega_{x,i,yj}}{2} \Delta v_{y,j}(k) \right].
\] (8)

Let us suppose now that at time \( k \) the neuron \( x_o \) is the only one that is “on” in group \( x \) and that neuron \( x_c \) is the candidate neuron in group \( x \) that is going to be “on” at time \( k+1 \). Then we have that if \( c \neq o \), then \( \Delta v_{xo}(k) = -1 \); \( \Delta v_{xc}(k) = 1 \); \( \Delta v_{xi}(k) = 0 \), \( \forall i = 1, \ldots, m \), \( i \neq o, c \). By substituting these values we have from (8) that

\[
\Delta E_x(k) = -\Delta v_{xo}(k) \left[ u_{xo}(k) + \frac{\omega_{xo,xo}}{2} \Delta v_{xo}(k) + \frac{\omega_{xo,xc}}{2} \Delta v_{xc}(k) \right]
\]

\[
-\Delta v_{xc}(k) \left[ u_{xc}(k) + \frac{\omega_{xc,xc}}{2} \Delta v_{xc}(k) + \frac{\omega_{xc,xo}}{2} \Delta v_{xo}(k) \right] = u_{xo}(k) - [u_{xc}(k) - K_{xo,xc}].
\]

**Corollary (Optimal dynamics of the CHOM).** Let \( M \) be a CHOM in which only one group \( x \), for \( x = 1, \ldots, n \), is selected for updating at every time \( k \). Let \( x_o \) be the neuron in group \( x \) with the output 1 at time \( k \). If the dynamics of the CHOM are given by

\[
v_{x,i}(k+1) = \begin{cases} 
1 & \text{if } u_{x,i}(k) - K_{xo,xi} = \max_{j=1,\ldots,m} \{ u_{x,j}(k) - K_{xo,xj} \}, \\
0 & \text{otherwise},
\end{cases}
\] (9)

where \( K_{xo,xj} = -\frac{1}{2}(\omega_{xo,xo} + \omega_{xj,xj} - 2\omega_{xo,xj}) \), then convergence of the energy function to a local/global minimum is guaranteed.

**Proof.** From (7), we have that if the neuron with the maximum value of \( \{ u_{x,i}(k) - K_{xo,xi} \} \) per group \( x \) is always selected as the candidate neuron \( x_c \), then the energy descent, \( \Delta E(k)_x \leq 0 \), is guaranteed since the condition

\[
u_{xc}(k) - K_{xo,xc} \geq u_{xo}(k) - K_{xo,xo} = u_{xo}(k)
\]

is satisfied. Moreover, the absolute value of the energy decrease \( |\Delta E_x(k)| \) is the maximum possible at every time \( k \).

Observe that, since \( E \) is bounded from above, the energy function will converge in step \( k_e \). In this equilibrium value, it is verified in every group \( x \) of neurons that

\[
u_{xo}(k_e) = \max_{j=1,\ldots,m} \{ u_{x,j}(k_e) - K_{xo,xj} \}.
\]

If we activate any other neuron \( x_c \neq x_o \) in any group \( x \) in this stable state of \( E \), then we will have \( \Delta E_x(k_e) > 0 \) if \( u_{xc}(k_e) - K_{xo,xc} < u_{xo}(k_e) \) and \( \Delta E_x(k_e) = 0 \) if \( u_{xc}(k_e) - K_{xo,xc} = u_{xo}(k_e) \). Hence, the network is in a state corresponding to a local/global minimum of \( E \).
Definition 2. Let $M$ be a CHOM. We shall say that $M$ is an OCHOM if the dynamics of the network are defined by (9).

Note that the statement in Corollary 1 is true irrespective of the strategy for choosing the index $x$ of the group of neurons whose state is to be updated at time $k$. Then, we can choose $x$ in some different systematic fashions.

Definition 3. Let $M$ be an OCHOM network and $x$ the index of the group updated at time $k$, for $x = 1, \ldots, n$.

1. If $x$ is selected randomly at each time $k$, then we shall say that $M$ is a Random-OCHOM network (R-OCHOM).
2. If $x$ is selected such that we cycle through the groups one after the other, i.e., $x = k \mod n + 1$, then we shall say that $M$ is a Cyclic-OCHOM network (C-OCHOM).
3. If $x$ is selected as the group that generates the best energy difference, i.e.,
   \[ \Delta E_x(k) = \min_{y=1,\ldots,n} \{ \Delta E_y(k) \}, \]
   then we shall say that $M$ is a Best-OCHOM network (B-OCHOM).

Obviously, the B-OCHOM network is the OCHOM network that converges to a stable state in the fewest number of steps. Moreover, simulation results in the MCP show that this two-level winner-take-all structure is also the most accurate design for a competitive Hopfield network in this problem.

5. The OCHOM network for MCP

In order to obtain the connection weights $\omega_{xi,yj}$ and the biases $\theta_{xi}$ of the neural network, we will compare the energy function (1) defined for the MCP and the Hopfield energy function (5). If we substitute these values in Hopfield’s updating rule (6), then we obtain the input of every neuron in the OCHOM, given by

\[ u_{xi} = -2 \sum_{y=0}^{n} t_{xy} v_{yi}. \] (10)

Note that the bias for every neuron is $\theta_{xi} = 0$, the neural self-connection is $\omega_{xi,xi} = 0$ and the interconnection between neurons that belong to the same group (row) $x$ is $\omega_{xi,xj} = 0$. Consequently, $K_{xi,xj} = 0$ for every two neurons $xi$ and $xj$ in the same group (row) $x$. Hence, the difference in the energy that would result only if the states of the neurons in the group $x$ were altered is $\Delta E = u_{xo} - u_{xc}$, where the neuron $xc$ candidate to be “on” in group $x$ is

\[ u_{xc}(k) = \max_{i=1,2} \{ u_{xi}(k) \}. \]

As stated in Corollary 1, an equilibrium value, i.e., a local/global minimum of $E$, is reached when it is verified that

\[ x \in V^+ \iff u_{s1} \geq u_{s2}; \quad x \in V^- \iff u_{s2} \geq u_{s1} \]
in every group \( x \in \{1,2,\ldots,n\} \), where we always impose \( 0 \in V^+ \). Next, we will show that if we use the OCHOM with the formulation of the MCP described in Section 2, then every local/global minimum corresponds to a maximal/maximum clique of the graph. Then, since convergence to a local/global minimum is guaranteed, the OCHOM network always converges to a state corresponding to a maximal/maximum clique of \( G \).

**Theorem 2.** Given a graph \( G \), the graph \( G_M \) is defined by adding a vertex \( 0 \) to \( \bar{G} \), where the weight matrix is given by (2). If the OCHOM is applied to minimize the weight summation over the same partition in \( G_M \), then every local minimum corresponds to a maximal clique of \( \bar{G} \).

**Proof.** Assume that the network reaches an equilibrium value, i.e., a local/global minimum of \( E \). First, we shall prove that \( V^+ \) in this state corresponds to a clique of \( G \). Let us suppose that \( V^+ \) in this equilibrium state is not a clique. Hence, there exist two different vertices \( a,b \in V^+ \) such that an edge exists between \( a \) and \( b \) in \( \bar{G} \). Let us define that

\[
S_a = \{i/i \in V^+, i \neq a, A_{ia} = 1\}; \quad Q_a = \{i/i \in V^-, i \neq a, A_{ia} = 1\}.
\]

Since \( b \in S_a \), then \( \mathcal{C}(S_a) \geq 1 \) (where \( \mathcal{C}(X) \) is the number of vertices in \( X \)). Because the network is in an equilibrium state and \( a \in V^+ \)

\[
\begin{align*}
 u_{a_1} \geq u_{a_2} & \iff -2 \sum_{y=0}^{n} t_{ay} v_{y_1} \geq -2 \sum_{y=0}^{n} t_{ay} v_{y_2} \iff t_{a_0} + \sum_{y=1}^{n} t_{ay} v_{y_1} \leq \sum_{y=1}^{n} t_{ay} v_{y_2}, \\
& \quad \text{where we apply that } v_{01} = 1 \text{ and } v_{02} = 0. \text{ By substituting for } t_{ij} \text{ from (2) this equation can be expressed as}
\end{align*}
\]

\[
\begin{align*}
\frac{1}{4} (GR(a) - 1) + \sum_{y=1}^{n} \frac{1}{4} A_{ay} v_{y_1} & \leq \sum_{y=1}^{n} \frac{1}{4} A_{ay} v_{y_2},
\end{align*}
\]

where \( GR(a) \) denotes the degree of the vertex \( a \) in the graph \( \bar{G} \).

Since \( GR(a) = \mathcal{C}(S_a) + \mathcal{C}(Q_a) \), then we have

\[
(\mathcal{C}(S_a) + \mathcal{C}(Q_a) - 1) + \mathcal{C}(S_a) \leq \mathcal{C}(Q_a) \Rightarrow \mathcal{C}(S_a) \leq \frac{1}{2}.
\]

However, \( \mathcal{C}(S_a) \geq 1 \) and then a contradiction is reached. Hence, it has been proved that \( V^+ \) is a clique. Next, we shall show that this clique is maximal.

Let us suppose that the clique \( V^+ \) is not maximal. Hence, there exists a vertex \( b \in V^- \) that is connected to all the vertices in \( V^+ \). If we define

\[
S_b = \{i/i \in V^+, i \neq b, A_{ib} = 1\}; \quad Q_b = \{i/i \in V^-, i \neq b, A_{ib} = 1\}
\]

then \( \mathcal{C}(S_b) = 0 \). Because the network is in an equilibrium state and \( b \in V^- \)

\[
\begin{align*}
 u_{b_1} \leq u_{b_2} & \iff -2 \sum_{y=0}^{n} t_{by} v_{y_1} \leq -2 \sum_{y=0}^{n} t_{by} v_{y_2} \iff t_{b_0} + \sum_{y=1}^{n} t_{by} v_{y_1} \leq \sum_{y=1}^{n} t_{by} v_{y_2},
\end{align*}
\]
By substituting for $t_{ij}$ from (2) this equation can be expressed as
\[
\frac{1}{4}(GR(b) - 1) + \sum_{y=1}^{n} \frac{1}{4} A_{ay}v_{y1} \geq \sum_{y=1}^{n} \frac{1}{4} A_{ay}v_{y2}
\]
\[\Rightarrow (\mathcal{C}(S_b) + \mathcal{C}(Q_b) - 1) + \mathcal{C}(S_b) \geq \mathcal{C}(Q_b) \Rightarrow \mathcal{C}(S_b) \geq \frac{1}{2}.
\]

However, $\mathcal{C}(S_b) = 0$ and then a contradiction is reached. Hence, it has been proved that the clique $V^+$ is maximal.

Based on the OCHOM network, the following procedure describes the proposed algorithm for solving the MCP.

1. Set $k = 0$ and $v_{01} = 1, v_{02} = 0$.
2. Assign the initial values of $v_{xi}$, for $x = 1$ to $n$, and $i = 1$ to $2$, by randomly setting the output of one neuron in each row to be 1 and the other neuron in the row to be 0.
3. Select a row $x \in \{1, 2, \ldots, n\}$.
4. Compute $u_{xi}(k)$, for $i = 1$ to 2: $u_{xi}(k) = -2 \sum_{y=0}^{n} t_{xy}v_{yi}(k)$.
5. Evaluate the values of $v_{xi}$, for $i = 1$ to 2
\[v_{xi}(k + 1) = 1 \text{ if } u_{xi}(k) = \max\{u_{x1}(k), u_{x2}(k)\}, \quad 0 \text{ otherwise}.
\]
6. Increment $k$ by 1.
7. If the state of the network reaches an equilibrium state then stop, else go to step 3.

In step 3, we can choose the row $x$ in different systematic fashions, generating different OCHOM neural networks. In the C-OCHOM, we consider $x = k \mod n + 1$. In the R-OCHOM, it is selected as a random number $x \in \{1, 2, \ldots, n\}$. In the B-OCHOM, step 3 can be defined as follows:

3.1. Compute $u_{xi}(k)$ for $x = 1$ to $n$, and $i = 1$ to 2.
3.2. Compute $\Delta E_x(k) = u_{xo}(k) - u_{xc}(k)$, for $x = 1$ to $n$, where $xo$ is the activated neuron in row $x$ and $xc$ is the other neuron in row $x$.
3.3. Select the row $x$ with the minimum value of $\Delta E_x(k)$.

Note that, in step 5, if the two neurons in row $x$ have the same value of $u$, then the algorithm must randomly select one of them. However, for simplicity, it selects the first neuron in the row with the maximum value of $u$, as it is also defined in the parallel maximum network.

Observe that one of the main differences between the maximum network and our OCHOM network is that we consider the input of the neuron $xi$ computed by the original Hopfield’s discrete updating rule (6) instead of (4). As Wang [30] pointed out, the incorrect updating rule (4) applied by Takefuji and Lee, that provides $\Delta u_{xi}$ instead of $u_{xi}$ itself, is computationally inefficient since a neuron may not be able to update its state until it has accumulated enough input during several evaluations of (4). For this reason, our extensive simulation runs show that the OCHOM in the MCP is computationally superior to the parallel maximum neural network.
Table 1
Simulation results for $n = 10$

<table>
<thead>
<tr>
<th>Density</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Percentage of error</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>16</td>
<td>10</td>
<td>12</td>
<td>10</td>
<td>11</td>
<td>15</td>
<td>8</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>C-OCHOM</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B-OCHOM</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Average clique size</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>1.81</td>
<td>2.10</td>
<td>2.51</td>
<td>2.76</td>
<td>3.03</td>
<td>3.80</td>
<td>4.53</td>
<td>5.23</td>
<td>6.97</td>
</tr>
<tr>
<td>C-OCHOM</td>
<td>1.57</td>
<td>1.99</td>
<td>2.43</td>
<td>2.63</td>
<td>3.16</td>
<td>3.88</td>
<td>4.32</td>
<td>5.22</td>
<td>7.02</td>
</tr>
<tr>
<td>B-OCHOM</td>
<td>1.75</td>
<td>2.11</td>
<td>2.68</td>
<td>2.86</td>
<td>3.22</td>
<td>4.06</td>
<td>4.61</td>
<td>5.34</td>
<td>7.06</td>
</tr>
<tr>
<td><strong>Standard deviation of the clique size</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>0.460</td>
<td>0.451</td>
<td>0.802</td>
<td>0.676</td>
<td>0.730</td>
<td>0.815</td>
<td>0.943</td>
<td>0.732</td>
<td>0.831</td>
</tr>
<tr>
<td>C-OCHOM</td>
<td>0.498</td>
<td>0.541</td>
<td>0.655</td>
<td>0.614</td>
<td>0.598</td>
<td>0.902</td>
<td>0.863</td>
<td>0.746</td>
<td>0.724</td>
</tr>
<tr>
<td>B-OCHOM</td>
<td>0.435</td>
<td>0.490</td>
<td>0.695</td>
<td>0.603</td>
<td>0.644</td>
<td>0.736</td>
<td>0.827</td>
<td>0.590</td>
<td>0.664</td>
</tr>
<tr>
<td><strong>Average computation time (ms)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>0.447</td>
<td>0.361</td>
<td>0.499</td>
<td>0.463</td>
<td>0.511</td>
<td>0.620</td>
<td>0.400</td>
<td>1.033</td>
<td>0.352</td>
</tr>
<tr>
<td>C-OCHOM</td>
<td>0.011</td>
<td>0.012</td>
<td>0.006</td>
<td>0.007</td>
<td>0.009</td>
<td>0.006</td>
<td>0.011</td>
<td>0.030</td>
<td>0.006</td>
</tr>
<tr>
<td>B-OCHOM</td>
<td>0.001</td>
<td>0.003</td>
<td>0.001</td>
<td>0.001</td>
<td>0.006</td>
<td>0.003</td>
<td>0.001</td>
<td>0.001</td>
<td>0.006</td>
</tr>
</tbody>
</table>

6. Simulation results

In this section, we illustrate the performance of the different OCHOM networks in the MCP on random graphs of various vertex sizes and densities. For comparison, the parallel maximum neural network proposed by Lee and Takefuji for this problem [18] has also been implemented. Computational experiments were performed on an AMD-ATHLON XP 1500 MHz with 256 MByte SDRAM by MATLAB (The Mathworks, Inc.). Twenty graph sizes from $n = 10$ to 1000 with nine graph densities in each size from 0.1 to 0.9 were simulated, where the density of an $n$-vertex graph is defined by the ratio between the number of given edges and $n(n - 1)/2$.

Tables 1–5 present the simulation results for the sizes $n = 10, 50, 100, 500$ and 1000. For the other 15 sizes that are not presented in the tables, the conclusions about solution quality and computation time are very similar. In each size and density, 10 different randomly generated graphs were solved, and for every graph we carried out 10 simulation runs from different randomly generated initial states.

In order to consider the same initial states, for both the OCHOM and the maximum network, we take them as suggested by Lee and Takefuji, that is, by computing the initial inputs of the neurons as a random number $u_{ij} \in (-1, 1)$. Since the parallel maximum network sometimes oscillates and does not reach an equilibrium state, a termination condition is needed in order to finalize this algorithm. Then, a maximum number of 1000 network updates is imposed if the equilibrium state is not reached. This condition is not needed in the OCHOM network whose convergence is guaranteed.

The simulation results for the R-OCHOM and C-OCHOM networks show that the solution quality is very similar in both models. However, the computation time is increased in the R-OCHOM due...
## Table 2
Simulation results for $n = 50$

<table>
<thead>
<tr>
<th>Density</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Percentage of error</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>10</td>
<td>19</td>
<td>15</td>
<td>20</td>
<td>15</td>
<td>23</td>
<td>20</td>
<td>32</td>
<td>28</td>
</tr>
<tr>
<td>C-OCHOM</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B-OCHOM</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

| **Average clique size** |     |     |     |     |     |     |     |     |     |
| MAXIMUM | 2.38 | 3.07 | 3.73 | 4.59 | 5.58 | 6.87 | 8.69 | 11.41 | 18.15 |
| C-OCHOM | 2.35 | 3.09 | 3.73 | 4.65 | 5.25 | 6.57 | 8.55 | 11.41 | 18.41 |
| B-OCHOM | 2.76 | 3.49 | 4.15 | 4.91 | 6.05 | 7.21 | 9.15 | 11.86 | 18.82 |

| **Standard deviation of the clique size** |     |     |     |     |     |     |     |     |     |
| MAXIMUM | 0.758 | 0.787 | 1.016 | 0.924 | 1.117 | 1.092 | 1.428 | 1.567 | 1.589 |
| C-OCHOM | 0.479 | 0.726 | 0.815 | 0.539 | 0.821 | 0.807 | 1.067 | 1.164 | 1.436 |
| B-OCHOM | 0.534 | 0.594 | 0.657 | 0.653 | 0.903 | 0.913 | 1.242 | 1.247 | 1.250 |

| **Average computation time (s)** |     |     |     |     |     |     |     |     |     |
| MAXIMUM | 0.333 | 0.427 | 0.477 | 0.469 | 0.405 | 0.505 | 0.564 | 0.757 | 0.704 |
| C-OCHOM | 0.007 | 0.006 | 0.007 | 0.005 | 0.005 | 0.005 | 0.006 | 0.006 | 0.005 |
| B-OCHOM | 0.003 | 0.004 | 0.004 | 0.005 | 0.005 | 0.005 | 0.004 | 0.004 | 0.004 |

## Table 3
Simulation results for $n = 100$

<table>
<thead>
<tr>
<th>Density</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Percentage of error</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>15</td>
<td>13</td>
<td>18</td>
<td>16</td>
<td>19</td>
<td>15</td>
<td>24</td>
<td>39</td>
<td>49</td>
</tr>
<tr>
<td>C-OCHOM</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B-OCHOM</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

| **Average clique size** |     |     |     |     |     |     |     |     |     |
| MAXIMUM | 2.64 | 3.47 | 4.40 | 5.21 | 6.36 | 8.22 | 10.72 | 14.87 | 24.45 |
| B-OCHOM | 3.04 | 3.85 | 4.85 | 5.85 | 7.20 | 8.80 | 11.49 | 15.76 | 25.04 |

| **Standard deviation of the clique size** |     |     |     |     |     |     |     |     |     |
| MAXIMUM | 0.784 | 0.805 | 0.873 | 1.098 | 1.228 | 1.310 | 1.466 | 1.668 | 2.802 |
| C-OCHOM | 0.532 | 0.500 | 0.615 | 0.700 | 0.647 | 0.928 | 1.159 | 1.390 | 1.983 |
| B-OCHOM | 0.424 | 0.411 | 0.592 | 0.730 | 0.816 | 0.964 | 1.105 | 1.304 | 1.831 |

| **Average computation time (s)** |     |     |     |     |     |     |     |     |     |
| MAXIMUM | 1.37 | 1.61 | 1.36 | 1.56 | 1.66 | 1.64 | 2.01 | 2.43 | 2.78 |
| C-OCHOM | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 |
| B-OCHOM | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.03 | 0.03 | 0.03 |
### Table 4
Simulation results for $n = 500$

<table>
<thead>
<tr>
<th>Density</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Percentage of error</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>12</td>
<td>17</td>
<td>10</td>
<td>17</td>
<td>19</td>
<td>18</td>
<td>43</td>
<td>39</td>
<td>55</td>
</tr>
<tr>
<td>C-OCHOM</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B-OCHOM</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Average clique size</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>3.25</td>
<td>4.40</td>
<td>5.72</td>
<td>6.90</td>
<td>8.95</td>
<td>11.70</td>
<td>15.21</td>
<td>22.44</td>
<td>39.44</td>
</tr>
<tr>
<td>C-OCHOM</td>
<td>3.20</td>
<td>4.55</td>
<td>5.48</td>
<td>6.83</td>
<td>8.82</td>
<td>11.33</td>
<td>14.89</td>
<td>21.40</td>
<td>37.92</td>
</tr>
<tr>
<td>B-OCHOM</td>
<td>3.63</td>
<td>4.93</td>
<td>6.13</td>
<td>7.64</td>
<td>9.64</td>
<td>12.14</td>
<td>16.23</td>
<td>23.67</td>
<td>42.28</td>
</tr>
<tr>
<td><strong>Standard deviation of the clique size</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>0.848</td>
<td>0.957</td>
<td>0.861</td>
<td>1.236</td>
<td>1.036</td>
<td>1.162</td>
<td>1.623</td>
<td>1.979</td>
<td>2.988</td>
</tr>
<tr>
<td>C-OCHOM</td>
<td>0.402</td>
<td>0.500</td>
<td>0.772</td>
<td>0.805</td>
<td>0.957</td>
<td>1.026</td>
<td>1.222</td>
<td>1.470</td>
<td>2.130</td>
</tr>
<tr>
<td>B-OCHOM</td>
<td>0.544</td>
<td>0.573</td>
<td>0.706</td>
<td>0.798</td>
<td>0.835</td>
<td>0.932</td>
<td>1.033</td>
<td>1.583</td>
<td>2.015</td>
</tr>
<tr>
<td><strong>Average computation time (s)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>41.60</td>
<td>39.42</td>
<td>32.14</td>
<td>37.96</td>
<td>35.89</td>
<td>31.35</td>
<td>47.11</td>
<td>46.05</td>
<td>58.25</td>
</tr>
<tr>
<td>C-OCHOM</td>
<td>0.33</td>
<td>0.18</td>
<td>0.22</td>
<td>0.18</td>
<td>0.17</td>
<td>0.22</td>
<td>0.18</td>
<td>0.18</td>
<td>0.21</td>
</tr>
<tr>
<td>B-OCHOM</td>
<td>6.96</td>
<td>6.78</td>
<td>6.81</td>
<td>6.82</td>
<td>6.75</td>
<td>6.75</td>
<td>6.60</td>
<td>6.45</td>
<td>6.10</td>
</tr>
</tbody>
</table>

### Table 5
Simulation results for $n = 1000$

<table>
<thead>
<tr>
<th>Density</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Percentage of error</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>10</td>
<td>12</td>
<td>11</td>
<td>14</td>
<td>15</td>
<td>26</td>
<td>29</td>
<td>47</td>
<td>62</td>
</tr>
<tr>
<td>C-OCHOM</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B-OCHOM</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Average clique size</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>3.57</td>
<td>5.11</td>
<td>6.33</td>
<td>7.91</td>
<td>10.11</td>
<td>12.82</td>
<td>17.30</td>
<td>25.35</td>
<td>46.84</td>
</tr>
<tr>
<td>C-OCHOM</td>
<td>3.51</td>
<td>4.71</td>
<td>5.88</td>
<td>7.67</td>
<td>9.85</td>
<td>12.71</td>
<td>16.70</td>
<td>24.44</td>
<td>44.95</td>
</tr>
<tr>
<td>B-OCHOM</td>
<td>4.06</td>
<td>5.30</td>
<td>6.67</td>
<td>8.52</td>
<td>10.56</td>
<td>13.52</td>
<td>18.43</td>
<td>26.80</td>
<td>49.43</td>
</tr>
<tr>
<td><strong>Standard deviation of the clique size</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>0.750</td>
<td>0.632</td>
<td>0.938</td>
<td>1.025</td>
<td>0.951</td>
<td>1.114</td>
<td>1.719</td>
<td>1.952</td>
<td>2.126</td>
</tr>
<tr>
<td>C-OCHOM</td>
<td>0.522</td>
<td>0.574</td>
<td>0.756</td>
<td>0.604</td>
<td>0.845</td>
<td>0.977</td>
<td>1.202</td>
<td>1.459</td>
<td>2.380</td>
</tr>
<tr>
<td>B-OCHOM</td>
<td>0.487</td>
<td>0.627</td>
<td>0.682</td>
<td>0.759</td>
<td>0.808</td>
<td>0.990</td>
<td>1.200</td>
<td>1.348</td>
<td>2.109</td>
</tr>
<tr>
<td><strong>Average computation time (s)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>240.1</td>
<td>216.5</td>
<td>198.4</td>
<td>187.6</td>
<td>162.3</td>
<td>173.9</td>
<td>166.1</td>
<td>197.1</td>
<td>233.1</td>
</tr>
<tr>
<td>C-OCHOM</td>
<td>0.87</td>
<td>0.81</td>
<td>0.92</td>
<td>1.01</td>
<td>1.01</td>
<td>1.01</td>
<td>1.01</td>
<td>1.02</td>
<td>1.04</td>
</tr>
<tr>
<td>B-OCHOM</td>
<td>55.58</td>
<td>55.65</td>
<td>55.46</td>
<td>55.16</td>
<td>55.13</td>
<td>55.18</td>
<td>54.61</td>
<td>53.40</td>
<td>51.89</td>
</tr>
</tbody>
</table>
to the implementation of the function that selects a row randomly on each step. For that reason, we present in Tables 1–5 only the simulation results for the C-OCHOM, B-OCHOM and for the parallel maximum network.

In each table, we first represent %Error, that is, the percentage of random initial states that do not converge to a clique of the graph. Obviously, this percentage is always 0% in all the OCHOM networks since they are designed to converge to a maximal clique. However, many times (from 8% to 62%) the parallel maximum network oscillates indefinitely between states not corresponding to cliques. Except for the very small size $n = 10$, it is observed that this percentage increases with the density in every size. This percentage is also very high in the very small density 0.1 and increases with the size of the graph. Hence, this network usually degrades the performance for very large size instances.

Secondly, we present in each table the average maximal clique sizes found in every method. Since the maximum network sometimes does not converge to maximal cliques, we can take into account, for this network, only the simulation runs that provide correct solutions. Then, it should be noted that when we compute the average clique size and the standard deviation for the maximum network, the sample size is always $(100 - \%Error)$. However, for the OCHOM networks, the sample size is always 100 for each size and density. It is observed that the solution quality is very similar in the C-OCHOM and in the maximum network for every size and density. However, the B-OCHOM network always provides better solution qualities than the C-OCHOM and the maximum network for each size and density. Thus, it is shown that the two-level competitive neural scheme of the B-OCHOM generates not only the fastest, but also the most accurate solutions in the MCP.

Next, we present in each table the standard deviation of the clique size to provide more detailed statistical data about the behaviors of the methods. It is observed that this standard deviation is always larger in the maximum network. It indicates the diversity of the solution quality obtained by this network, even when we consider only the runs that converge to maximal cliques. Therefore, in the maximum network, the solution quality must be improved by repeating the trials with different initial states. However, the B-OCHOM varies little from its mean, and then usually finds a better result when it is run only once.

Finally, the average computation time in seconds taken by every network to converge is presented in each table. It is observed that the OCHOM networks are much superior to the maximum network in terms of the computation time on a usual sequential machine such as a PC. For instance, for $n = 500$ and density 0.9 the C-OCHOM network is 277 times faster than the maximum network. If we now compare the computation times in the B-OCHOM and in the C-OCHOM networks, it can be observed that the C-OCHOM is usually faster for the larger sizes $n \geq 100$. This is due to the computation time required to implement the function that selects the best row in each step in the B-OCHOM. However, note that the B-OCHOM network always reaches the best solution quality.

To conclude, in order to evaluate not just the relative, but also the absolute performance quality, a comparison with the exact solution is presented for the smaller problems. Observe that for medium and larger sizes it requires prohibitively a long computation time to find the exact optimal clique. In these simulations, we generated random graphs of $n = 10$ and 20 vertices, with nine graph densities, from 0.1 to 0.9. In each size and density, 100 randomly generated graphs were solved, and for every graph we carried out 10 simulation runs from different
randomly generated initial states. Then, as the algorithms are run 1000 times in each size and density, our comparisons with the exact optimal clique report a total of 18,000 solutions for each method.

Tables 6 and 7 summarize the simulation results. First, we present the percentage of successes, that is, the probability to find the exact optimal solution when the algorithms are run only once. Secondly, Tables 6 and 7 present the average error in the clique quality. We define the error in the clique quality as the percentage of the size that is smaller than the size of the maximum clique. Observe that in each size and density, the best proportion of successes and the minor error in the clique quality is always obtained by the B-OCHOM network.

Tables 6 and 7 also compare maximum and OCHOM networks from another point of view. We now consider the averages computed for the same test sets but taking into account only the best results obtained by ten runs of the algorithms on each graph with randomly chosen initial states. Considering the best solution among many others is a natural performance improvement technique, used frequently for such local search methods. We present for each algorithm the average error in the clique quality and the probability to find the maximum clique by ten independent runs. Observe that the best results in each size and density are obtained by the B-OCHOM network. It should be noted that, although for all the algorithms the probability to find the maximum clique by ten runs is closer to 100%, ten runs of the maximum network may require a long computation time for the larger sizes. On the other hand, one run of the OCHOM for the bigger sizes $n \geq 1000$ takes only seconds even on a conventional PC. Therefore, we can find good quality maximum or near-maximum cliques in a reasonable computation time.
Table 7
Comparison with the exact solution for \( n = 20 \)

<table>
<thead>
<tr>
<th>Density</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Percentage of successes</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>42.7</td>
<td>41.4</td>
<td>28.7</td>
<td>32.9</td>
<td>28.2</td>
<td>20.2</td>
<td>20.4</td>
<td>21.6</td>
<td>27.9</td>
</tr>
<tr>
<td>C-OCHOM</td>
<td>48.0</td>
<td>34.3</td>
<td>23.3</td>
<td>30.7</td>
<td>21.7</td>
<td>22.6</td>
<td>22.1</td>
<td>22.4</td>
<td>30.9</td>
</tr>
<tr>
<td>B-OCHOM</td>
<td>59.1</td>
<td>54.4</td>
<td>36.9</td>
<td>42.5</td>
<td>37.0</td>
<td>30.6</td>
<td>28.8</td>
<td>30.0</td>
<td>37.8</td>
</tr>
<tr>
<td><strong>Percentage of error in clique quality</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-OCHOM</td>
<td>23.31</td>
<td>23.87</td>
<td>24.80</td>
<td>22.13</td>
<td>22.46</td>
<td>19.21</td>
<td>17.36</td>
<td>14.58</td>
<td>9.52</td>
</tr>
<tr>
<td>B-OCHOM</td>
<td>16.03</td>
<td>16.10</td>
<td>18.33</td>
<td>16.02</td>
<td>15.73</td>
<td>15.44</td>
<td>14.60</td>
<td>11.64</td>
<td>7.86</td>
</tr>
<tr>
<td><strong>Percentage f successes (by 10 runs)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>91</td>
<td>94</td>
<td>90</td>
<td>92</td>
<td>90</td>
<td>77</td>
<td>80</td>
<td>86</td>
<td>88</td>
</tr>
<tr>
<td>C-OCHOM</td>
<td>67</td>
<td>73</td>
<td>61</td>
<td>72</td>
<td>63</td>
<td>63</td>
<td>67</td>
<td>80</td>
<td>83</td>
</tr>
<tr>
<td>B-OCHOM</td>
<td>98</td>
<td>96</td>
<td>91</td>
<td>92</td>
<td>90</td>
<td>87</td>
<td>89</td>
<td>88</td>
<td>95</td>
</tr>
<tr>
<td><strong>Percentage of error in clique quality (by 10 runs)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>3.50</td>
<td>1.90</td>
<td>2.53</td>
<td>1.78</td>
<td>1.89</td>
<td>3.81</td>
<td>2.65</td>
<td>1.52</td>
<td>1.01</td>
</tr>
<tr>
<td>C-OCHOM</td>
<td>12.84</td>
<td>9.21</td>
<td>10.10</td>
<td>6.89</td>
<td>7.56</td>
<td>6.67</td>
<td>4.64</td>
<td>2.17</td>
<td>1.68</td>
</tr>
<tr>
<td>B-OCHOM</td>
<td>0.78</td>
<td>1.27</td>
<td>2.27</td>
<td>1.78</td>
<td>1.89</td>
<td>2.06</td>
<td>1.46</td>
<td>1.30</td>
<td>0.42</td>
</tr>
</tbody>
</table>

7. Conclusions

We have presented new neural network algorithms for solving the MCP based on the Optimal Competitive Hopfield Model. Three different competitive neural schemes have been proposed, showing that the two-level winner-take-all structure of the B-OCHOM network is the most accurate neural network strategy in this problem.

The neural network algorithm for the MCP based on the parallel maximum neural network has also been implemented. Our simulation results show that the B-OCHOM is superior to the maximum network in terms of the solution quality for all the sizes and densities. Experimental results also show that many times (from 8% to 62%) the maximum neural network does not converge to a clique of the graph, where this percentage increases with the size of the graph. Hence, this network degrades the performance for large size instances. In contrast, the OCHOM networks provide accurate solutions even for large-scale problems. Moreover, the OCHOM networks are always faster than the maximum network on a conventional sequential machine, where this proportion is increased for \( n \geq 100 \).

However, observe that the computation mode in the maximum network can be implemented on a parallel machine with \( n \) processors (where \( n \) is the number of vertices of the graph), considerably decreasing the computation times. The computation mode in the B-OCHOM is also suitable for parallel processing of the inputs of the \( n \) rows, in order to select the best row in each step, also considerably decreasing the computation times. However, this is not possible in the C-OCHOM, where only a group of two neurons is considered each time. We conclude that, while most of the existing algorithms require expensive machines, our algorithm is faster on an inexpensive sequential machine such as a personal computer. Hence, very large-scale problems can certainly be solved on a
reasonable computation time using more sophisticated machines. Moreover, the B-OCHOM algorithm is very simple to implement and guarantees good quality maximum or near-maximum cliques of the considered graph.

References


Gloria Galán-Marín received the B.S. degree in Industrial Engineering from the University of Sevilla, Spain, and the Ph.D. degree in Industrial Engineering from the University of Málaga, Spain, she is an Associate Professor at the University of Extremadura, Spain, in the Department of Electronics and Electromechanical Engineering. Her research interests include neural net computing, combinatorial optimization and applications of new techniques in industry and business problems.

Enrique Mérida-Casermeiro received the B.S. degree in Mathematics, and the Ph.D. degree in Computer Sciences from the University of Málaga, Spain, since 1990 he is a Professor at the same University in the Department of Applied Mathematics. His current research interests include neural networks, pattern recognition and combinatorial optimization.

José Muñoz-Pérez received the B.S. degree in Mathematics from the University of Granada, Spain, and the Ph.D. degree in Mathematics from the University of Sevilla, Spain, since 1989 he is a Professor at the University of Málaga in the Department of Languages and Computation Sciences in the Area of Computing and Artificial Intelligence. He is also the director of the Image Processing Institute in the Technological Park of Andalucia. His current research interests include neural networks, pattern recognition and image processing. He is a member of the Spanish Association for Artificial Intelligence.