International Journal of Neural Systems, Vol. 19, No. 4 (2009) 241–252

World Scientific
www.worldscientific.com

ADAPTIVENESS IN MONOTONE PSEUDO-BOOLEAN OPTIMIZATION AND STOCHASTIC NEURAL COMPUTATION

GIULIANO GROSSI
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano, Via Comelico 39
I-20135 Milano, Italy
grossi@dsi.unimi.it

Hopfield neural network (HNN) is a nonlinear computational model successfully applied in finding near-optimal solutions of several difficult combinatorial problems. In many cases, the network energy function is obtained through a learning procedure so that its minima are states falling into a proper subspace (feasible region) of the search space. However, because of the network nonlinearity, a number of undesirable local energy minima emerge from the learning procedure, significantly effecting the network performance.

In the neural model analyzed here, we combine both a penalty and a stochastic process in order to enhance the performance of a binary HNN. The penalty strategy allows us to gradually lead the search towards states representing feasible solutions, so avoiding oscillatory behaviors or asymptotically instable convergence. Presence of stochastic dynamics potentially prevents the network to fall into shallow local minima of the energy function, i.e., quite far from global optimum. Hence, for a given fixed network topology, the desired final distribution on the states can be reached by carefully modulating such process.

The model uses pseudo-Boolean functions both to express problem constraints and cost function; a combination of these two functions is then interpreted as energy of the neural network. A wide variety of NP-hard problems fall in the class of problems that can be solved by the model at hand, particularly those having a monotonic quadratic pseudo-Boolean function as constraint function. That is, functions easily derived by closed algebraic expressions representing the constraint structure and easy (polynomial time) to maximize.

We show the asymptotic convergence properties of this model characterizing its state space distribution at thermal equilibrium in terms of Markov chain and give evidence of its ability to find high quality solutions on benchmarks and randomly generated instances of two specific problems taken from the computational graph theory.

Keywords: Hopfield neural networks; stochastic dynamics; nonlinear pseudo-Boolean optimization; penalty strategies; heuristics.

1. Introduction

The HNN model represents a paradigmatic artificial neural network with a wide range of applications. It has been used to model the process of associative memory and more generally as classification algorithm. Furthermore, HNN and its variants have been applied to several combinatorial optimization problems as, for instance, the travelling salesman problem and several graph problems as well as graph bipartitioning, weighted matching, independent set.

In some cases, the optimization process can be bettered by approaches based on stochastic modifications of the network to find local minima that satisfy a set of given constraints characterizing a restricted set of admissible solutions. Unfortunately, because of its nonlinear character, often the network can also exhibit non-desirable, local optima. A common approach to handling constraints in neural optimization is to apply a penalty function to bias the search toward a feasible solution. In other terms,

penalty consists on a learning strategy that adaptively chooses the network weights in order to determine a distributions on state space at the thermal equilibrium which will favor valid and better solutions. In such a way the probability to find non-admissible solutions continuously decreases up to negligible values.

The idea in our architecture, is related to the notion of asymptotic stability and performance of HNN with stochastic nature. We must alter the stability by introducing "noise" (penalization) when the search direction does not promise to satisfy essentials requirements, falling toward undesirable local energy minima with hight probability. To avoid such a behavior, the network performs a constraint-satisfaction search process that begins with "weak" constraints and then proceeds by gradually strengthening them until a feasible state is found. Based on a stochastic state transition mechanism, the process adaptively impose network connections in such a way that the energy function associated with the network is optimized for a set of desired network states. The model is thus described by Markov chain showing an equilibrium distribution like those characterizing Boltzmann machines, even if the latest has substantial difference with the network model studied here. Also a convergence analysis which explains both why the number of violations tends to zero and why the maximality of the solution is given.

A family of functions that often plays an important role in optimization models are the so-called pseudo-Boolean functions.[6] Their polynomial representation, with particular regard to the quadratic and symmetric ones, corresponds in a natural way to the objective function of many optimization problems. Our aim here is to use them to discriminate admissible solutions from non-admissible ones by introducing suitable pseudo-Boolean penalty functions, i.e., mappings from the family of subsets of a finite ground set to the set of integers. Specifically, we consider monotone multivariate functions based on a linear combination of a fixed collection of bivariate boolean functions (binary constraints) whose optima are characteristic vectors of admissible solutions.

With the aim of dealing with a wide class of problems, we show that this neural computing model is promising when applied to the optimization of a particular class of NP-hard constrained combinatorial optimization problems[10,23] having monotone and symmetric quadratic pseudo-Boolean constraint functions. To show the performances of the proposed heuristic, we have done some computational experiments on two general problems on graph, namely the minimum vertex cover and maximum clique problems. We compare its performances with those of other heuristics known in literature. We found that it outperforms these algorithms in many cases or in the worst case it gives comparable results.

This paper is organized as follows. Section 2 presents some works on related concepts present in literature. Section 3 explains why monotonic pseudo-Boolean functions can be used in neural optimization area. Section 4 considers a generalization of the adaptive stochastic neural model previously presented for maximum clique problem and presents its convergence analysis. Some experiments that validate the heuristic for two problems taken from graph theory are reported in Sec. 5. Finally, conclusions are given in Sec. 6.

2. Related Works

The natural connections between pseudo-Boolean functions and nonlinear binary optimization have motivated and strongly influenced some of the first studies in this area.[16,17] Since then the study of pseudo-Boolean functions has grown to a major area of research with hundreds of related publications in the last 30 years.[6]

On the other hand, the reputation of neural networks for combinatorial optimization, widely documented in over two decades of research, has known various degrees of success. In some cases they showed they were not competitive with ad-hoc heuristics designed for a specific problem; nevertheless, almost every type of combinatorial problem has been tackled by neural networks. Many neural approaches result in behavior comparable to alternative techniques in terms of solution quality (for a review see the paper of Smith[29]; see also Refs. 27 and 7).

Among them, the role played by stochastic dynamics in Hopfield and analogous models has been relevant and intensely used since the introduction of the basic algorithm[19] and its first application to combinatorial optimization.[20,21] For instance in the paper[32] a learning algorithm for the Hopfield

neural network is presented for solving the minimum set cover problem. The learning algorithm adjusts a parameter in the energy function so that the local minimum that the network once falls into vanishes and the network can continue updating in a gradient descent direction of energy. Particularly interesting is also the work[31] which presents a stochastic optimal competitive Hopfield network to solve NP-hard partitional clustering problem, which is to partition a data set into a specified number of clusters according to certain criteria, e.g., a square error function. The method permits temporary energy increases and includes stochastic dynamics, which helps the Hopfield network escape from local minima by introducing a hill-climbing dynamics. In 2003, Galán-Marín *et al.*[9] presented a competitive Hopfield model that always guarantees convergence to a global/local minimum of the energy function for the maximum/maximal clique problem of a given graph. It consists on a discrete Hopfield model based on the notion of group update introduced by Takefuji *et al.*[30] in the maximum neural network. The result is a new network, namely the optimal competitive Hopfield model, with a high speed of convergence even for large-scale problems.

Also penalty strategies are widely used to lead the network toward a subset of states in a more general state space. As an example, in Ref. 3 a nonlinear neural dynamics is applied to minimum weight design of space trusses subjected to stress and displacement constraints. The global convergence and the stability of the neural dynamical system is reached by adopting an exterior penalty function method. This issue is also addressed in Ref. 28 in which a set of constraint-specific penalty or weighting coefficients are defined by learning-based approach. These values are in turn used to compute values of network weights, effectively eliminating the guesswork in defining weight values for a given static optimization problem.

3. Constrained Pseudo-Boolean Optimization

Constrained optimization based on pseudo-Boolean functions is a subject of two previous papers,[13,14] briefly recalled here.

We represent pseudo-Boolean functions by means of (the unique) multi-linear polynomials having the form:

$$f(x_1, \ldots, x_n) = \sum_{S \subseteq V} c_S \prod_{k \in S} x_k,$$

where c_S are integer coefficients and, by convention, $\prod_{k \in \emptyset} x_k = 1$. The size of the largest subset $S \subseteq V$ for which $c_S \neq 0$ is called the degree of f, and is denoted by $\deg(f)$. Naturally, a pseudo-Boolean function f is quadratic if $\deg(f) \leq 2$. A pseudo-Boolean function f is monotone non-decreasing if $f(\mathbf{x}) \leq f(\mathbf{y})$ for every $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$ such that $\mathbf{x} \leq \mathbf{y}$ (componentwise), while it is monotone non-increasing if $f(\mathbf{x}) \geq f(\mathbf{y})$.

For a given problem $\mathcal{P}$ of size n, let us denote with $\mathcal{C}_P$ its set of variable pairs (x_i, x_j) for which a binary constraint is defined, that is, a Boolean functions of the form $g : \{0,1\}^2 \to \{0,1\}$ such that $g(x_i, x_j) = 1$ if and only if the assignment $\mathbf{x} = \{x_1, \ldots, x_n\}$ locally satisfy the constraint.

Under this setting, the above constraints are linearly combined into the following monotone (non-decreasing) quadratic pseudo-Boolean function:

$$\psi(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{(x_i, x_j) \in \mathcal{C}_P} \gamma_{ij} g(x_i, x_j) \tag{1}$$

that we call the constraint function. The parameters γ_{ij} are the coefficients or weights associated to the constraints.

Thus, a maximization problem $\mathcal{P}$ with linear objective function $f = \sum_{i=1}^{n} x_i$ and unitary coefficients γ_{ij} can be translated in the following general form

$$\begin{aligned} &\mathsf{maximize} && \sum_{i=1}^{n} x_i \\ &\mathsf{subject\ to} && \psi(\mathbf{x}) = |\mathcal{C}_P|. \end{aligned} \tag{2}$$

Observe that, by changing the sign to the cost function f, also minimization problems can be treated in this framework.

Combinatorial optimization problems, which arise naturally in many theoretical or applicative areas such as logic, set and graph theory, programming, electronics, etc., can be easily formulated as pseudo-Boolean optimization problems. Some example are: max independent set, max CSP (constraint satisfaction problems), min vertex cover and max k-colorable induced subgraph.

For instance, in order to find a vertex cover of an undirected graph, i.e., a subset of vertices such that

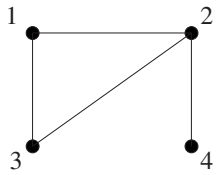

Fig. 1. Instance of the the min vertex cover problem: $V = \{1,2,3,4\}$ and $E = \{(1,2),(1,3),(1,4),(2,3),(2,4)\}$.

each edge has at least one end-point in the subset, we translate the problem in the form:

$$\begin{aligned} &\text{maximize} && -\sum_{i\in V} x_i \\ &\text{subject to} && \sum_{\{i,j\}\in E} (x_i \vee x_j) = |E|. \end{aligned}$$

As an example, consider the graph $G = \langle V, E\rangle$ depicted in Fig. 1.

Since at each clause $(x_i \vee x_j)$ corresponds the multi-linear polynomial $x_i + x_j - x_i x_j$, the constraint function becomes:

$$\begin{aligned} \psi(\mathbf{x}) = {} & 2x_1 + 3x_2 + 2x_3 + x_4 - x_1x_2 - x_1x_3 \\ & - x_2x_3 - x_2x_4. \end{aligned}$$

Its maximum value is 4 and is given, for instance, by the assignments $(1,1,0,0)$ and $(0,1,1,0)$; notice that the latter also represent the best solutions for the min vertex cover instance.

4. The Model

In this section we recall the adaptive stochastic neural model introduced in the recent papers.[13,14]

The energy function combines together the cost and the constraint function as shows the following quadratic pseudo-Boolean form (up to an additive costant term):

$$E(x_1,\ldots,x_n) = \alpha \sum_{i=1}^{n} x_i + \sum_{(i,j)\in\mathcal{C}_P} \gamma_{ij} g(x_i, x_j). \tag{3}$$

Here α represents a positive constant useful to improve the effectiveness of the neural optimizer. The parameters γ_{ij} are positive integers modified by learning process in order to find admissible solutions.

For each $\alpha > 0$ the energy preserves the optima of the problem. Infact, if $\mathbf{y}$ is an admissible solution, the quadratic term $\sum_{(i,j)\in\mathcal{C}_P} \gamma_{ij} g(x_i, x_j)$ assumes its maximum value on it because all constraints are satisfied, i.e.,

$$\psi(\mathbf{y}) = \sum_{(i,j)\in\mathcal{C}_P} \gamma_{ij} = M = \max_{\mathbf{x}\in\{0,1\}^n} \psi(\mathbf{x}).$$

Thus, every admissible solution $\mathbf{y}$ has energy

$$E(\mathbf{y}) = \alpha \sum_{i=1}^{n} x_i + M,$$

making the linear term $\sum_{i=1}^{n} x_i$ the only responsible of the cost.

The network architecture is derived from the problem (for instance, the graph topology), with the set of neurons (or units) isomorphic to the set of binary variables representing the solutions. The connections between neurons are achieved according to the logical dependencies between variable pairs: the absence of such a connection implies neither strength in the synaptic connection between neurons nor constraint between the variables themselves.

Each unit i $(1 \le i \le n)$ is stochastic, assuming the state according to the rule

$$x_i = \begin{cases} 1, & \text{with probability } \phi_\beta(h) \\ 0, & \text{with probability } 1 - \phi_\beta(h) \end{cases}$$

where,

$$\phi_\beta(h) = \frac{1}{1 + e^{-\beta h}} \tag{4}$$

is the classical *Glauber*[12] or *logistic* sigmoid-shaped function conditioned by parameter $\beta > 0$. Usually, in recurrent stochastic neural networks β controls the gain (or slope) of the activation function (e.g., annealing algorithms[24,25] or Boltzmann machines[2]) and it is proportional to the inverse of temperature. But the main difference between our model and the classical stochastic ones is that instead of define a cooling protocol for the temperature, we assign to β a fixed value ranging from 1 (to favorite the hight energy level) to α (to speed up the search), and we will simply let the system adaptively choose its weights to lead the search toward a feasible solution. Therefore, the role played by annealing, consisting on leading the system along the saturation region of the sigmoid function, is here covered by the learning phase. Therefore, by adopting the saturated-nonlinear activation function (4) and letting the network evolve with asynchronous dynamics, the resulting dynamical system locally minimizes the network energy with high probability.

The proposed weight learning process consists on a penalty method applied at the end of one or more updatings of all neurons even if the network has not reached the thermal equilibrium: for each unit pair (x_i, x_j) that violates the problem constraint $g(x_i, x_j)$, the absolute value of the weight γ_{ij} is increased. More formally, it can be written as follows:

$$\forall (x_i, x_j) \in \mathcal{C}_P \quad \textbf{if } g(x_i, x_j) = 0 \\ \textbf{then } \gamma_{ij} = \gamma_{ij} + 1.$$

As said above, the neuron state changes according to the alternation of units updating and weights strengthening stages. A characteristic sequence of states assumed by a neuron during a simulation and relative probability are showed in Fig. 2. It is evident typical fluctuations that are due to the non-stationarity character of the system because the probability distribution $\phi(h_i)$ changes continuously.

A key issue is to link the energy gain at each step during the network evolution with the penalty strategy explained above to show the ability of neural system to discriminate in the state space.

To update the network state, a unit i is chosen at random or in sequence and its input is calculated by the analytical differentiation of the energy function (i.e., by the relationship that bind the (3) and the (4)):

$$\begin{aligned} h_i(x_1, \ldots, x_n) &= \frac{\partial}{\partial x_i} E(x_1, \ldots, x_n) \\ &= \alpha + \sum_{j \in \mathcal{N}(i)} \gamma_{ij} \frac{\partial}{\partial x_i} g(x_i, x_j) \\ &= \Delta E_i. \end{aligned}$$

Thus, flipping the state of the unit i produces a variation of energy ΔE_i which does not depend on x_i. This energy dependency helps the algorithm converge toward a feasible solution, because h_i is a derivative of a function depending on all variables connected with x_i which grows indefinitely when the previous updating schema is applied. We can state:

Proposition 4.1. *Let $h_i = \frac{\partial E}{\partial x_i}$ involving all variables x_j such that $j \in \mathcal{N}(i)$. Then*:

$$\sup_{\gamma_{ij} \in \mathbf{R}^+} h_i(\mathbf{x}) = \begin{cases} \pm\infty, & \text{if } \exists j \mid g(x_i, x_j) = 0 \\ \alpha, & \text{otherwise} \end{cases}.$$

Proof. The proof is easily given recalling the assumption that the quadratic Boolean function g has as partial derivative the linear function

$$\frac{\partial g(x_i, x_j)}{\partial x_i} = b + c x_j,$$

where $b \in \{0, 1, -1\}$ and $c \in \{1, -1\}$ are the coefficients of the linear and quadratic term of $g(x_i, x_j)$

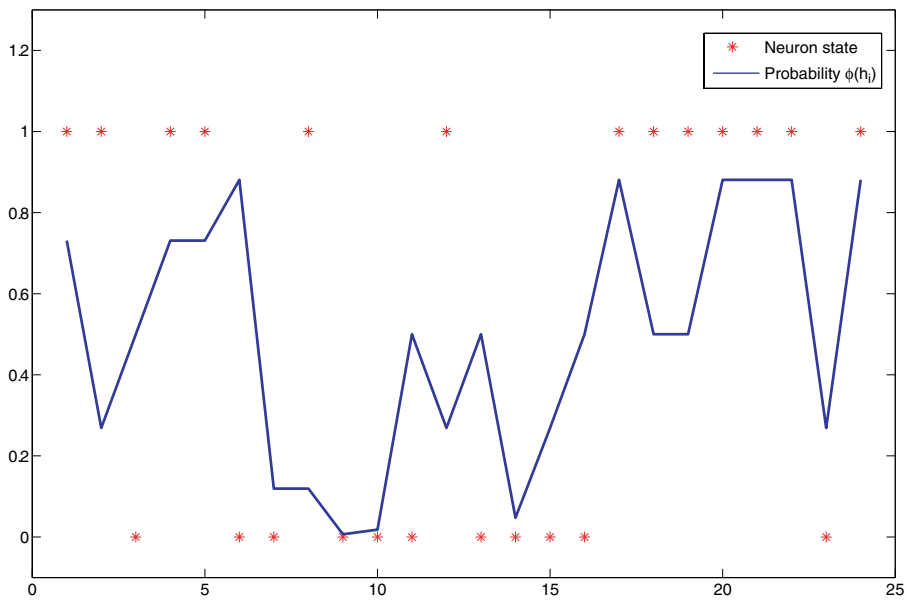

Fig. 2. State fluctuation of a neuron during the network evolution.

respectively. Note that, if $b \neq 0$ then $b = -c$ because $g(x_i, x_j)$ has maxima and minima on the set $\{0, 1\}$.[2]

This gives rise to a linear combination of all positive or all negative terms (the sign depends on c) in h_i:

$$h_i(x_1, \ldots, x_n) = \alpha + \sum_{j \in \mathcal{N}(i)} \gamma_{ij} \frac{\partial}{\partial x_i} g(x_i, x_j)$$
$$= \alpha + bN_i\gamma_{ij} + \sum_{j \in \mathcal{N}(i)} c\gamma_{ij}x_j,$$

where $N_i = |\mathcal{N}(i)|$. Since all γ_{ij} are positive, it is obvious that $\sup h_i(\mathbf{x}) = \alpha$ when the variables x_j are all null and $b = 0$ or they are all 1 and $b \neq 0$, respectively. Vice versa it is $\pm\infty$. □

The two main phases of the ASNM algorithm (Adaptive Stochastic Neural Model) that simulates the neural network behavior, alternating units updating and selective penalization, is sketched as flow diagram in Fig. 3.

4.1. *Markov chain analysis*

Consider the previous system in thermal equilibrium with 2^n different states and associated energies $E(\mathbf{x}_1), E(\mathbf{x}_2), \ldots, E(\mathbf{x}_{2^n})$. Dynamics of state transitions can be described as a Markov chain with the state mixing matrix $P = (p_{ab})$ such that the probability p_{ab} of a transition from a state of energy $E(\mathbf{x}_a)$ to another state of energy $E(\mathbf{x}_b)$ is given by

$$p_{ab} = \frac{1}{1 + e^{\beta(E(\mathbf{x}_b) - E(\mathbf{x}_a))}}.$$

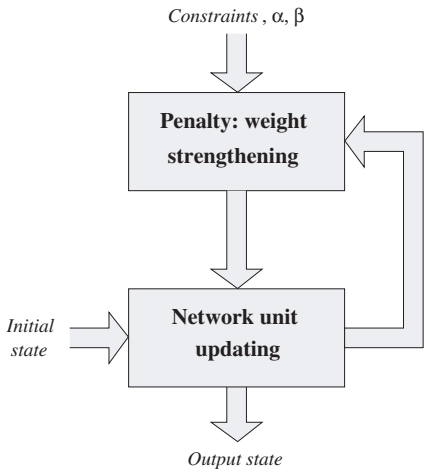

Fig. 3. Two main phases of the ASNM algorithm.

According to the Boltzmann distribution the probability p_a that a system assumes the energy level $E(\mathbf{x}_a)$ during thermal equilibrium is

$$p_a = \frac{e^{\beta E(\mathbf{x}_a)}}{Z},$$

where $Z = \sum_{a=1}^{2^n} e^{\beta E(\mathbf{x}_a)}$ is a normalizing factor known as the state sum.

In constraint optimization the transition probability must be designed in such a way that the limiting distribution of the undesiderable states becomes arbitrary "little", then the infeasible solutions can be altogether avoided. Moreover, it must also be guaranteed that the better solutions have more chance to be selected within the admissible state space.

In this model we have exactly this behavior. If $\mathbf{x}_a = (x_{a1} \cdots x_{an})$ is the binary representation of the state a its probability at equilibrium is

$$p_a = \frac{1}{Z} e^{\beta(\alpha \sum_{i=1}^{n} x_i + \sum_{(i,j) \in \mathcal{C}_P} \gamma_{ij} g(x_{ai}, x_{aj}))}.$$

By denoting as $M_a = \sum_{(i,j) \in \mathcal{C}_P} \gamma_{ij} g(x_{ai}, x_{aj})$ and defining M to be the maximum above all M_a, if we divide numerator and denominator of previous expression by the term $e^{\beta M}$ we obtain:

$$p_a \sim \frac{e^{\alpha \sum_{i=1}^{n} x_{ai}}}{e^{M - M_a}}.$$

Depending on the character of state a, it can be observed two kinds of behaviors of the probability p_a:

- if a represents a feasible state the probability is proportional to the size of the correspondent solution because $M_a = M$, that is

$$p_a \sim e^{\alpha \sum_{i=1}^{n} x_{ai}};$$

- if a is not a feasible state, p_a is reduced by factor of

$$e^{M - M_a} > 1.$$

Notice that when the algorithm runs, if the stop condition is not reached, the term $e^{M - M_a}$ tends to infinity since at least one of the parameters γ_{ij} continues to increase because of the alternation of neuron update and weight strengths, as explained better in the next section. Figure 4 shows how the growing values of parameters γ_{ij} forces the system to choose the probability in the saturation regions of the sigmoid.

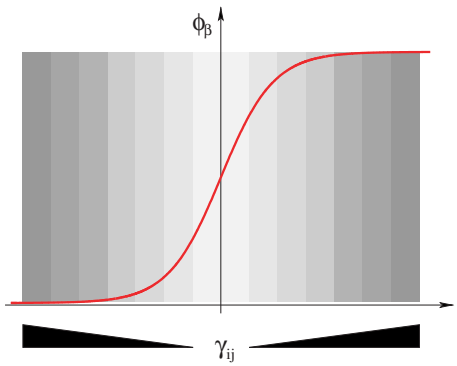

Fig. 4. Saturation regions of the sigmoid and its relation with the parameters γ_{ij}.

4.2. *Convergence analysis*

The purpose of this section is to prove that under the conditions given on the constraints and a random initialization, the stochastic process deriving by neural network simulation stops with hight probability. In particular, it must be proven that the stable state reached by the sequence of neural networks not only represents a valid solution, but also that it is a maximal solution of the optimization problem embedded in the network structure.

According to this, it holds:

Theorem 1. *Let* $\mathbf{x}(t)$ *be the state found by the algorithm* ASNM *at time* t *and let* $Z(t)$ *be the random variable denoting the number of violations at the same time. Then,*

$$\lim_{t\to+\infty} \boldsymbol{Pr}\{Z(t) > 0\} = 0,$$

that is, the probability of having violations in $\mathbf{x}(t)$ *vanishing as* t *tends to infinity.*

Proof. Let us give a maximization problem $\mathcal{P}$ with constraint set $\mathcal{C}_P$, of size $m = |\mathcal{C}_P|$, as described in (1) and cost function of the form given in (3). Without loss of generality, we also assume from now on $\beta = 1$.

First of all, we observe that, at every time t, if the stop condition for ASNM algorithm is false, at least one constraint among $g(x_i, x_j)$ is not satisfied causing the related weight γ_{ij} in (3) to increase in absolute value; otherwise there are no violations and the algorithm stops. Thus, for all γ_{ij} in (3) it holds that:

$$1 \leq \cdots \leq \gamma_{ij}(t-1) \leq \gamma_{ij}(t) \leq \cdots.$$

Let $Z_{ij}(t) \in \{0, 1\}$ be a random variable which tells whether the violation of the constraint involving the pair (x_i, x_j) has occurred at time t, i.e., $Z_{ij}(t) = 1$ if occurred, $Z_{ij}(t) = 0$ otherwise. Let us assume that the function $g(x_i, x_j)$ is monotone non-increasing, so its partial derivative are positive and the coefficient of the quadratic term c is negative. In this case we have a violation with high probability if both $\phi(h_i(\mathbf{x}(t)))$ and $\phi(h_j(\mathbf{x}(t)))$ are near 1, i.e., when it is verified that:

$$\phi(h_i(\mathbf{x}(t))) \approx 1 \quad \wedge \quad \phi(h_j(\mathbf{x}(t))) \approx 1. \tag{5}$$

The probability of having a violation on the link (i, k) (condition (5)) is:

$$\begin{aligned}\mathbf{Pr}\{Z_{ij}(t) = 1\} &= \mathbf{Pr}\{g(x_i(t), x_j(t))) = 0\} \\ &= \max\{\phi(h_i(\mathbf{x}(t))), \phi(h_j(\mathbf{x}(t)))\}.\end{aligned}$$

Thus, at time t, we have:

$$\begin{aligned}\mathbf{Pr}\{x_i(t) = 1\} &= \phi(h_i(\mathbf{x}(t))) \\ &= \frac{1}{1 + e^{-(\alpha - \sum_{k\in\mathcal{N}(i)} \gamma_{ik}(t)x_k(t))}} \\ &\geq \frac{1}{1 + e^{-(\alpha - \gamma_{ij}(t)x_j(t))}},\end{aligned}$$

which tends to 0 when $\gamma_{ij}(t) \to +\infty$. Obviously, the same holds for $\phi(h_j(\mathbf{x}(t))$. It is easy to extend the previous proof to the other cases in which the constraint function has a negative partial derivative and the coefficient of the quadratic term satisfies $c > 0$.

Let $Z(t) = \sum_{(x_i,x_j)\in\mathcal{C}_P} Z_{ij}(t)$ be a random variable which denotes the number of violations at time t in $\mathbf{x}(t)$, and then assuming values on the set $[0, \ldots, m]$. Since the updates of each constraint are independent from each other, it holds

$$\begin{aligned}&\lim_{t\to+\infty} \mathbf{Pr}\{Z(t) > 0\} \\ &\quad= \lim_{t\to+\infty} \mathbf{Pr}\left\{\sum_{(x_i,x_j)\in\mathcal{C}_P} Z_{ij}(t) > 0\right\} \\ &\quad= \sum_{(x_i,x_j)\in\mathcal{C}_P} \lim_{t\to+\infty} \mathbf{Pr}\{Z_{ij}(t) > 0\} = 0.\end{aligned}$$

□

About the solution quality, it can be objected that the system tends to converge toward states that trivially satisfy all constraints, that is vectors with zeros or ones in all components. Actually, we prove that the neural system always find optimal solutions

i.e., solutions that are not subset of another one, as assured by the following:

Corollary 1. *Let $\mathbf{x}^S$ be the stable state of the network found by the ASNM algorithm. Then $\mathbf{x}^S$ represents, with high probability, an optimal solution, i.e., S is a maximal subset.*

Proof. Theorem 1 shows that the solution $\mathbf{x}^S$ found by the ASNM algorithm is admissible, which means no constraint violation and thus $E(\mathbf{x}^S) = \alpha \sum_{i=1}^{n} x_i^S$.

Let us suppose now that a maximal set $T \supset S$ exists such that $\sum_{i=1}^{n} x_i^T - x_i^S = k > 0$ and consequently $E(\mathbf{x}^T) - E(\mathbf{x}^S) = k\alpha$, which implies for each $v \in T - S$ that

$$\begin{aligned}\mathbf{Pr}\{x_v = 0\} &= 1 - \phi(h_v(\mathbf{x}(t))) \\ &= 1 - \frac{1}{1 + e^{-\Delta E_v}} \\ &= 1 - \frac{1}{1 + e^{-\alpha}} = \frac{1}{1 + e^{\alpha}}.\end{aligned}$$

Since $\alpha > 1$, we have that $\mathbf{Pr}\{x_v = 0\} < 0.11$ and it is about $4,5 * 10^{-5}$ when we set $\alpha = 10$, as was done in the experiments. □

Differently from systems based on the Ising model with Glauber dynamics,[12] like Boltzmann machines[2,1] and stochastic Hopfield network,[18] the proposed model does not provide a control parameter like temperature. Nevertheless, for maximization (minimization) problems, this dynamical system naturally "moves" in the direction of increasing (decreasing) energy but this also allows it to move in the opposite direction. The probability of such a move is initially high, but it decreases during the system evolution (the argument of the activation function gets far away from zero, in the regions of saturation). In other words, the probability of making a contrary move depends on the absolute value of (4), which is proportional to the number of violations.

The previous results are asymptotic and therefore not suitable for determining the time spent by the algorithm in reaching a feasible state. To this end, is more useful the worst case analysis in order to give a lower bound to the time t (number of execution of the updating cycle of ASNM) for having the average number of violations at time t, $\mathbf{E}[Z(t)]$ less then one. Remember that the number of violations is an integer number in the range $[0 \cdots |\mathcal{C}_P|]$ and that we want to derive a time $\tilde{t}$ such that for every $t > \tilde{t}$ it holds $\mathbf{E}[Z(t)] < 1$, indefinitely. This statement is proved in the following:

Theorem 2. *Let $Z(t)$ be the random variables defined in Theorem 1. For every problem $\mathcal{P}$ satisfying the condition of Sec. 2 such that $m = |\mathcal{C}_P|$ and for each input parameter $\alpha > 1$, it holds that*

$$\boldsymbol{E}[Z(t)] < 1 \quad \textit{when } t > \tilde{t} = m(\alpha + \ln m).$$

Proof. Let $\mathcal{P}$ be a maximization problem with $m = |\mathcal{C}_P|$, $\mathbf{x}(t)$ be the state found by the algorithm ASNM at time t and let $Z(t) = \sum_{(x_i,x_j)\in\mathcal{C}_P} Z_{ij}(t)$ be the random variable denoting the number of violations at the same time, defined in Theorem 1. By definition and by applying linearity, the expectation $\mathbf{E}[Z(t)]$ of the sum of variables $Z_{ij}(t)$ can be bounded as follows:

$$\begin{aligned}\mathbf{E}[Z(t)] &= \mathbf{E}\left[\sum_{(x_i,x_j)\in\mathcal{C}_P} Z_{ij}(t)\right] \\ &= \sum_{(x_i,x_j)\in\mathcal{C}_P} \mathbf{E}[Z_{ij}(t)] \\ &\leq \sum_{(x_i,x_j)\in\mathcal{C}_P} \max\{\phi(h_i(\mathbf{x}(t))), \phi(h_j(\mathbf{x}(t)))\} \\ &= \sum_{k=1}^{m} \psi_k,\end{aligned}$$

where $\psi_k = \max\{\phi(h_i(\mathbf{x}(t))), \phi(h_j(\mathbf{x}(t)))\}$ by definition.

To obtain $\mathbf{E}[Z(t)] < 1$, that is less than one violation in mean, is equivalent to impose

$$\max_{k\in[1\cdots m]} \psi_k = \phi(h_p(\mathbf{x}(t))) \leq \frac{1}{m},$$

for some $p \in [1 \cdots m]$. This implies:

$$\frac{1}{1 + e^{-\alpha h_p(\mathbf{x}(t))}} \leq \frac{1}{m},$$

and then

$$-\sum_{j\in\mathcal{N}(p)} c\gamma_{pj}(t)x_j(t) \geq \ln(m-1) + \alpha.$$

The worst case hypothesis implies:

- only one variable $x_q \in \mathcal{N}(p)$ is different from zero
$$\gamma_{pq}(t) \geq \ln(m-1) + \alpha,$$
where $\gamma_{pq}(t)$ is the number of upgradings of constraint $(p,q) \in \mathcal{C}_P$ inside the cycle;
- there is exactly only one updating for each execution of the cycle.

Under these assumptions we can conclude that:

$$\tilde{t} = m(\alpha + \ln m),$$

as stated by theorem. □

5. Experimental Results

To give an idea of the performances (solution quality and computation time) of the ASNM heuristic we chose two problems from the computational graph theory: one of maximization, MAX CLIQUE and one of minimization, MIN VERTEX COVER (see Sec. 3 for an example). For both the problems we consider randomly generated graphs of various size. We compare the ASNM heuristic with a plethora of heuristics and approximation algorithms given for the two problems and already presented in literature.

Regarding the experiments, a fundamental parameter which deserves a discussion is the parameter α because of its dependencies on the quality of ASNM performances. The simulations stress the fact that small values of α w.r.t. n ($\alpha \ll \frac{n}{10}$) give fast execution time, but poor performances in terms of solution quality. On the contrary, according to the theoretical results, large values of α ($\alpha \gg \frac{n}{10}$) cause large execution time without significantly increasing the performances. This behavior has been observed in the following experiment involving instances randomly generated for MIN VERTEX COVER: for various values of n and p ($10 \leq n \leq 500$, $p = 0.1, 0.5, 0.9$), 30 p-random graphs have been generated and fixed; for all α ($1 \leq \alpha \leq n$) the average size of the vertex cover found by ASNM has been computed. As an example, in Fig. 5 results for $n = 100$ and $p = \frac{1}{2}$ are reported.

Apparently, there is not a clear rule that defines the best values to assign to the parameter α, we can only turn to empirical laws that find a compromise between time and quality. Because of its not deterministic character, we execute the heuristic many times (10 in general) picking then the better value carried out, a typical practice which exploits the intrinsic randomness of the algorithm making it possible to better explore the solution space.

5.1. *First experiment*

The MAX CLIQUE instances taken into account here are p-random graphs. They are represented by the pair $\langle V, E\rangle$, where $V = \{1, \ldots, n\}$ and E is obtained selecting $\{i, j\}$ as edge with probability p ($1 \leq i < j \leq n$). To show the behavior of the algorithm on these instances we give a direct comparison with the meta-heuristic IHN presented and tested in Ref. 5, which is in some sense the deterministic version of ASNM. IHN builds a finite sequence of discrete Hopfield networks in which the energy function of the $(t+1)$-th network is the energy function of the t-th network augmented by a penalty factor depending on the violations. We choose such a meta-heuristic

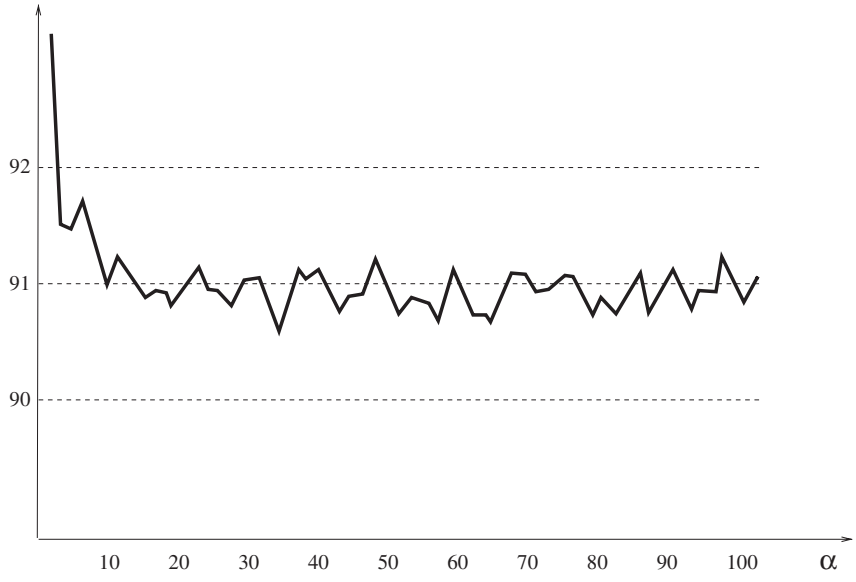

Fig. 5. Dependence of the performances of ASNM on parameter α.

Table 1. Clique average size with relative standard deviation at confidence level 95% obtained by ASNM and IHN (columns 4 and 5) on p-random graphs for various values of n and p (column 1). Column 2 gives the expected size $C(n,p)$ of the maximum clique while column 3 reports the average time (in seconds) spent by ASNM to converge.

n-p	$C(n,p)$	Av. time	Average ± stdev	
			ASNM	IHN
200-0.2	6.2	2.15	5.76 ± 0.07	5.53 ± 0.09
200-0.5	11.6	0.96	10.73 ± 0.09	10.23 ± 0.11
200-0.8	26.6	0.25	24.13 ± 0.12	22.96 ± 0.12
400-0.2	7.0	29.14	6.20 ± 0.07	6.06 ± 0.04
400-0.5	13.3	13.84	12.30 ± 0.08	11.83 ± 0.10
400-0.8	31.7	3.48	28.70 ± 0.14	26.80 ± 0.17
600-0.2	7.4	135.7	6.96 ± 0.03	6.76 ± 0.07
600-0.5	14.2	69.48	13.10 ± 0.05	12.53 ± 0.10
600-0.8	34.6	17.11	31.40 ± 0.10	29.83 ± 0.14

because its performances have been already shown good in Ref. 5 and 15, in which it was compared with many other well known heuristics for MAX CLIQUE presented at DIMACS challenge.[22] Also in this case we set the parameter $\alpha = 10$, run the algorithm 10 times for each instance graph and take the best value obtained.

In Table 1 (column 4) we report, for some values of n and p, the clique average size (on 30 graphs randomly generated for each pair n and p) found by ASNM at confidence level 95%. Observe that the average values found by ASNM are always better than those found by IHN (column 5). These results are also compared with the theoretical evaluation of the expected maximum clique for p-random graphs of size n, obtained according to an asymptotic result[a] (column 2).

Summarizing, on random graphs the clique sizes found by ASNM are always better than those found by IHN and they are at least 90% of the optimal estimate $C(n,p)$.

5.2. *Second experiment*

In this simulation the behavior of ASNM is experimentally analyzed and compared with that of other heuristics for MIN VERTEX COVER on p-random graphs.

More precisely, the ASNM algorithm has been compared on random graphs with other heuristics such as MM (Maximal Matching Algorithm),[11] Greedy Algorithm, WG (Weighted Greedy Algorithm),[8] BE (Bar-Yehuda Even Algorithm).[4] The Greedy algorithm repeatedly picks an edge that has not yet been covered, and places one of its endpoints in the current covering set. This algorithm does not achieve any bounded ratio: on the contrary, the modified Greedy algorithm WG obtains ratio 2. The basic idea is to assign weights to the vertices: each time a vertex is placed in the cover, each of its neighbors has its weight reduced by an amount equal to the ratio of the selected vertex current weight and degree. Another heuristic that achieves ratio 2 is MM algorithm: picks a maximal matching M in the graph and place both endpoints of edges in M into the cover. The last heuristic we consider is BE, obtained starting from a relaxation of an integer programming formulation of min vertex cover; this algorithm achieves a ratio $2 - \frac{\log\log n}{2\log n}$.

All these algorithms have been compared both regarding the quality of solutions and the computation time. For each instance size we generated 30 random graphs and executed the algorithms on each one. We report here the average results

Table 2. Results of the simulations of ASNM and others algorithms for Min Vertex Cover problem.

n	Algorithms				
	ASNM	Greedy	WG	BE	MM
100	68.6	72.7	76.9	91.8	93.6
200	157.3	164.8	172.7	190.6	192.8
300	251.6	260.3	273.3	290.9	293.2
400	348.7	357.8	372.1	390.6	393.6
500	447.4	458.1	475.2	491.1	494.3

[a]The expected number of cliques of size k in a p-random graph of size n is $\binom{n}{k} p^{\binom{k}{2}}$. For p fixed ($0 < p < 1$) and sufficiently large n, let $C(n,p)$ be the real number k such that $\binom{n}{k} p^{\binom{k}{2}} = 1$; it was shown[26] that, for $n \to \infty$, the probability that a p-random graph of size n has a clique of size $C(n,p)$ approaches 0. Therefore, $C(n,p)$ estimates quite well the expected size of the maximum clique.

obtained, while the standard deviation is in general little for all algorithms and no more than 2.33.

As regards the solution quality, the ASNM algorithm performs better than the others, while the worse performances are given by the algorithms MM and BE (see Table 2).

6. Conclusions

We have introduced a model of computation based on adaptivity and stochasticity to explore the solution space in order to find solutions constrained to satisfy a fixed structure or a given property. In the model the pseudo-Boolean functions are used both to express the constraints and to define the cost function of the problem, consequently interpreted as energy function of a neural network with stochastic dynamics. A wide variety of NP-had problems fall in the class of problems that can be solved by the model at hand, particularly those having a pseudo-Boolean constraint function easy (polynomial time) to maximize (or minimize).

The tests carried out show that it is a feasible and a good heuristic when compared with other well-known methods to solve the problems considered. The future work concerns the analysis (worst case) of the performances of the algorithm in terms of computational complexity and approximation properties, because most of the problems recalled here admit an approximation algorithm with guaranteed performance ratio.

References

1. E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing* (John Wiley & Sons, Inc., New York, NY, USA, 1989).
2. D. H. Ackley, G. E. Hinton and T. J. Sejnowski, A learning algorithm for boltzmann machines, *Cognitive Science* **9** (1985).
3. H. Adeli and H. S. Park, Optimization of space structures by neural dynamics, *Neural Networks* **8**(5) (1995) 769–781.
4. R. Bar-Yehuda and S. Even, A local-ratio theorem for approximating the weighted vertex cover problem, *Annals of Discrete Mathematics* **25** (1985) 27–45.
5. A. Bertoni, P. Campadelli and G. Grossi, A neural algorithm for the maximum clique problem: Analysis, experiments and circuit implementation, *Algoritmica* **33**(1) (2002) 71–88.
6. E. Boros and P. L. Hammer, Pseudo-boolean optimization, *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science* **123** (2002).
7. Y. Y. Chen and K. Y. Young, An som-based algorithm for optimization with dynamic weight updating, *International Journal of Neural Systems* **17**(3) (2007) 171–181.
8. K. L. Clarkson, A modification of the greedy algorithm for vertex cover, *Information Processing Letters* **16** (1983) 23–25.
9. G. Galán-Marín, E. Mérida-Casermeiro and J. Munoz-Pérez, Modelling competitive hopfield networks for the maximum clique problem, *Comput. Oper. Res.* **30**(4) (2003) 603–624.
10. M. R. Garey and D. S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness* (W. H. Freeman & Co., San Francisco, CA, 1979).
11. F. Gavril, Quoted in Ref. 10, p. 134.
12. R. J. Glauber, Time-dependent statistics of the Ising model, *Journal of Mathematical Physics* **4**(2) (1963) 294–307.
13. G. Grossi, A discrete adaptive stochastic neural model for constrained optimization, in *16th International Conference on Artificial Neural Networks (ICANN '06)*, Springer (2006), pp. 641–650.
14. G. Grossi and F. Pedersini, FPGA implementation of a stochastic neural network for monotonic pseudo-boolean optimization, *Neural Networks* **21**(6) (2008) 872–879.
15. 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 '02)*, Springer-Verlag (2002), pp. 64–74.
16. P. L. Hammer, I. Rozemberg and S. Rudeanu, Application of discrete linear programming to the minimization of boolean functions, *Rev. Mat. Pures App.* **8** (1963) 459–475.
17. P.L. Hammer and S. Rudeanu, *Boolean Methods in Operations Research and Related Areas* (Springer-Verlag, 1968).
18. J. Hertz, A. Krogh and R. G. Palmer, *Introduction to the Theory of Neural Computation* (Addison-Wesley, 1991).
19. J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences of the United States of America* **79**(8) (1982) 2554–2558.
20. J. J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons, in *Proceedings of the National Academy of Sciences* **81** (NAS, 1984) 3088–3092.
21. J. J. Hopfield and D. W. Tank, Neural computation of decision in optimization problems, *Biological Cybernetics* **52** (1985) 141–152.

22. 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).
23. R. M. Karp, Reducibility among combinatorial problems, Complexity of Computer Computations (Plenum Press, New York, 1972), pp. 85–103.
24. S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, Optimization by simulated annealing, *Science* **220** (1983) 671–680.
25. P. J. M. Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications, Mathematics and Its Applications* (Reidel Publisching Company, 1987).
26. D. Matula, On the complete subgraph of a random graph, *Combinatory Mathematics and Its Applications* (1970), pp. 356–369.
27. R. V. Mayorga and M. Arriaga, Non-linear global optimization via parameterization and inverse function approximation: An artificial neural networks approach, *International Journal of Neural Systems* **17**(5) (2007) 353–368.
28. G. Serpen, Hopfield network as static optimizer: Learning the weights and eliminating the guesswork, *Neural Process. Lett.* **27**(1) (2008) 1–15.
29. K. Smith, Neural networks for combinatorial optimization: A review of more than a decade of research, *INFORMS Journal on Computing* **11** (1999) 15–34.
30. Y. Takefuji, *Neural Network Parallel Computing* (Kluwer Academic Publishers, Norwell, MA, USA, 1992).
31. J. Wang and Y. Zhou, Stochastic optimal competitive hopfield network for partitional clustering, *Expert Systems with Applications* **36** (2009) 2072–2080.
32. R. L. Wang, P. Zhang and K. Okazaki, An efficient learning algorithm of the hopfield neural network for the minimum set cover problem, *IJCSNS International Journal of Computer Science and Network Security* **6**(9) (2007) 28–31.