

SOLVING MAXIMUM INDEPENDENT SET BY ASYNCHRONOUS DISTRIBUTED HOPFIELD-TYPE NEURAL NETWORKS

GIULIANO GROSSI¹, MASSIMO MARCHI¹
AND ROBERTO POSENATO²

Abstract. We propose a heuristic for solving the maximum independent set problem for a set of processors in a network with arbitrary topology. We assume an asynchronous model of computation and we use modified Hopfield neural networks to find high quality solutions. We analyze the algorithm in terms of the number of rounds necessary to find admissible solutions both in the worst case (theoretical analysis) and in the average case (experimental Analysis). We show that our heuristic is better than the greedy one at 1% significance level.

Mathematics Subject Classification. 68W15, 90C59, 05C69.

1. INTRODUCTION

The Maximum Independent Set problem (MIS) requires finding the largest independent set in a graph G , *i.e.* the maximum subset of vertices of G such that no two vertices are joined by an edge. This is one of the first problems shown to be NP-hard [17] and, according to the result of Feige *et al.* [7] for the Maximum Clique problem (the same problem as MIS on the complementary graph), even approximating MIS within a constant factor is NP-hard. In particular, if

Keywords and phrases. Max independent set, hopfield networks, asynchronous distributed algorithms.

¹ Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, via Comelico 39, 20135 Milano, Italy; grossi@dsi.unimi.it

² Dipartimento di Informatica, Università degli Studi di Verona, strada le grazie 15, 37134 Verona, Italy.

$\text{NP} \neq \text{ZPP}^1$, no polynomial time algorithm can approximate MIS within a factor $n^{1-\varepsilon}$ for any $\varepsilon > 0$, where n is the number of vertices of the graph [13].

This problem is relevant for many theoretical research areas and practical applications. For instance, the clustering problem for peer-to-peer mobile wireless networks can be easily reduced to the problem of finding a maximum independent set of nodes in the network graph [9].

In this paper we propose an *asynchronous* distributed algorithm for the MIS problem, synchronous variant of which has been discussed in [11]. The original method from which we derive both distributed algorithms is based on a finite sequence of discrete-time Hopfield networks [14], the attractor sequence of which converges to an admissible locally optimal solution [4]. The networks generated by the algorithm are progressively determined by a simple and efficient weights updating rule which controls the relaxation process in order to allow the system to escape from not admissible local minima.

This method always ensures the finding of an admissible solution; moreover, the quality of the solution is high as shown in the simulation results of previous works and ratified here (*i.e.*, in the distributed case). Since Hopfield neural networks do not guarantee that an admissible solution to constrained optimization problems will be found, many researches have been focused on improving the original model in order to obtain a high percentage of admissible high quality solutions. One of the most relevant works has been done by Li [19], who proposes a new method to solve constrained optimization problems combining augmented Lagrange multipliers and Hopfield networks. To corroborate the fact that our algorithm does not suffer the dilemma between quality and convergence, we experimentally show that our weights updating rule is more effective in finding good solutions than the Li's method.

We refer to an asynchronous model of computation in which the topology of the network of processors is described by an arbitrary graph. We do not make any assumptions except for the following: the time is discrete, the message delivering and receiving is done within a fixed time and the time taken for local computations is considered to be negligible. We define the *time complexity* in terms of the rounds needed by the system to find a solution. In every round a processor is allowed to send messages to the processors linked with it. We determine the time complexity both in the worst case (theoretical) and in the average case (experimental).

To test the solution quality of our algorithm, we experimentally compare it with a very good probabilistic heuristic derived by the so called Ant Colony Optimization technique [18], with the standard greedy algorithm, widely used in distributed applications, and with a very fast self-stabilizing algorithm for asynchronous distributed system [15].

The solution quality found by our algorithm on various randomly generated instances shows it has on average better performances than the other algorithms mentioned here at 1% significance level.

¹The class of problems $\text{RP} \cap \text{coRP}$, denoted by ZPP, is the class which admits polynomial-time randomized algorithms with zero probability of error (Las Vegas algorithms) [21].

2. RELATED WORKS

With reference to the MIS problem, several heuristic techniques have been developed in recent years and most of them have a good ability to find good approximate solutions for the problem. These techniques include simulated annealing [5], neural networks [4], tabu search [3], greedy randomized adaptive search (GRASP) [8], genetic and evolutionary algorithms [1, 20] and, more recently, Ant Colony Optimization [18]. All these techniques allow us to find approximate solutions to the problem in polynomial-time.

In the field of distributed algorithms, many heuristics have been developed to solve a related problem called **Maximal Independent Set** problem that consists in finding a maximal independent set but not the largest. This problem is solvable in polynomial time by a very simple greedy algorithm². The interest for **Maximal Independent Set** problem is due the fact that in many applications it is more important to find a maximal independent set quickly than to find the largest one. In [15], the authors present a fault-containing self-stabilizing algorithm for the **Maximal Independent Set** problem in asynchronous distributed systems and they claim that it is the most efficient algorithm known at the publication time. In [10] a fast synchronous algorithm is shown for ad hoc networks of constant dimension but with variable topology. Finally, in [2] a greedy distributed algorithm for the efficient determination of a maximal weighted independent set for wireless network is proposed and analyzed.

3. PRELIMINARIES

We consider arbitrary undirected graphs $G = \langle V, E \rangle$, where $V = \{1, \dots, n\}$ is the set of vertices and $E \subseteq V \oplus V$ (not ordered pairs) is the set of edges $e = \{i, j\}$. For graph G , a subset $S \subseteq V$ of vertices is an *independent set* if and only if every pair of distinct vertices are not joined by an edge in E . An independent set is *maximal* if it is not a subset of another independent set.

The MIS problem is a maximization problem which consists in finding a maximum independent set (among the maximals) of a given graph G . Formally

Max Independent Set (MIS)

INSTANCE: Graph $G = \langle V, E \rangle$.

SOLUTION: An independent set of G , *i.e.*, a subset $S \subseteq V$ such that no two distinct vertices in S are joined by an edge in E .

MEASURE: Cardinality of the independent set, *i.e.*, $|S|$.

GOAL: MAX.

Let $A = (a_{ij})$ denote the *adjacency matrix* of G and d_i the degree of the vertex i . Unlike many formulations introduced for MIS problem, we reduce it to an integer

²The algorithm starts with an empty set V . Then it searches for a vertex v that is not connected to any vertex in V and if such v is found, adds v to V . The algorithm stops when it cannot find v not connected to any vertex in V . This results in an independent set that is not contained in any larger independent set.

linear programming problem subject to quadratic constraints. To this end, let $S \subseteq \{1, \dots, n\}$ and let (x_1, \dots, x_n) be the characteristic vector of S . Then $\sum_{i < j} a_{ij} x_i x_j$ represents the number of edges in E with end-points in S . It is easy to show that S is an independent set if and only if $\sum_{i < j} a_{ij} x_i x_j = 0$.

Based on this remark, the MIS problem can be expressed as:

$$\begin{aligned} \text{maximize} \quad & \Psi_G(\mathbf{x}) = \alpha \sum_{i=1}^n x_i \\ \text{subject to} \quad & \Omega_G(\mathbf{x}) = \sum_{\{i,j\} \in E} x_i x_j = 0 \\ & \mathbf{x} \in \{0, 1\}^n, \end{aligned} \tag{P}$$

where $\alpha \geq 1$ is an integer constant. The role of α is extremely important because on one hand it preserves the local optima of the equivalent and more natural program (that with $\alpha = 1$), whereas on the other hand it strengthens the ability of the neural heuristic to find better solution on average.

From now on, we will focus on the function $\Phi : \{0, 1\}^n \rightarrow \mathbf{N}^+$ defined as:

$$\Phi(\mathbf{x}) = \text{"\# vertices"} - \text{"\# unsatisfied constraints"} = \Psi_G(\mathbf{x}) - \Omega_G(\mathbf{x}).$$

A property of Φ that we will use in the design of the algorithm concerns the relation between its maximum value and the cost of the optimum solution of the instance, as described in the following proposition.

Proposition 3.1. *Given a graph G and $\mathbf{y} \in \{0, 1\}^n$ such that $\Omega_G(\mathbf{y}) = 0$; \mathbf{y} is an optimum solution for MIS problem if and only if $\Phi(\mathbf{y}) = \sup_{\Omega_G(\mathbf{x})=0} \Phi$.*

4. ISHN: OVERVIEW OF THE ALGORITHM

In this section we present an approximation algorithm for MIS based on the discrete Hopfield networks [14] for which we briefly report the main results.

We denote a Hopfield network \mathcal{R} of n neurons with states in $\{0, 1\}$ by the pair $\mathcal{R} = \langle W, \boldsymbol{\lambda} \rangle$, where $W = (w_{ij})_{n \times n}$ is the (symmetric) weight matrix and $\boldsymbol{\lambda} = (\lambda_i)_{1 \times n}$ the threshold vector; both the matrix and the vector have integer components.

We consider the discrete-time dynamics with *sequential* updating. Let $U_i(t)$ be the state of the neuron i at time t , the dynamics is formally described as follows:

$$U_i(t+1) = \text{HS} \left(\sum_{1 \leq j < i} w_{ij} U_j(t+1) + \sum_{i < j \leq n} w_{ij} U_j(t) - \lambda_i \right) \quad i = 1, \dots, n, \tag{1}$$

where HS is the heavy side step function.

Under these assumptions, the following Lyapunov function, called *energy*, can be associated to every network $\mathcal{R} = \langle W, \lambda \rangle$:

$$\mathcal{E}_{\mathcal{R}}(U_1, \dots, U_n) = -\frac{1}{2} \sum_{i \neq j} w_{ij} U_i U_j + \sum_i \lambda_i U_i.$$

Given an initial condition $\mathbf{U}(0) = \mathbf{U}_0$, equation (1) describes a unique trajectory $\{\mathbf{U}(t)\}_{t \geq 0}$, the last state of which represents a local minimum for the energy function. An upper bound to the length $l_{\mathcal{R}, \mathbf{U}_0}$ of the transient is given in the following theorem.

Theorem 4.1. *The trajectory $\{\mathbf{U}(t)\}_{t \geq 0}$ generated by the Hopfield network $\mathcal{R} = \langle W, \lambda \rangle$ with initial condition \mathbf{U}_0 admits an attractor $\tilde{\mathbf{y}}$, and the length $l_{\mathcal{R}, \mathbf{U}_0}$ of the transient is bounded by*

$$l_{\mathcal{R}, \mathbf{U}_0} \leq \max_{\mathbf{U}} \{\mathcal{E}_{\mathcal{R}}(\mathbf{U})\} - \min_{\mathbf{U}} \{\mathcal{E}_{\mathcal{R}}(\mathbf{U})\}.$$

We denote by $\tilde{\mathbf{y}}$ the attractor of \mathcal{R} initialized by \mathbf{U}_0 .

If equation (1) is modified as

$$U_i(t+1) = \text{HS} \left(- \sum_{1 \leq j < i} w_{ij} U_j(t+1) - \sum_{i < j \leq n} w_{ij} U_j(t) + \lambda_i \right) \quad i = 1, \dots, n, \quad (2)$$

then the trajectory $\{\mathbf{U}(t)\}_{t \geq 0}$ admits as attractor a point of local maximum for the energy \mathcal{E} .

We are now able to present an algorithm, called INDEPENDENT-SET-HOPFIELD-NETS (ISHN), that finds approximate solutions for (P) . This algorithm is based on a sequence of discrete Hopfield networks in which the neurons correspond to the vertices of G .

The sequence $\{\mathcal{R}_k\}_{k \geq 0}$ of Hopfield networks is inductively defined by

- (1) \mathcal{R}_0 is the network with energy function $\Phi_0(\mathbf{y}) = \Psi_G(\mathbf{y}) - \Omega_G(\mathbf{y})$, i.e. $W = A$ and $\lambda = (\alpha, \alpha, \dots, \alpha)$, and $\tilde{\mathbf{y}}^{(0)}$ is the attractor of \mathcal{R}_0 initialized with $(1, \dots, 1)$;
- (2) let $\tilde{\mathbf{y}}^{(k)}$ be the attractor of the network \mathcal{R}_k initialized with $\tilde{\mathbf{y}}^{(k-1)}$ ($k > 0$); \mathcal{R}_k is the network with energy function

$$\Phi_k(\mathbf{y}) = \Phi_{k-1}(\mathbf{y}) - \sum_{\tilde{y}_i^{(k)} = \tilde{y}_j^{(k)} = 1} a_{ij} y_i y_j. \quad (3)$$

For all k , the set $\{\{i, j\} \mid \{i, j\} \in E \text{ and } \tilde{y}_i^{(k)} = \tilde{y}_j^{(k)} = 1\}$ is the set of constraints violated by the attractor $\tilde{\mathbf{y}}^{(k)}$.

The algorithm consists of two alternating phases: one is the evolution of the current network, the other is the weights updating when the current network has reached an attractor. Both phases are repeated as long as there are violated constraints.

ISHN is sketched in Algorithm 1.

Algorithm 1 ISHN

Input: a graph $G = \langle V, E \rangle$, an integer $\alpha > 0$

$\mathcal{R}_0 \leftarrow$ Hopfield net with energy $\Phi_0(\mathbf{y}) = \Psi_G(\mathbf{y}) - \Omega_G(\mathbf{y})$, *i.e.* $W = A$ and

$\boldsymbol{\lambda} = (\alpha, \alpha, \dots, \alpha)$;

$\tilde{\mathbf{y}}^{(0)} \leftarrow$ attractor of \mathcal{R}_0 initialized with $(1, \dots, 1)$

$F_0 \leftarrow \left\{ \{i, j\} \mid \{i, j\} \in E \text{ and } \tilde{y}_i^{(0)} = \tilde{y}_j^{(0)} = 1 \right\}$

$k \leftarrow 0$

while $F_k \neq \emptyset$ **do**

$k \leftarrow k + 1$

$\mathcal{R}_k \leftarrow$ Hopfield net with energy $\Phi_k(\mathbf{y}) = \Phi_{k-1}(\mathbf{y}) - \sum_{\{i,j\} \in F} y_i y_j$

$\tilde{\mathbf{y}}^{(k)} \leftarrow$ attractor of \mathcal{R}_k initialized with $\tilde{\mathbf{y}}^{(k-1)}$

$F_k \leftarrow \left\{ \{i, j\} \mid \{i, j\} \in E \text{ and } \tilde{y}_i^{(k)} = \tilde{y}_j^{(k)} = 1 \right\}$

$S \leftarrow \left\{ i \mid \tilde{y}_i^{(k)} = 1 \right\}$

end while

Output: a maximal independent set S of G

As far as the analysis of ISHN is concerned, the following result states that the algorithm converges, *i.e.*, the sequence of Hopfield networks is finite.

Theorem 4.2. *For every input graph $G = \langle V, E \rangle$, ISHN outputs a maximal independent set of G after $\alpha \cdot |E|$ iterations of the **while** cycle at most.*

Proof. Let $A = (a_{ij})_{n \times n}$ be the adjacency matrix of G , α be a positive integer and

$$\Phi_k(y_1, \dots, y_n) = - \sum_{i < j} w_{ij}^{(k)} y_i y_j + \alpha \sum_i y_i$$

be the energy function of the network \mathcal{R}_k constructed at the k -th step. Note that $(w_{ij}^{(0)})_{n \times n} = (a_{ij})_{n \times n}$, the adjacency matrix of G .

By induction on k it is easy to prove:

- (1) if $\{i, j\} \notin E$ then $w_{ij}^{(k)} = 0$ for all $k = 1, \dots, n$;
- (2) if $\{i, j\} \in E$ then

$$1 = w_{ij}^{(0)} \leq \dots \leq w_{ij}^{(k)} \leq \dots \tag{4}$$

To determine an upper bound to $w_{ij}^{(k)}$, let $\tilde{\mathbf{y}}^{(k)}$ be the attractor of the network \mathcal{R}_k initialized with $\tilde{\mathbf{y}}^{(k-1)}$; for any $i = 1, \dots, n$:

$$\tilde{y}_i^{(k)} = \text{HS} \left(-w_{ij}^{(k)} \tilde{y}_j^{(k)} - \sum_{s \neq j} w_{is}^{(k)} \tilde{y}_s^{(k)} + \alpha \right). \tag{5}$$

Under the hypothesis that $\{i, j\} \in E$ and $\tilde{y}_i^{(k)} = \tilde{y}_j^{(k)} = 1$:

$$-w_{ij}^{(k)} - \sum_{s \neq j} w_{is}^{(k)} \tilde{y}_s^{(k)} + \alpha \geq 0 \tag{6}$$

which implies

$$w_{ij}^{(k)} \leq w_{ij}^{(k)} + \sum_{s \neq j} w_{is}^{(k)} \tilde{y}_s^{(k)} \leq \alpha.$$

So

$$w_{ij}^{(k)} \leq \alpha. \tag{7}$$

and, by the updating rule of the algorithm,

$$w_{ij}^{(k+1)} = w_{ij}^{(k)} + 1. \tag{8}$$

Obviously, the same result holds for j .

Now, $\mathcal{R}_k \neq \mathcal{R}_{k-1}$ if and only if there is $\{i, j\} \in E$ such that $\tilde{y}_i^{(k)} = \tilde{y}_j^{(k)} = 1$. Therefore, by (8) and (7), the number of networks constructed by the algorithm is at most $\alpha \cdot |E|$.

Let us show now that the output S is a maximal independent set. Let $\tilde{\mathbf{y}}^{(h)}$ be the attractor of the last network \mathcal{R}_h , so that $S = \{i \mid \tilde{y}_i^{(h)} = 1\}$ and $\tilde{y}_i^{(h)} = \text{HS}(-\sum_s w_{is}^{(h)} \tilde{y}_s^{(h)} + \alpha)$ for all $i = 1, \dots, n$. Because of the termination condition, we know that

$$\left\{ \{i, j\} \mid \{i, j\} \in E \text{ and } \tilde{y}_i^{(h)} = \tilde{y}_j^{(h)} = 1 \right\} = \emptyset,$$

this implies that S is an independent set of G .

Let us suppose that S is not maximal, *i.e.*, there is $v \notin S$ such that $a_{vs} = 0$ for all $s \in S$. Then:

$$0 = \tilde{y}_v^{(h)} = \text{HS} \left(-\sum_{s \in S} a_{vs} \tilde{y}_s^{(h)} - \sum_{s \notin S} w_{vs}^{(h)} \tilde{y}_s^{(h)} + \alpha \right) = \text{HS}(\alpha) = 1. \quad \square$$

Remark 4.1. In ISHN neural networks are determined by a simple weights updating rule: if two adjacent nodes are in the solution found by a network, the weight of their edge is incremented by 1 in the following network.

It has been shown that the model of Hopfield networks solving optimization problems with constraints can be improved with respect to the quality of the

solutions and to the computation time if one considers the Augmented Lagrangian variant of the Hopfield model [19], where the dilemma between the goal of finding good quality solutions and the convergence necessity is effectively solved. In short, given a problem:

$$\begin{aligned} \min_{\mathbf{x} \in \mathfrak{R}^n} \quad & F(\mathbf{x}) \\ \text{subject to} \quad & c_i(\mathbf{x}) = 0 \quad i = 1, \dots, t \end{aligned} \quad (9)$$

where F and c_i are continuous functions, Augmented Lagrange methods transform this problem into an equivalent one without constraints:

$$\min_{\mathbf{x}, \mathbf{m} \in \mathfrak{R}^n} \quad L(\mathbf{x}, \mathbf{m}, \rho) \quad (10)$$

where L is a Lagrange function with the same local extremum points of F , \mathbf{m} is the vector of Lagrange multipliers and ρ is the penalty factor of the method. For example, a typical Augmented Lagrange quadratic function for the problem (9) is:

$$L_q(\mathbf{x}, \mathbf{m}, \rho) = F(x) - \mathbf{m}^T c(\mathbf{x}) + \frac{\rho}{2} c(\mathbf{x})^T c(\mathbf{x}). \quad (11)$$

To determine an optimal solution, the Augmented Lagrangian Hopfield Networks method sets up an iterative process where, in each phase, a Hopfield network is built to find a solution to the problem (10) where the vector \mathbf{m} is given by an external estimator and the value of ρ is fixed. It has been shown that the solution \mathbf{x} converges to the optimal point \mathbf{x}^* as \mathbf{m} converges to the optimal \mathbf{m}^* . So, it is crucial to have a good multipliers estimator. As an example, a simple multipliers estimator of the first order is given by

$$\mathbf{m}^{(i+1)} = \mathbf{m}^{(i)} - \rho c(\mathbf{x}^{(i)}). \quad (12)$$

To study the Augmented Lagrangian Hopfield Networks method for the MIS problem, we have built an Augmented Lagrangian Hopfield Networks algorithm with an appropriate quadratic Lagrange function and an appropriate linear multipliers estimator (ALHN algorithm) and we have experimentally compared its solutions with ISHN solutions. The instances groups chosen for the test were the same used in [11]. We have repeated the test for different initial values of \mathbf{m} multipliers and for different value of penalty factor ρ .

In all tests, ISHN solutions were better than the solutions calculated by ALHN at 1% significance level, while the computation times of the latter were better by an order of magnitude for almost all instances. For details, see Table 1. The experiments suggest that for the MIS problem, ISHN can be considered more effective than a quadratic Augmented Lagrange Hopfield Networks method with respect to the quality of the solutions found.

Remark 4.2. As we will show in the experimental results section, the sequence of networks obtained by ISHN is a necessary process in order to obtain a good heuristic for the problem at hand. In particular, ISHN performs better than the

standard GREEDY heuristic³, even if it requires more time to find a solution (see Sect. 6). Even so, it can be shown that as the process of network building progresses, the behavior of the networks tends to be that of the greedy. Since the weights sequence tends to the input constant α , *i.e.*, $\lim_{k \rightarrow \infty} w_{ij}^{(k)} = \alpha$ for all $\{i, j\} \in E$, it is easy to prove that the network \mathcal{R} with all the weights $w_{ij} = \alpha$ has really the same behavior as the GREEDY heuristic. To conclude, we can informally write:

$$\lim_{w_{ij}^{(k)} \rightarrow \alpha} \text{ISHN} = \text{GREEDY}.$$

5. DISHN: THE DISTRIBUTED VERSION OF ISHN

The ISHN algorithm presented in the previous section is based on two properties of fundamental importance for the architecture of the distributed system we now introduce:

- (1) the neurons updating (during the evolution phase) depends only on the state of the adjacent neurons;
- (2) the weights updating is done locally by each neuron.

As a consequence, we can directly implement ISHN on a network of processors p_1, \dots, p_n for which we wish to find a maximal independent set of processors (Distributed ISHN for MIS problem). With reference to the *message passing system*, each processor p_k is an independent processing unit which simulates the neuron k , it keeps in memory the state of the neuron, the state of its neighbourhoods $\mathcal{N}(k)$ (*i.e.*, the adjacent processors) and the weights w_s for all $s \in \mathcal{N}(k)$. It sends and receives messages to and from its neighbours on dedicated two-way communication channels.

Moreover, we assume that:

- the time is *discrete* and it is divided into units called TIME-SLICES;
- the message delivering and receiving is done within a TIME-SLICE, while the local computations are instantaneous (negligible time);
- the *topology* of the network is static and it is described by the graph (also called *communication graph*) the nodes of which are the processors and the edges are the communication channels;
- the processors do not refer to a global time, so the system is *asynchronous*.

The main difference between ISHN and DISHN lies in the updating rule of the neurons. In the former case the updating rule is the sequential (asynchronous) dynamics described by (2), while in the latter case it consists of partially synchronous dynamics. In fact, for DISHN we adopt a weaker condition with respect to the dynamics used in ISHN, in which each processor randomly chooses a time-slice

³The GREEDY heuristic sorts the vertices of the graph with respect to their degree, repeatedly picks a nodes in order and places it in the solution if it is not adjacent to any vertex already present.

from which to simulate the neuron activity. From the point of view of the Hopfield model, since there is a positive probability to update linked neurons simultaneously, it is well known that the convergence to an equilibrium state (attractor) may be compromised in favor of an oscillatory behavior. Nonetheless, this risk does not compromise the whole convergence process to a valid solution for the following two reasons.

First, the expected number of “neuron collisions” may be controlled through the width of the temporal window in which to pick out the time-slice for to the neuron updating. Let M be a positive integer representing the number of TIME-SLICES in which each processor picks out (with probability $\frac{1}{M}$) the one for the updating. Since the neuron connections (channels) agree with the set of edges E of the graph describing the network topology, the expected number of collisions is $\frac{|E|}{M}$, which is less than 1 when $|E| < M$. Therefore, by choosing M in order to have no collisions, the expected behavior of DISHN is that of ISHN.

Second, even if the intermediate networks in the sequence $\{\mathcal{R}_k\}_{k \geq 0}$ do not reach an attractor, thanks to the fact that the weights are incrementally updated (see Th. 4.2, Eq. (8)), when they reaches the value α , also under the hypothesis that the units involved are updated synchronously (a very little probability), they will assume state 0, always assuring an admissible solution.

After these preliminary remarks, we now present in more details the DISHN algorithm starting by a general description of the three phases in which it can be logically divided.

Preparatory phase: To optimize the number of messages in the broadcasting operations, we find a *minimum spanning tree* of the communication graph that becomes the actual *communication network* used by the processors to swap messages. The diameter D of the communication network is at most $n - 1$ (the maximum length over all paths).

There are two kinds of messages that can circulate on the communication network: one called CHANGE-STATE, sent by a processor to its neighbourhood when the neuron changes its state; the other, called KEEP-ALIVE, which is broadcast on the communication network when a processor detects a violation of the constraints. The number of KEEP-ALIVE messages broadcast in this case is equal to the diameter of the network. Please note that the messages size is 1 bit.

RUN phase: in this phase each processor simulates the neuron activity until some criterion of stop execution occurs.

WAIT phase: in this phase each processor evaluates if a valid configuration is reached. It waits for a fixed time “listening to” the network to capture both the KEEP-ALIVE and CHANGE-STATE messages; when a message arrives to the processor, the processor immediately switches to the RUN phase, otherwise, after a fixed number of TIME-SLICES (time-out), it terminates the computation because the solution is found. This phase is not necessary in the centralized version since the solution is totally accessible, and therefore totally checkable by the central process.

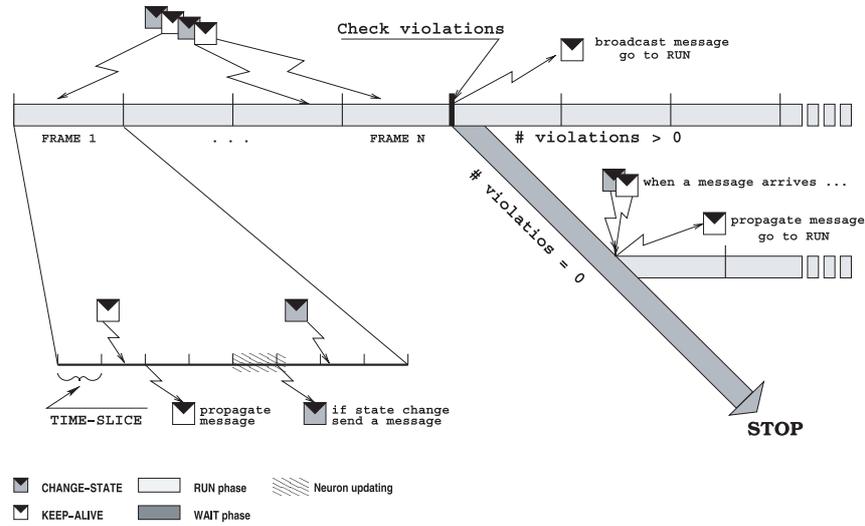


FIGURE 1. Phases of a processor that executes the DISHN.

Therefore, we say that a processor is in the RUN, WAIT or STOP state. The dynamics by which the processor p_k , during the execution, passes from one state to the other is given by the pseudo-code of Algorithm 2.

Algorithm 2 DISHN for processor p_k

```

building of a distributed spanning tree (communication network)
repeat
  state  $\leftarrow$  RUN
  if no constraint violations then
    state  $\leftarrow$  WAIT
  else
    broadcast KEEP-ALIVE on the communication network
    update the weights (relevant to the violations)
    state  $\leftarrow$  RUN
  end if
until state  $\neq$  STOP

```

A temporal diagram of the processor phases is given in Figure 1.

Each processor executes independently from the others and changes its state depending on the check done after a fixed time (number of neuron activations). If there are no constraint violations it passes to the WAIT state, otherwise it continues in the RUN state after doing the suitable weights updating. Through the broadcast system, every processor is informed about possible constraints violations occurred in other units. The time (number of TIME-SLICE) necessary to inform every processor depends on the diameter of the network and on the length of the

RUN phase. This system of notification allows every processor to determine when the neural network has reached a stable state on the basis of the arrival time of the last message.

Here is a detailed description of the RUN and WAIT phases with the relative pseudo-code (Algorithm 3 and Algorithm 4).

RUN phase

- It consists of a sequence of N FRAMEs and each FRAME is divided into a fixed number M of TIME-SLICEs. Within every FRAME the neuron selects, with uniform probability, an activation TIME-SLICE in order to recalculate its state. If the state changes, it sends the message CHANGE-STATE to its neighbours, otherwise it will not do anything.
- The incoming KEEP-ALIVE messages are broadcast through the communication network; the incoming CHANGE-STATE message causes a local memory change to hold the new state.
- At the end of this phase, if no violations with its neighbours occur, the processor goes into the WAIT state, otherwise it modifies the weights according to the weights updating rule.

Algorithm 3 DISHN: RUN phase for processor p_k

```

state  $\leftarrow$  RUN
for  $i = 1$  to  $N$  do
  time_clk  $\leftarrow$  0
  rand_update  $\leftarrow$  random_number_in  $\{1, \dots, M\}$ 
  for  $j = 1$  to  $M$  do
    time_clk  $\leftarrow$  time_clk + 1
    if msg KEEP-ALIVE is arrived then
      propagate KEEP-ALIVE on communication network
    end if
    if time_clk = rand_update then
      old_state  $\leftarrow$  neuron_state
      neuron_state  $\leftarrow$  HS  $\left( - \sum_{s \in \mathcal{N}(k)} w_s \cdot \text{neuron\_state}_s + \alpha \right)$ 
      if neuron_state  $\neq$  old_state then
        sends CHANGE-STATE to neighbourhood
      end if
    end if
  end for
end for

```

WAIT phase

- In the WAIT phase the processor waits for incoming messages. When a message KEEP-ALIVE or CHANGE-STATE arrives, the processor sends the KEEP-ALIVE message and it starts a new RUN phase. In case of CHANGE-STATE it stores the state of the neighbour in its local memory.

- If no message arrives within a fixed time interval (STOP-WINDOW), then the processor enters into the STOP state and it stops.

Finally, we have investigated the worst case time complexity of DISHN. By assuming that in each communication round every processor is allowed to send a message to each of its neighbours, the time complexity is the number of rounds the algorithm needs to solve the problem. There are three factors that concur to determine this value: the worst case number of networks $\alpha|E|$ (given in Th. 4.2), the number M of TIME-SLICES in the RUN phase and the number $M + D$ of TIME-SLICES in the WAIT phase.

Algorithm 4 DISHN: WAIT phase for processor p_k

```

state ← WAIT
time_clk ← 0
while time_clk < STOP-WINDOW do
  time_clk ← time_clk + 1
  if msg KEEP-ALIVE or CHANGE-STATE arrive then
    propagate KEEP-ALIVE on communication network
    go to RUN phase
  end if
end while
state ← STOP

```

In conclusion, assuming that the number of expected neuron collisions is less than one and using the upper bound of D , we have that:

$$\#\text{rounds} = \alpha|E|(2M + D) \leq 2\alpha(n - 1)|E|^2.$$

Therefore, for sparse graphs, *i.e.* such that $|E| = O(n)$, we have a complexity $O(n^3)$, while for dense graphs, *i.e.* when $|E| = \Omega(n^2)$, we have a time complexity $O(n^5)$ in the worst case.

In the following section, we will experimentally show that the assessment of the number of rounds obtained in the worst case is rather pessimistic.

6. SIMULATION RESULTS

In this section we present an experimental analysis of the behaviour of DISHN based on two types of computer simulations.

The goal of the first type of simulations is to compare ISHN⁴ with the AS-MISP heuristic [18], built by means of a recent technique inspired by the behaviour of colonies of real ants [6], and with ALHN, the variant based on the augmented

⁴Performances of ISHN have been already shown to be good in [11], in which it was compared with many other well known heuristics for MIS presented at the second DIMACS implementation challenge [16].

Lagrange multipliers in order to complete the experimental analysis of the model.

We have chosen AS-MISP because in [18] there is a wide comparison between AS-MISP and other heuristics such as the genetic algorithm of Bäck and Khuri [1] and the GRASP program of Feo *et al.* [8].

The goal of second type of simulations is twofold. First, it is to verify that the DISHN performances do not vary significantly when compared to the ISHN ones: the migration from the centralized model to the distributed one does not alter the solution quality. Second, it is to compare DISHN with other well-known distributed heuristics, such as the GREEDY algorithm and the FCSS (Fault-Containing Self-Stabilizing) algorithm [15]. GREEDY is the best known and most widely-used algorithm in distributed environments [12]. FCSS is a fault-containing self-stabilizing algorithm that finds a maximal independent set for an asynchronous distributed system and exhibits a $O(\Delta)$ stabilization time, where Δ is the maximum node degree in the system [15]. In [15], the authors claim that “*FCSS algorithm can be considered to be the most efficient fault-containing self-stabilizing algorithm for the maximal independent set finding up to this point*”. We compare our algorithm with FCSS even if the latter is designed for the Maximal (no Maximum) Independent Set problem because we wish provide experimental evidence that our algorithm approximate solutions are always better than exact solutions of a very efficient distributed algorithm for the previously mentioned easier variant of the problem.

We concentrate our analysis on the goodness of solutions, on the number of rounds (time complexity) and on the number of messages (message complexity) required by the algorithm to find a legal solution.

With regard to the solution quality, Table 1 reports the most interesting results obtained on a group of instances given by the so-called *p-random* graphs⁵. The first group of algorithms (AS-MISP, ALHN and ISHN) have been run as centralized process, the second group (GREEDY, FCSS and DSHN) have been run as distributed process. For each type of instances, we have considered a set of 30 random instances. The data reported in the table are the maximum and the average values respectively found for each set of instances. The average values are given at 99% confidence level. Moreover, we have evaluated the one-sided null hypothesis H_0 that the average results of GREEDY are better than those of DSHN. The experimental results show that it is possible to reject the null hypothesis at 1% significance level.

With respect to the performance of ALHN, we have noted that the results of DISHN are always better than those of ALHN even for instances proposed in [18]. The results indicate that DISHN and ISHN perform slightly better than AS-MISP and outperform all the other heuristics.

We have implemented all algorithms in C language, compiled by gcc 3.2 and the simulations have been executed on a workstation with Pentium IV 2.0 GHz processor, 512MB RAM and Linux 2.4.20 operating system.

⁵A *p-random* graphs of size n is a graph $\langle V, E \rangle$ where $V = \{1, \dots, n\}$ and E is obtained selecting $\{i, j\}$ as edge with probability p ($1 \leq i < j \leq n$ and $0 \leq p \leq 1$).

TABLE 1. Simulations results on different p -random graphs. The “ $Mn-p$ ” label represents a group of 30 p -graphs of size n , “max” the MIS maximum size found among all graphs and “ μ ” the MIS average size calculated at 99% confidence level. For brevity, the confidence interval is shown only for the first two best distributed algorithms: Greedy and DSHN. In the two groups of heuristics, the best averages are in bold face.

Instances	Centralized algorithms						Distributed algorithms					
	AS-MISP		ALHN		ISHN		GREEDY		FCSS		DSHN	
	max	μ	max	μ	max	μ	max	μ	max	μ	max	μ
M200-0.2	25	22.6	22	20.4	25	22.7	21	19.70 \pm 0.46	20	16.6	25	22.87 \pm 0.39
M200-0.5	11	10.0	10	8.8	12	10.3	9	8.10 \pm 0.37	9	7.0	12	10.40 \pm 0.25
M200-0.6	9	8.2	8	6.9	10	8.7	9	6.83 \pm 0.38	7	5.7	10	8.53 \pm 0.29
M200-0.83	5	5.0	5	4.2	6	5.1	5	4.17 \pm 0.23	4	3.5	6	5.07 \pm 0.13
M200-0.9	5	4.0	5	3.6	4	4.1	4	3.37 \pm 0.24	4	3.0	4	4.13 \pm 0.17
M400-0.2	28	26.3	27	24.2	30	27.8	27	22.97 \pm 0.72	22	19.1	30	27.97 \pm 0.43
M400-0.5	12	11.1	10	9.8	13	12.2	11	9.47 \pm 0.41	10	8.5	13	11.97 \pm 0.25
M400-0.6	9	9.0	8	7.9	11	9.8	9	7.43 \pm 0.34	8	6.8	10	9.80 \pm 0.20
M400-0.83	6	5.4	5	4.5	6	6.0	6	4.43 \pm 0.31	5	3.8	6	6.00 \pm 0.00
M400-0.9	5	4.3	4	3.9	5	5.0	4	3.60 \pm 0.24	4	3.1	5	4.97 \pm 0.09
M600-0.2	31	28.3	27	25.7	31	30.1	28	24.80 \pm 0.71	23	21.3	31	30.13 \pm 0.45
M600-0.5	13	11.6	12	10.7	13	12.9	11	9.90 \pm 0.43	12	8.7	13	12.90 \pm 0.15
M600-0.6	10	9.3	10	8.4	11	10.6	9	7.97 \pm 0.31	9	7.0	11	10.60 \pm 0.25
M600-0.83	6	5.8	6	5.1	7	6.4	5	4.50 \pm 0.25	5	4.2	6	6.00 \pm 0.00
M600-0.9	5	4.7	4	4.0	6	5.1	4	3.73 \pm 0.22	4	3.2	5	5.00 \pm 0.00
M800-0.2	32	29.8	28	27.1	33	31.8	31	26.47 \pm 0.68	26	22.9	33	31.80 \pm 0.31
M800-0.5	13	12.2	12	10.8	15	13.7	12	10.43 \pm 0.36	11	9.1	15	13.60 \pm 0.25
M800-0.6	11	9.3	10	9.0	12	11.1	9	8.40 \pm 0.25	9	7.4	12	11.10 \pm 0.15
M800-0.83	6	6.0	6	5.2	7	6.4	6	4.77 \pm 0.25	5	4.3	7	6.40 \pm 0.25
M800-0.9	5	4.3	5	4.1	6	4.9	5	4.07 \pm 0.23	4	3.4	5	5.00 \pm 0.00
M1000-0.2	33	31.1	29	27.8	35	33.5	30	26.90 \pm 0.68	26	23.9	35	33.53 \pm 0.37
M1000-0.5	13	12.5	12	11.3	14	13.8	12	10.53 \pm 0.41	12	9.6	14	13.90 \pm 0.15
M1000-0.6	12	10.2	10	9.3	12	11.6	10	8.40 \pm 0.31	9	7.5	11	11.00 \pm 0.00
M1000-0.83	7	6.0	6	5.4	7	6.5	6	5.03 \pm 0.26	5	4.2	7	6.70 \pm 0.23
M1000-0.9	5	5.0	5	4.2	6	5.2	5	4.23 \pm 0.28	5	3.5	6	5.40 \pm 0.25

To give an idea of the actual time complexity, we have included a chart (Fig. 2) showing the number of rounds required by DISHN for various graph sizes and densities. Please note that the number of rounds increases with both the size and the density; therefore, the best performances can be obtained on sparse graphs, *i.e.*, graphs having a density less than 20% of the maximum number of edges.

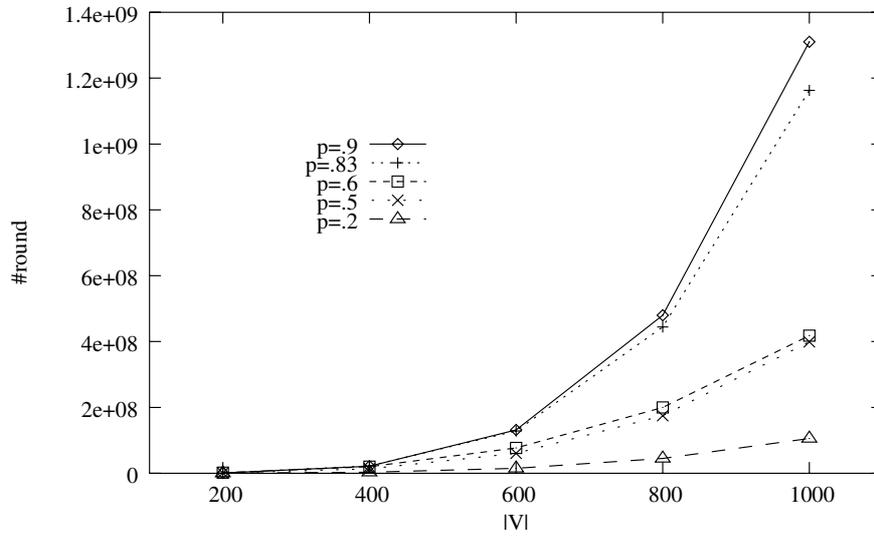


FIGURE 2. Graph size *vs.* #rounds required by DISHN for different graph density ($p = 0.2, 0.5, 0.6, 0.83, 0.9$).

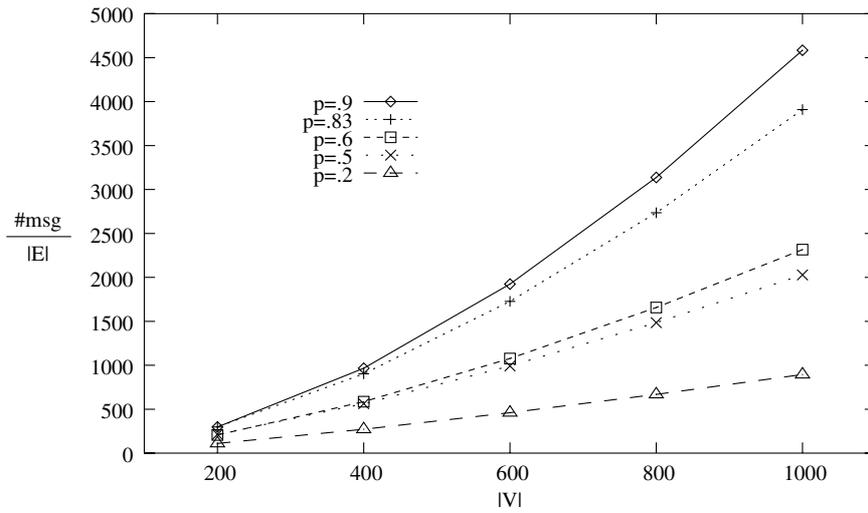


FIGURE 3. Graph size *vs.* $\#msg/|E|$ required by DISHN for different graph density ($p = 0.2, 0.5, 0.6, 0.83, 0.9$).

As far as the message complexity is concerned, we have included the average number of messages for channel in Figure 3. Similarly to the previous parameter, this ratio increases with the size and the density of the graph and, consequently, the performances are better for low density graphs.

7. CONCLUSIONS

In this paper, an asynchronous distributed not-deterministic algorithm for the MIS problem is proposed together with some computer simulation showing the goodness of the solutions found. The algorithm uses Hopfield-type neural networks in which the neurons behaviour is simulated by independently running processors. The time and message complexity analysis shows that the lower the graph density, the better the performances.

Further work must be done in order to generalize the distributed algorithm for topology that changes dynamically, as in the peer-to-peer mobile network framework. From a theoretical point of view, the model is robust against any changing of the neighbourhood of neurons since the sequence of networks always guarantees the convergence of the system. However, a better knowledge about the system behaviour when the network topology changes may be obtained by means of an intensive experimental analysis.

A problem closely connected with MIS is the problem concerning the search of the minimum set of vertices of a graph such that every edge of the graph has at least one of its end points in the set. This problem is known as **Minimum Vertex Cover**. It is easy to show that this problem can be solved approximately by the neural system described here.

REFERENCES

- [1] T. Bäck and S. Khuri, An evolutionary heuristic for the maximum independent set problem, in *Proc. First IEEE Conf. Evolutionary Computation, IEEE World Congress on Computational Intelligence*, edited by Z. Michalewicz, J.D. Schaffer, H.-P. Schwefel, D.B. Fogel and H. Kitano, *Orlando FL, June 27-29*. IEEE Press, Piscataway NJ **2** (1994) 531-535.
- [2] S. Basagni, Finding a maximal weighted independent set in wireless networks. *Telecommunication Systems, Special Issue on Mobile Computing and Wireless Networks* **18** (2001) 155-168.
- [3] R. Battiti and M. Protasi, Reactive local search for the maximum clique problem. *Algorithmica* **29** (2001) 610-637.
- [4] A. Bertoni, P. Campadelli and G. Grossi, A neural algorithm for the maximum clique problem: Analysis, experiments and circuit implementation. *Algorithmica* **33** (2002) 71-88.
- [5] I.M. Bomze, M. Budinich, M. Pelillo and C. Rossi, Annealed replication: A new heuristic for the maximum clique problem. *Discrete Appl. Math.* **121** (2000) 27-49.
- [6] M. Dorigo, G. Di Caro and L.M. Gambardella, Ant algorithms for discrete optimization. *Artificial Life* **5** (1996) 137-172.
- [7] U. Feige, S. Goldwasser, S. Safra, L. Lovàsz and M. Szegedy, Approximating clique is almost NP-complete, in *Proceedings of the 32nd Annual IEEE Symposium on the Foundations of Computer Science* (1991) 2-12.
- [8] T.A. Feo, M.G.C. Resende and S.H. Smith, Greedy randomized adaptive search procedure for maximum independent set. *Oper. Res.* **41** (1993).
- [9] M. Gerla and J.T.-C. Tsai, Multicluster, mobile, multimedia radio network. *Wireless Networks* **1** (1995) 255-265.
- [10] W. Goddard, S.T. Hedetniemi, D.P. Jacobs and P.K. Srimani, Self-stabilizing protocols for maximal matching and maximal independent sets for ad hoc networks, in *Workshop on Advances in Parallel and Distributed Computational Models* (2003).

- [11] G. Grossi and R. Posenato, A distributed algorithm for max independent set problem based on Hopfield networks, in *Neural Nets: 13th Italian Workshop on Neural Nets (WIRN 2002)*, edited by M. Marinaro and R. Tagliaferri. Springer-Verlag. *Lect. Notes Comput. Sci.* **2486** (2002) 64–74.
- [12] M.M. Halldórsson and J. Radhakrishnan, Greedy is good: Approximating independent sets in sparse and bounded-degree graphs. *Algorithmica* **18** (1997) 145–163.
- [13] J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$, in *Proc. of the 37rd Annual IEEE Symposium on the Foundations of Computer Science* (1996) 627–636.
- [14] J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, in *Proceedings of the National Academy of Sciences of the United States of America* **79** (1982) 2554–2558.
- [15] L. Ji-Cherng and T.C. Huang, An efficient fault-containing self-stabilizing algorithm for finding a maximal independent set. *IEEE Transactions on Parallel and Distributed Systems* **14** (2003) 742–754.
- [16] D.S. Johnson and M.A. Trick, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society (1996).
- [17] R.M. Karp, *Reducibility among Combinatorial Problems*. Complexity of Computer Computations. Plenum Press, New York (1972) 85–103.
- [18] G. Leguizamón, Z. Michalewicz and M. Schütz, An ant system for the maximum independent set problem, in *Proceedings of VII Argentine Congress of Computer Science (CACIC 2001)* (2001).
- [19] S.Z. Li, Improving convergence and solution quality of Hopfield-type neural networks with augmented Lagrange multipliers. *IEEE Trans. Neural Networks* **7** (1996) 1507–1516.
- [20] E. Marchiori, A simple heuristic based genetic algorithm for the maximum clique problem, in *Proc. ACM Symp. Appl. Comput* (1998) 366–373.
- [21] C.H. Papadimitriou, *Computational Complexity*. Addison-Wesley (1994).