

Energy Function-Based Approaches to Graph Coloring

Andrea Di Blas, *Member, IEEE*, Arun Jagota, and Richard Hughey, *Member, IEEE*

Abstract—We describe an approach to optimization based on a multiple-restart quasi-Hopfield network where the only problem-specific knowledge is embedded in the energy function that the algorithm tries to minimize. We apply this method to three different variants of the graph coloring problem: the minimum coloring problem, the spanning subgraph k -coloring problem, and the induced subgraph k -coloring problem. Though Hopfield networks have been applied in the past to the minimum coloring problem, our encoding is more natural and compact than almost all previous ones. In particular, we use k -state neurons while almost all previous approaches use binary neurons. This reduces the number of connections in the network from $(Nk)^2$ to N^2 asymptotically and also circumvents a problem in earlier approaches, that of multiple colors being assigned to a single vertex. Experimental results show that our approach compares favorably with other algorithms, even nonneural ones specifically developed for the graph coloring problem.

Index Terms—Energy functions, energy minimization, graph coloring, Hopfield neural networks, optimization, simulated annealing.

I. INTRODUCTION

GIVEN an undirected graph $G = \{V, E\}$, composed of a set of $N = |V(G)|$ vertices, and a set of $M = |E(G)|$ edges, with $V = \{v_i, i = 1 \dots N\}$ and $E = \{e_{ij} | v_i \text{ and } v_j \text{ are connected}\}$, a k -coloring of G is a partition of V into k sets (color classes) $V_1 \dots V_k$ such that $\forall e_{ij}, v_i \in V_l \Rightarrow v_j \notin V_l$. The chromatic number $\chi(G)$ is the smallest value of k such that G is k -colorable.

Finding the chromatic number is referred to as the *minimum coloring* problem, and is known to be NP-hard [14]. The graph coloring problem is interesting not only because it is computationally intractable, but also because it has numerous practical applications, for instance: register assignment in code compilation [15], several CAD problems [8], [13], timetabling [9], and frequency channel assignment [37], [39]. For these reasons, the graph coloring problem has been addressed by a number of approaches over the past few decades. However, no one approach has turned out to be a clear winner.

In recent years, neural networks have been applied to this problem, mostly based on Hopfield or Hopfield-like networks that employ binary neurons. k neurons per vertex have been used to represent the k colors that a vertex may be assigned, in-

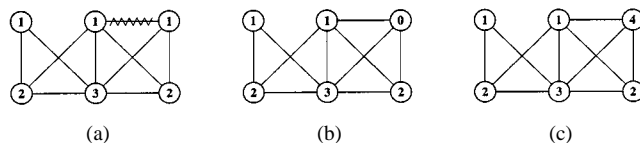


Fig. 1. Spanning subgraph k coloring (a) and induced subgraph k coloring (b) with $k = 3$, and minimum coloring (c) that requires $k = 4$. Numbers in the nodes are colors, with 0 meaning uncolored. The broken line marks an improperly colored edge.

creasing the network size (the number of connections) to $(Nk)^2$. In principle $\log k$ neurons would suffice, but no one has come up with such an encoding. Our k -state neuron approach originates from the idea that is more natural for a neuron to assume one of the k states that represent the k colors. This eliminates the possibility of conflicts among neurons in the same cluster and allows a more compact network representation with size of the order of N^2 .

The main benefit of our approach in this paper is that it provides a generic methodology for deriving heuristic algorithms for difficult problems from energy functions in a largely problem-independent manner. Using this methodology on particular energy functions yields particular heuristic algorithms, some novel. Additionally, like most of the neural-network-based algorithms, ours are intrinsically easy to parallelize [32], especially on SIMD parallel computers [10], [23]. Finally, using these methods for solving difficult optimization problems it is easy to tradeoff solution quality versus execution time.

In this paper, in addition to the minimum coloring (MC) problem [Fig. 1(c)], we also solve two other important variants:

- The *spanning subgraph k -coloring* (SSC) problem is to minimize the number of improperly colored edges while coloring all the vertices in the graph using at most k colors [Fig. 1(a)].
- The *induced subgraph k -coloring* (ISC) problem is to find the largest set of vertices that can be properly colored using k colors, possibly leaving some vertices in the graph uncolored [Fig. 1(b)].

We will also use SSC and ISC to approximately solve minimum coloring by using a binary search to select the number of colors k .

This paper is organized as follows: after a brief survey of the previous work in Section II, we describe the quasi-Hopfield optimizing neural network approach in Section III and propose and analyze three simple energy functions for the three graph coloring problems in Section IV. Then we detail the algorithm and the implementation in Section V and, finally, we report experimental results on benchmark graphs and a comparison with other methods in Section VI.

Manuscript received November 27, 2000; revised July 27, 2001. This work was supported in part by the National Science Foundation under Grants MIP-9423 985 and EIA-9 722 730.

The authors are with the Department of Computer Engineering, Baskin School of Engineering, University of California, Santa Cruz, CA 95064 USA (e-mail: andrea@cse.ucsc.edu).

Publisher Item Identifier S 1045-9227(02)00365-X.

II. RELATED WORK

After the work by Hopfield and Tank [19], [20] several researchers have addressed the issue of mapping optimization problems onto neural networks. In particular, a number of solutions have been proposed to solve the graph coloring problem.

The earliest efforts were focused on solving the four-colorability problem of planar graphs. Dahl [7] builds a neural network where k neurons corresponding to the k colors are assigned to each of the N regions in the map. An energy function consisting of three terms is then minimized using a gradient descent dynamic. In order to avoid local minima and improper coloring, the value of three parameters requires adjustment according to the map size.

Philipsen and Stok [36] transform the graph coloring problem into the maximum weight stable set problem, building an augmented graph with a clique of d vertices (where d is the maximum degree of the graph plus one) for each vertex of the original graph. To enforce feasibility of the solution, they use a previous approach proposed by Peterson and Soderberg [34] based on the mean field theory Potts neuron.

Takefuji and Lee [40] use a Hopfield network with binary neurons to solve the four and k -coloring problem of planar geographical maps. Their algorithm was later modified by Gassen and Carothers [15] to deal with the more general problem of generic graph k -coloring, applied to register allocation optimization. Their modification includes a minimization term to reduce the number of colors used. Another extension to the k -coloring problem of the Takefuji and Lee algorithm was later devised by Berger [1], who modified the energy function in order to guarantee convergence without resorting to the heuristic trimming of parameters necessary to the original algorithm.

Later, Jagota [22] proposed a reduction of the coloring problem to the maximum independent set problem, followed by a mapping onto a Hopfield network. The network would then evolve using a dynamic algorithm called *discrete descent*, that had been used in earlier works to solve the maximum clique problem [21], [23], and that is also adapted to the modified approach presented in this paper.

An interesting network-based algorithm that also encodes the color of a vertex in the state of a single cell is described in the work by Wu [41]. An array of coupled oscillators is considered as a graph by associating the oscillators to the vertices and couplings to edges. When the array is synchronized, the phase in each oscillator can be considered as the color of the corresponding vertex.

There are several other approaches to the graph coloring problem that do not make use of neural networks [2]–[4], [6], [11], [12], [16], [17], [35], [24], [26]–[28], [31], [33], [38]. The last three of these, which we believe represent the state-of-the-art, will be briefly described in Section VI and compared with our results.

III. THE NEURAL NETWORK

Here we propose a *multivalued* neural network, where the state S_i of each neuron can assume one of k or $k + 1$ different values, that is $S_i \in \{1, \dots, k\}$ or $S_i \in \{0, 1, \dots, k\}$.

Given an undirected graph G we build a corresponding neural network composed of N nodes. An adjacency matrix has elements $A_{ij} = 1$ if $\{v_i, v_j\} \in E$, 0 otherwise. Each node i has an output that corresponds to its state S_i , and receives as input the states of all its neighboring nodes. A specific energy function $E(\vec{S})$ is defined, based on the specific problem. The goal of the algorithm is to minimize the energy.

We will update the network serially, specifically at time $t + 1$ a single neuron i will be selected and its state $S(t + 1)$ set to a new value c such that

$$\Delta E_i(c, t) \equiv E(S_i(t + 1) = c) - E(S_i(t)) < 0. \quad (1)$$

Thus, the energy will decrease in every time step. So long as the energy function is bounded from below, the network will always converge to a (local) minimum, that is, the network is *globally stable*.

IV. ENERGY FUNCTIONS

In this section we propose an energy function for each of the three graph coloring problems mentioned above. For each of them, we give the energy difference according to (1). We then analyze these energy functions to show that they exhibit some desirable properties. In particular we show that the solutions found are feasible and locally optimal, and that lower energy minima yield better solutions.

For the *spanning subgraph k -coloring* problem we propose the energy function

$$E_{\text{SSC}}(\vec{S}) = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N A_{ij} \delta(S_i, S_j) \quad (2)$$

where $\vec{S} \in \{1, \dots, k\}^N$ denotes a color assignment to all vertices of G from the set of colors $\{1, \dots, k\}$. The function $\delta()$ is a Kronecker delta function (in a slightly different notation) defined as

$$\delta(a, b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$$

with a, b integers. Here we see that the energy function in (2) simply represents the number of *violations*, that is, the number of improperly colored edges. Since $E_{\text{SSC}} \geq 0$, for any given k a (local) minimum $E_{\text{SSC}} > 0$ (locally) minimizes the number of violations and $E_{\text{SSC}} = 0$ corresponds to a properly colored graph.

For the *induced subgraph k -coloring* problem we propose the energy function

$$E_{\text{ISC}}(\vec{S}) = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N A_{ij} \text{sgn}(S_i S_j) \delta(S_i, S_j) - \gamma \sum_{i=1}^N \text{sgn}(S_i) \quad (3)$$

where it is important to note that now $\vec{S} \in \{0, \dots, k\}^N$, where $S_i = 0$ will denote that vertex i is not colored. The function $\text{sgn}()$ is the sign function, defined as

$$\text{sgn}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0. \end{cases}$$

The (first) constraint term of (3) represents the number of improperly colored edges. We introduce the $\text{sgn}()$ term so that uncolored vertices do not generate improperly colored edges. The (second) objective term represents the number of colored vertices, regardless of their color value. Minimizing this energy function, therefore, means trying to minimize the number of improperly colored edges and at the same time trying to maximize the number of colored nodes. A balance parameter $\gamma > 0$ controls the tradeoff between these two terms.

For the *minimum coloring* problem we propose the energy function

$$E_{\text{MC}}(\vec{S}) = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N A_{ij} \delta(S_i, S_j) + \gamma \sum_{i=1}^N S_i \quad (4)$$

where $\vec{S} \in \{1, \dots, k\}^N$. Note that uncolored nodes are not allowed. Also note that k has to be sufficiently large so that a proper coloring of the entire graph is feasible; for instance, we can set $k = \Delta(G) + 1$, where $\Delta(G)$ is the maximum degree in the graph. Here $\gamma > 0$ balances the constraint term of minimizing the number of improperly colored edges with the objective term of minimizing the number of colors used.

1) *Energy Difference*: At each time step we compute the energy differences to implement the serial update so as to satisfy (1). We now compute the energy differences for the three energy functions.

For the SSC energy function (2)

$$\Delta E_i(c, t) = \sum_{j=1}^N A_{ij} [\delta(c, S_j(t)) - \delta(S_i(t), S_j(t))]. \quad (5)$$

Here, $\Delta E_i(c, t) < 0$ if and only if assigning color c to node i reduces the number of improperly colored edges among those touching node i .

For the ISC energy function (3)

$$\begin{aligned} \Delta E_i(c, t) = & \sum_{j=1}^N A_{ij} [\text{sgn}(c S_j(t)) \delta(c, S_j(t)) \\ & - \text{sgn}(S_i(t) S_j(t)) \delta(S_i(t), S_j(t))] \\ & - \gamma [\text{sgn}(c) - \text{sgn}(S_i(t))]. \end{aligned} \quad (6)$$

Here, for $c > 0$, $\Delta E_i(c, t) < 0$ if and only if one of the following holds: 1) node i is uncolored and coloring it c increases the number of violations less than γ or 2) node i is colored $c' \neq c$ and recoloring it c reduces the number of violations. For $c = 0$, $\Delta E_i(c, t) < 0$ if and only if node i is colored and uncoloring it reduces the number of violations by more than γ .

For the MC energy function (4)

$$\begin{aligned} \Delta E_i(c, t) = & \sum_{j=1}^N A_{ij} [\delta(c, S_j(t)) - \delta(S_i(t), S_j(t))] \\ & + \gamma [c - S_i(t)]. \end{aligned} \quad (7)$$

Here, $\Delta E_i(c, t) < 0$ if and only if one of the following holds: 1) when $c < S_i(t)$, recoloring node i with c increases the number of violations by less than $\gamma(S_i - c)$ or 2) when $c > S_i(t)$, recoloring node i with c reduces the number of violations by more than $\gamma(c - S_i)$. Notice that in both cases one is trading off quality of the coloring versus the number of improperly colored edges.

2) *Feasibility*: We now show that the local minima of these three energy functions represent only feasible solutions of their associated problems. In the ISC and MC problems we will need to impose specific conditions on the parameter γ for this purpose.

In the case of SSC, *any* coloring, even an improper one, is feasible, as long as all vertices are colored. In our encoding this is always the case because of our choice $\vec{S} \in \{1, \dots, k\}^N$.

In the case of ISC, a feasible coloring is a coloring that has no improperly colored edges. Intuitively, we see that when γ is large, a local minimum does not necessarily correspond to a feasible solution, i.e., some violations may be present, and that when γ is small all local minima correspond to feasible solutions. Here we determine a critical value γ_{ISC}^* such that for $\gamma < \gamma_{\text{ISC}}^*$ every local minimum is feasible. This value is critical in the sense that we will show a local minimum state in a graph with $\gamma = \gamma_{\text{ISC}}^*$ that is not feasible.

Lemma 1: When $\gamma < \gamma_{\text{ISC}}^* = 1$ all minima of the ISC energy function (3) are feasible solutions.

Proof: A state \vec{S} is a minimum of the energy function (3) when no possible serial update can decrease the energy. Therefore, from (6)

$$\begin{aligned} \Delta E_i(c) = & \sum_{j=1}^N A_{ij} [\text{sgn}(c S_j) \delta(c, S_j) - \text{sgn}(S_i S_j) \delta(S_i, S_j)] \\ & - \gamma [\text{sgn}(c) - \text{sgn}(S_i)] \geq 0 \end{aligned} \quad (8)$$

must hold $\forall i \in \{1, \dots, N\}, \forall c \in \{0, \dots, k\}$ (we drop the time dependency for simplicity). Let us rewrite (8) as follows:

$$\Delta E_i(c) = A(c) - B(S_i) - \gamma C(c, S_i) \geq 0 \quad (9)$$

where

$$\begin{aligned} A(c) &= \sum_{j=1}^N A_{ij} \text{sgn}(c S_j) \delta(c, S_j) \\ B(S_i) &= \sum_{j=1}^N A_{ij} \text{sgn}(S_i S_j) \delta(S_i, S_j) \\ C(c, S_i) &= \text{sgn}(c) - \text{sgn}(S_i). \end{aligned}$$

Here, $A(c)$ represents the number of edges among those touching node i that would be colored improperly if node i were colored c , $B(S_i)$ the number of edges among those

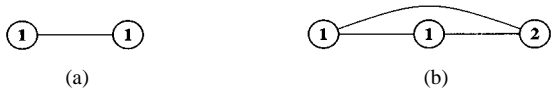


Fig. 2. An infeasible coloring that is a local minimum for the ISC energy function when (a) $k = 1$ and $\gamma = \gamma_{\text{ISC}}^* = 1$ and (b) an infeasible coloring that is a local minimum for the MC energy function when $\gamma = \gamma_{\text{MC}}^* = 1/2$.

touching node i that are colored improperly in the present state, and $C(c, S_i)$ represents the change in the total number of colored nodes that would occur if node i were (re)colored c . For $c = 0$, $A(c) = 0$ hence, from (9)

$$B(S_i) \leq -\gamma C(0, S_i). \quad (10)$$

Since $C(0, S_i) \in \{0, -1\}$, (10) implies $B(S_i) \leq \gamma$. Now, if $\gamma < \gamma_{\text{ISC}}^* = 1$ then $B(S_i) = 0$ since it is an integer, and there are no violations in \vec{S} . \square

Thus, we have shown that when $\gamma < 1$ every state \vec{S} which is a local minimum is feasible. To prove that this value is critical, we need simply offer an example: a graph G is two nodes connected by an edge, and $k = 1$ [Fig. 2(a)]. If $\gamma = \gamma_{\text{ISC}}^* = 1$, then a state in which both nodes are colored 1 is a local minimum but is infeasible. If $\gamma < 1$, on the other hand, the network will converge to a state where one of the nodes is uncolored and the coloring is feasible.

A feasible solution of the MC problem requires all nodes to be colored properly. Once again, it is easy to see that when γ is large, a local minimum will not necessarily be feasible, and when γ is sufficiently small all local minima are feasible.

As in the ISC case, we can compute a critical value γ_{MC}^* such that if $\gamma < \gamma_{\text{MC}}^*$ we are guaranteed that every local minimum is a feasible solution.

Lemma 2: When $\gamma < \gamma_{\text{MC}}^* = 1/\Delta(G)$ all minima of the MC energy function (4) are feasible solutions.

Proof: This proof is similar to that of Lemma 1. In a local minimum the following holds from (7): $\forall i \in \{1, \dots, N\}, \forall c \in \{1, \dots, k\}$

$$\Delta E_i(c) = A(c) - B(S_i) + \gamma C(c, S_i) \geq 0 \quad (11)$$

where

$$A(c) = \sum_{j=1}^N A_{ij} \delta(c, S_j),$$

$$B(S_i) = \sum_{j=1}^N A_{ij} \delta(S_i, S_j)$$

and

$$C(c, S_i) = c - S_i. \quad (12)$$

Here too, $A(c)$ represents the number of edges among those touching node i that would be colored improperly if node i were colored c , $B(S_i)$ the number of edges among those touching node i that are colored improperly in the present state, and $C(c, S_i)$ is the change in the objective function.

In a local minimum, the graph will be colored with colors from 1 to k' , with $k' \leq \Delta(G) + 1$. Therefore,

$\forall i, \exists c' \in \{1, \dots, \Delta(G) + 1\}$ such that $A(c') = 0$, and (11) implies

$$B(S_i) \leq \gamma C(c', S_i). \quad (13)$$

Since $C(c', S_i) \leq \Delta(G)$, if $\gamma < \gamma_{\text{MC}}^* = 1/\Delta(G)$ then $B(S_i) = 0$. \square

As in the ISC case, the value $\gamma_{\text{MC}}^* = 1/\Delta(G)$ is critical. As an example, consider a three-node graph completely connected [Fig. 2(b)], where $\Delta(G) = 2$ and $\gamma_{\text{MC}}^* = 1/2$. A state in which two nodes are colored 1 and the third node is colored 2 has one violation. However, if $\gamma = \gamma_{\text{MC}}^*$, this state is a local minimum. On the other hand, if $\gamma < \gamma_{\text{MC}}^*$, the energy function will converge to a feasible state where either one of the nodes that are colored 1 is recolored 3.

3) Local Optimality: Feasible solutions can be arbitrarily poor. For instance, in SSC a graph where all nodes are assigned the same color is feasible under the above definition, or, for ISC, a completely uncolored graph is feasible. Clearly these solutions are poor. However, it will turn out that the local minima of our energy functions will not only be feasible, but also locally optimal according to the definitions below. First we will define a state \vec{S}' to be a *neighbor* of state \vec{S} if $\vec{S}, \vec{S}' \in \{k_1, \dots, k_2\}^N$ and $\exists! i | S_i \neq S'_i$, that is if \vec{S} and \vec{S}' differ in exactly one component.

Definition 1: A minimal-violations k -coloring is a (not necessarily proper) coloring $\vec{S} \in \{1, \dots, k\}^N$ of all nodes in G such that no coloring \vec{S}' that is a neighbor of \vec{S} has fewer violations.

Definition 2: A maximal k -coloring is a coloring of G in which a subset $C \subseteq V$ is properly colored and this coloring is not extendible, that is $\forall v \in V - C, \exists c \in \{1, \dots, k\}$ such that $C \cup \{v \text{ colored } c\}$ is a proper coloring.

Definition 3: A minimal coloring is a proper coloring $\vec{S} \in \{1, \dots, k\}^N$ of all the nodes in G such that for every proper coloring \vec{S}' that is a neighbor of \vec{S} , $n(\vec{S}') \geq n(\vec{S})$, where $n(\vec{S})$ is the number of colors used in state \vec{S} .

From the SSC energy function (2) it follows immediately that every local minimum yields a minimal-violations k coloring. Similarly, in the case of ISC, when $\gamma < \gamma_{\text{ISC}}^*$, every local minimum of the energy function (3) yields a maximal k -coloring.

It turns out that also in the case of MC, when $\gamma < \gamma_{\text{MC}}^*$, local minima of the energy function (4) are minimal colorings. However, this is not immediately intuitive to see from the energy function definition as in the SSC and ISC cases. Therefore we have the following.

Lemma 3: Every local minimum of the MC energy function (4) with $\gamma < \gamma_{\text{MC}}^*$ is a minimal coloring.

Proof: Let \vec{S} be a local minimum with $\gamma < \gamma_{\text{MC}}^*$. Let $n_c(\vec{S})$ denote the number of nodes colored c in state \vec{S} . We will prove by contradiction that \vec{S} is a minimal coloring. Let \vec{S}' be a neighbor of \vec{S} that is also a proper coloring, and let us suppose that $n(\vec{S}') < n(\vec{S})$. Let i be the component in which \vec{S}' differs from \vec{S} . Let $S_i = c$ and $S'_i = c' \neq c$. Since $n(\vec{S}') < n(\vec{S})$, $n_c(\vec{S}) = 1$. Since \vec{S} is a local minimum, all colors of value less than c are represented in $N(i)$, where $N(i)$ is the set of nodes connected by an edge to node i . Now, consider any color $d > c$ used in \vec{S} . Any node j colored d is also

in $N(i)$ (if not, then j could be recolored c , which would imply that \vec{S} is not a local minimum). Hence, every color used in \vec{S} except c is represented in $N(i)$. Thus $S'_i = c' \neq c$ would imply \vec{S}' has violations, which contradicts assumption. \square

4) *Representability*: We have seen that all local minima for the three problems yield locally optimal solutions. It turns out that for the SSC and ISC problems the converse is also true—every locally optimal coloring is represented as a local minimum of the energy function. We call this property *representability*. Specifically, we note the following.

- Every *minimal-violations k -coloring* is represented in some local minimum of the SSC energy function.
- Every *maximal k -coloring* is represented in some local minimum of the ISC energy function with $\gamma < \gamma_{\text{ISC}}^*$

On the other hand, it is not true that every minimal coloring is represented in a local minimum of the MC energy function with $\gamma < \gamma_{\text{MC}}^*$. For instance a graph with two nodes connected by an edge where the nodes are colored one and three is a minimal coloring but not a local minimum.

5) *Order Preservation*: In mapping an optimization problem to an energy function, another desirable property is that lower minima of the energy function yield strictly better solutions of the problem, i.e., $E(\vec{S}) < E(\vec{S}') \Rightarrow Q(\vec{S}) > Q(\vec{S}')$ where \vec{S} and \vec{S}' are two local minima states and $Q(\vec{S})$ is the quality of solution \vec{S} . A feasible mapping with this property is said to be *order-preserving*.

One reason why the order-preserving property is a useful one is that if a mapping is feasible, representable, and order-preserving, then global optima of the energy function correspond to globally optimal solutions of the mapped problem. An even more practical value of the order-preserving property is based on the fact that one is rarely able to find an optimal solution, especially when the problem is NP-hard. Thus, since one will usually go only for local energy minima, the order-preserving property implies that deeper minima yield better solutions, while a mapping that preserves global optima might not be a useful one unless it is also order-preserving.

We now discuss the three mappings from this perspective. First we need to define a measure of solution quality. For the SSC problem we will define this as the number of properly colored edges, for the ISC problem as the number of colored nodes, and for the MC problem as the negative of the number of distinct colors used.

It can be easily seen that both the SSC and the ISC mappings are order-preserving. Unfortunately, the MC mapping (4) is *not* order-preserving since lower energies do not necessarily correspond to colorings that use fewer colors (Fig. 3).

An idealized variant of this mapping, however, is *almost order-preserving* in the following sense: $E(\vec{S}) < E(\vec{S}') \Rightarrow Q(\vec{S}) \geq Q(\vec{S}')$ for any two states \vec{S}, \vec{S}' . Specifically, we generalize the MC energy function (4) to

$$E_{\text{MC}}(\vec{S}) = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N A_{ij} \delta(S_i, S_j) + \gamma \sum_{c=1}^k w_c n_c(\vec{S}) \quad (14)$$

where w_c denotes the weight of color c and $n_c(\vec{S})$ is the number of components colored c in \vec{S} . With $w_c = c$, (14) reduces to (4).

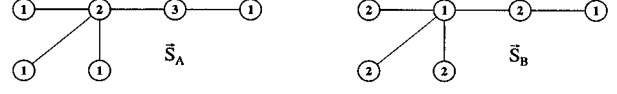


Fig. 3. Example of nonorder-preservation (not even almost order-preservation) for MC: both states are minima with $E(\vec{S}_A) = 9\gamma < E(\vec{S}_B) = 10\gamma$, but $Q(\vec{S}_A) < Q(\vec{S}_B)$ since \vec{S}_A has three colors and \vec{S}_B has two.

Lemma 4: With $w_c = N^c$ a feasible mapping of MC with regard to (14) is almost order preserving.

Proof by Contradiction: In a feasible mapping, let \vec{S} and \vec{S}' be two local minima, and suppose $E(\vec{S}) < E(\vec{S}')$ (if no such minima exist, the mapping is trivially order-preserving). Let k and k' represent the highest color values used in \vec{S} and \vec{S}' , respectively, and suppose $k > k'$. Then

$$N^{k'+1} \leq \sum_{c=1}^k N^c n_c(\vec{S})$$

since $n_{k'+1} \geq 1$ and since there are no gaps in the coloring \vec{S} . Furthermore,

$$\sum_{c=1}^{k'} N^c n_c(\vec{S}') \leq N^{k'+1}.$$

Finally, since $E(\vec{S}) < E(\vec{S}')$ and both minima are feasible

$$\sum_{c=1}^k N^c n_c(\vec{S}) < \sum_{c=1}^{k'} N^c n_c(\vec{S}').$$

We have a contradiction. \square

Earlier, Korst and Aarts [29] encoded the MC problem to a Boltzmann machine with binary neurons (we use multivalued neurons). They obtained an analog of Lemma 4 in which they defined w_c recursively in terms of w_1, \dots, w_{c-1} . It turns out that Lemma 4 is asymptotically equivalent to their condition, while the proof above is simpler. Neither mapping is practical, though, since the weights w_c grow exponentially with c . Like Korst and Aarts, we will take a pragmatic approach. In particular, we will go back to our original encoding (4) with $w_c = c$. We will see that the network using (4) finds good solutions, comparable in quality to the ones found by state-of-the-art methods.

V. THE ENERGY MINIMIZATION ALGORITHM

Here we describe a generic energy minimization algorithm applicable to the mapping of all three problems.

The algorithm has two nested loops, an inner loop and an outer loop. The inner loop starts from a certain initial state and serially updates states until a minimum is reached. The outer loop repeatedly restarts the inner loop from different, randomly chosen, initial states and outputs the best solution found in any restart. The value of the outer loop is in compensating for the limited optimization ability of the inner loop by using multiple restarts.

In the case of the ISC and MC problems, an intermediate loop will also perform an *annealing* of the parameter γ , interpreted as a temperature. Annealing will permit occasional uphill moves to escape local minima.

```

inner_loop( $\vec{S}, k, \gamma$ )
{  $\Delta\vec{E} \leftarrow \text{compute\_DeltaE}(\vec{S}, k, \gamma)$ ;           // compute initial  $\Delta\vec{E}$ 
   $(i, c) = F(\Delta\vec{E})$ ;                               // pick a first vertex-color pair
  while( $i \neq 0$ )                                     // while an energy decrease is possible
  {  $old\_c \leftarrow S_i$ ;  $S_i \leftarrow c$ ;           // assign new color  $c$  to node  $i$ 
    update_DeltaE( $i, c, old\_c, \Delta\vec{E}, \gamma$ );       // update  $\Delta\vec{E}$ 
     $(i, c) = F(\Delta\vec{E})$ ;                             // pick next vertex-color pair
  }
  return( $\vec{S}$ );
}

```

Fig. 4. The inner loop.

1) *The Inner Loop*: At each time step, the color of one node is changed to reduce energy. When no change can further reduce energy, a local minimum is reached and the loop ends. At each time step, the node i whose color is to be changed and the new color c are picked as

$$(i, c) = F(\Delta\vec{E}(t)) \quad (15)$$

where $\Delta\vec{E}(t)$ is the matrix of $\Delta E_i(c, t)$ for all pairs (i, c) and F is any function of this matrix that returns *some* neuron-color pair (i, c) satisfying $\Delta E_i(c, t) < 0$ when this is possible and $(0, 0)$ when this is not. When $i > 0$, recoloring node i with c will necessarily reduce energy. The pseudo-code of the inner loop is shown in Fig. 4.

We can make different choices of F to embody different heuristics. Note that all will reduce energy. Let $D = \{(i, c) \mid \Delta E_i(c, t) < 0\}$. Reasonable choices of F include:

- *greedy*: F returns the pair $(i, c) \in D$ with the minimum value of $\Delta E_i(c, t) \in D$;
- *random*: F returns any random pair $(i, c) \in D$ with equal probability;
- *probabilistic-greedy*: F returns a pair $(i, c) \in D$ with probability $(|\Delta E_i(c, t)|) / (\sum_{(i', c') \in D} |\Delta E_{i'}(c', t)|)$.

In all cases, F will return $(0, 0)$ if D is empty.

The subroutine `update_DeltaE()`, rather than computing $\Delta\vec{E}$ from scratch as does `compute_DeltaE()`, creates the new $\Delta\vec{E}$ by updating the old $\Delta\vec{E}$. This results in a significant speedup over `compute_DeltaE()`. We will distinguish two cases: 1) update $\Delta\vec{E}_i(t)$, the row of $\Delta\vec{E}(t)$ corresponding to node i that has just changed color, and 2) update $\Delta\vec{E}_j(t)$ for all nodes j adjacent to node i in G . Rows in $\Delta\vec{E}(t)$ of nonneighbors of i will not need updating.

Let us first consider case 1). Our goal is to compute $\Delta E_i(c', t)$ for all possible colors c' . Rather than recomputing all values from scratch, we can compute them as the sum of their previous value (at time $t-1$) and the variation caused by the recoloring of node i at time t , the *delta energy variation*, as in

$$\Delta E_i(c', t) = \Delta E_i(c', t-1) + \Delta^2 E_i(c', t). \quad (16)$$

Computing $\Delta\vec{E}_i(t)$ therefore reduces to computing $\Delta^2 E_i(c', t)$ for all colors c' . From (16)

$$\Delta^2 E_i(c', t) = \Delta E_i(c', t) - \Delta E_i(c', t-1).$$

To compute this delta energy variation for SSC, ISC, and MC we consider that $S_j(t-1) = S_j(t)$, since only node i has been recolored. In the following equations, we use $S_i(t)$ rather than

c to indicate the color of node i at time t , to avoid confusion. For the SSC energy function, from (5)

$$\Delta^2 E_i(c', t) = \sum_{j=1}^N A_{ij} [\delta(S_i(t-1), S_j(t)) - \delta(S_i(t), S_j(t))].$$

For the ISC energy function, from (6)

$$\begin{aligned} \Delta^2 E_i(c', t) = & \sum_{j=1}^N A_{ij} [\text{sgn}(S_i(t-1)S_j(t))\delta(S_i(t-1), S_j(t)) \\ & - \text{sgn}(S_i(t)S_j(t))\delta(S_i(t), S_j(t))] \\ & - \gamma [\text{sgn}(S_i(t-1)) - \text{sgn}(S_i(t))]. \end{aligned}$$

For the MC energy function, from (7)

$$\begin{aligned} \Delta^2 E_i(c', t) = & \sum_{j=1}^N A_{ij} [\delta(S_i(t-1), S_j(t)) \\ & - \delta(S_i(t), S_j(t))] + \gamma [S_i(t-1) - S_i(t)]. \end{aligned}$$

Note that none of the $\Delta^2 E_i(c', t)$ depend on c' . Thus, while the values $\Delta E_i(c', t)$ may be different for different colors c' , a single value is used to update them from their previous values. Using this idea, the complexity of computing the vector $\Delta\vec{E}_i(t)$ is reduced from $\Theta(Nk)$ using (1) to $O(N+k)$ using (16).

Let us now consider case 2), the update of the ΔE 's of all node i 's neighbors. As in the previous case, we compute the variation in the delta energies for all possible colors c' in each node $j \in N(i)$.

For SSC, from (5)

$$\begin{aligned} \Delta^2 E_j(c', t) = & \delta(c', S_i(t)) - \delta(S_j(t), S_i(t)) \\ & - \delta(c', S_i(t-1)) + \delta(S_j(t), S_i(t-1)) \end{aligned}$$

for ISC, from (6)

$$\begin{aligned} \Delta^2 E_j(c', t) = & \text{sgn}(c' S_i(t))\delta(c', S_i(t)) - \text{sgn}(S_j(t)S_i(t)) \\ & \times \delta(S_j(t), S_i(t)) - \text{sgn}(c' S_i(t-1)) \\ & \times \delta(c', S_i(t-1)) + \text{sgn}(S_j(t)S_i(t-1)) \\ & \times \delta(S_j(t), S_i(t-1)) \end{aligned}$$

and for MC, from (7)

$$\begin{aligned} \Delta^2 E_j(c', t) = & \delta(c', S_i(t)) - \delta(S_j(t), S_i(t)) \\ & - \delta(c', S_i(t-1)) + \delta(S_j(t), S_i(t-1)). \end{aligned}$$

Therefore, the computational complexity of computing the vectors $\Delta\vec{E}_j(t)$'s of all neighbors j of node i is reduced from $\Theta(N^2k)$ in the worst case using (1) to $O(Nk)$ using (16).

```

annealing_loop( $\vec{S}, k, \gamma_L, \gamma_H$ )
{  $\vec{S} \leftarrow \text{inner\_loop}(\vec{S}, k, \gamma_L)$ ;           // first find a proper coloring
  do
  {  $\vec{S}_{prev} \leftarrow \vec{S}$ ;                       // save previous (best) coloring
     $\vec{S} \leftarrow \text{inner\_loop}(\vec{S}, k, \gamma_H)$ ;   // allow up-hill moves
     $\vec{S} \leftarrow \text{inner\_loop}(\vec{S}, k, \gamma_L)$ ;   // then return to a proper coloring
  } while( $E(\vec{S}) < E(\vec{S}_{prev})$ );             // keep repeating while solution improves
  return( $\vec{S}_{prev}$ );
}

```

Fig. 5. The annealing loop performed in the ISC and MC programs. In the SSC program, the annealing loop reduces to calling the inner loop function.

```

find_gammaH( $k$ )
{  $\vec{S} \leftarrow \text{random\_S}(k)$ ;
   $\gamma \leftarrow \gamma_L$ ;
   $\vec{S}_{baseline} \leftarrow \text{inner\_loop}(\vec{S}, k, \gamma)$ ; // first find a proper coloring
  do
  {  $\gamma \leftarrow \gamma + \delta_\gamma$ ;                 // increment  $\gamma$ 
     $\vec{S} \leftarrow \text{inner\_loop}(\vec{S}, k, \gamma)$ ;     // reach a local minimum
  } until( $\text{gammaH\_found}(\vec{S}, \vec{S}_{baseline})$ ); // repeat until  $\gamma$  is large enough
  return( $\gamma$ );
}

```

Fig. 6. This function is used to set the value of γ_H .

```

outer_loop( $k$ )
{  $\vec{S}_{best} \leftarrow \phi$ ;                          // initialize best coloring
   $\gamma_H \leftarrow \text{find\_gammaH}(k)$ ;             // set value of  $\gamma_H$  for annealing loop
  for( $r \leftarrow 1$ ;  $r \leq \text{RESTARTS}$ ;  $r \leftarrow r + 1$ )
  {  $\vec{S} \leftarrow \text{random\_S}(k)$ ;                  // randomly color the given graph with  $k$  colors
     $\vec{S} \leftarrow \text{annealing\_loop}(\vec{S}, k, \gamma_L, \gamma_H)$ ; // energy descent with annealing
    if( $E(\vec{S}) < E(\vec{S}_{best})$ )                    //  $\vec{S}$  is the minimum just found
       $\vec{S}_{best} \leftarrow \vec{S}$ ;                  // save the best solution
  }
  return( $\vec{S}_{best}$ );
}

```

Fig. 7. The multiple-restarts outer loop.

2) *The Annealing Loop*: In the ISC and MC cases, the multiplier γ in (3) and (4) may be used to play the role of a temperature. Large values of γ permit uphill moves in the landscape of the energy function of small γ . In the ISC case, when γ is large more vertices may be colored at the expense of some edges becoming improperly colored. In the MC case, when γ is large a vertex colored with a high-valued color may be recolored with a lower-valued color (possibly reducing the total number of colors used) even at the expense of some edges becoming improperly colored. These observations suggest that we include in our algorithm an annealing loop. After experimentation we decided upon the algorithm in Fig. 5.

Two values of γ, γ_L and γ_H , are passed from the outside, with γ_L in the feasibility region ($\gamma_L < \gamma^*$) and γ_H in the infeasibility region. The annealing loop first finds a proper coloring (using γ_L) and then follows a two-step schedule in which γ_H and γ_L are used alternatively while the solution quality keeps improving. When the quality of the solution found is not better than in the previous iteration, the loop ends and returns the best solution found.

In this procedure, γ_L and γ_H play a fundamental role. The value of γ_L must be less than γ^* to obtain a proper coloring, but

experimentally the energy descent evolution is almost insensitive to its specific value. Therefore, the value of γ_L can be computed directly. For example, in the ISC case we can set $\gamma_L = 0.9$ and in the MC case we can set $\gamma_L = 1/(2\Delta(G))$.

On the other hand, the annealing procedure is strongly sensitive to the value of γ_H . When γ_H is small, the uphill moves permitted are limited, and the probability of reaching a different, lower proper minimum in the subsequent energy descent is small. When γ_H is large, too many uphill moves are permitted, and too much information related to the previous proper coloring is lost. Our heuristic is based on the idea of improving the current solution by permitting a large number of uphill moves while still preserving much of the current feasible state.

To set the value of γ_H we introduce the function `find_gammaH()`, whose pseudocode is shown in Fig. 6. This function is invoked in the outer loop (Fig. 7) before the first call to the annealing loop.

In the ISC case, for a certain value of k , the first proper coloring will color a certain fraction of the nodes. To implement the heuristic, we simply keep increasing the value of γ (in increments of value δ_γ) until all nodes are colored. The first value of γ for which this happens will be used as γ_H . The

TABLE I

SSC AND ISC PROGRAMS ON SOME OF THE DIMACS COLORING BENCHMARK GRAPHS. IN SSC, v IS THE NUMBER OF VIOLATIONS AND IN ISC n IS THE NUMBER OF COLORED NODES. THE VALUE OF k IS SET EQUAL TO χ OR TO ITS BEST KNOWN APPROXIMATION (IN PARENTHESES). TIMES ARE IN SECONDS ($t[s]$)

name	Graph		k	SSC			ISC		
	$N = V $	$M = E $		v	$v/M\%$	$t [s]$	n	$n/N\%$	$t [s]$
DSJC1000.1	1000	99258	(21)	441.8	0.45	19.7	909.0	90.9	20.6
DSJC1000.5	1000	499652	(84)	738.4	0.15	203.6	858.0	85.8	248.9
DSJC1000.9	1000	898898	(226)	603.8	0.07	735.0	868.2	86.8	1123.2
DSJC250.1	250	6436	(8)	91.8	1.43	0.3	227.5	91.0	0.6
DSJC250.5	250	31336	(28)	140.4	0.45	1.1	218.9	87.6	3.2
DSJC250.9	250	55794	(72)	101.6	0.18	2.8	221.9	88.8	8.0
DSJC500.1	500	24916	(12)	241.4	0.97	2.0	445.2	89.0	2.8
DSJC500.5	500	125248	(48)	320.6	0.26	12.7	433.3	86.7	24.5
DSJC500.9	500	224874	(126)	260.4	0.12	36.7	429.9	86.0	72.1
DSJR500.1	500	7110	12	21.8	0.31	1.1	492.6	98.5	0.6
DSJR500.5	500	117724	(123)	114.4	0.10	12.8	456.6	91.3	18.3
flat1000_50.0	1000	245000	50	2967.0	1.21	157.6	571.6	57.2	256.6
flat1000_76.0	1000	246708	76	999.0	0.40	180.2	815.2	81.5	280.9
flat300_20.0	300	21375	20	613.2	2.87	1.6	222.5	74.2	6.8
flat300_28.0	300	21695	28	250.4	1.15	1.8	246.7	82.2	5.5
fpsol2.i.1	496	11654	65	19.4	0.17	2.2	477.5	96.3	2.9
fpsol2.i.2	451	8691	30	45.6	0.52	1.4	432.7	95.9	1.8
inithx.i.1	864	18707	54	44.2	0.24	7.7	829.8	96.0	6.3
inithx.i.2	645	13979	31	75.2	0.54	3.5	614.6	95.3	3.1
le450_15a	450	8168	15	128.2	1.57	1.2	419.2	93.2	1.5
le450_15c	450	16680	15	567.8	3.40	1.9	351.1	78.0	4.9
le450_25a	450	8260	25	23.4	0.28	1.3	441.9	98.2	1.2
le450_25c	450	17343	25	188.0	1.08	2.1	409.6	91.0	3.9
le450_5a	450	5714	5	584.0	10.22	1.1	432.4	96.1	2.0
le450_5c	450	9803	5	80.8	0.82	1.9	438.2	97.4	1.4
miles1500	128	10396	73	4.4	0.04	0.3	126.6	98.9	0.7
miles250	128	774	8	1.8	0.23	0.1	127.7	99.8	0.1
miles750	128	4226	31	5.6	0.13	0.1	125.9	98.4	0.3
mulsol.i.1	197	3925	49	3.6	0.09	0.3	194.5	98.7	0.4
myciel5	47	236	6	0.0	0.00	0.1	47.0	100.0	0.1
myciel6	95	755	7	0.4	0.05	0.1	95.0	100.0	0.1
myciel7	191	2360	8	0.8	0.03	0.2	190.8	99.9	0.2
queen13_13	169	6656	13	88.4	1.33	0.2	148.3	87.8	0.6
queen14_14	196	8372	(18)	16.6	0.20	0.3	192.2	98.1	0.5
queen15_15	225	10360	(17)	54.4	0.53	0.4	211.1	93.8	0.7
queen16_16	256	12640	(18)	63.2	0.50	0.5	240.4	93.9	1.0
school1	385	19095	14	119.8	0.63	2.2	384.1	99.8	3.5
zeroin.i.1	211	4100	49	9.6	0.23	0.3	202.8	96.1	0.6

function `gammaH_found()` in the ISC case will return a TRUE value only if all nodes are colored. The increment δ_γ in the ISC case will be set to one. This is justified by the fact that the number of violations that are permitted for coloring a previously uncolored vertex is equal to the value of γ_{ISC} (6). Therefore increments of less than a unit do not increase the possibility of uphill moves.

In the MC case, after the first proper coloring, we increase the value of γ until the number of distinct colors used decreases below a certain fraction f of the colors used in the first coloring ($f = 3/4$, for instance). In the MC case, therefore, the function `gammaH_found()` requires both the current state \vec{S} and the initial feasible state $\vec{S}_{baseline}$. The increment δ_γ in the MC case is set to the value γ_{MC}^* . From (7) and the proof of Lemma 2, we see that when increasing γ of an amount $\delta_\gamma = \gamma_{MC}^*$ the number of violations allowed for recoloring vertices with lower color values will increase by at least one.

3) *The Outer Loop*: Fig. 7 shows the pseudocode for the outer loop. At each restart, based on the value of k , the nodes' states are randomly initialized and then the annealing loop function is called. If the solution found is better than the previous

best, it then becomes the new current best solution. The algorithm eventually outputs the best solution found.

VI. EXPERIMENTAL RESULTS

Our programs (SSC, ISC, and MC) were written in C, compiled with the GNU gcc compiler (-O3 optimization) and run on a 466 MHz Alpha. The programs were tested on the official benchmarks from the 1993 DIMACS graph coloring challenge [18], [25]. These graphs include several random graphs (DSJCx.y), geometric random graphs (DSJRx.y and Rx.y[c]), "quasirandom" graphs (flatx.y_0), graphs from real-life applications (fpsol2.i.x, inithx.i.x, mulsol.i.1, and zeroin.i.1 map register allocation problems, school1 and school1_nsh are class scheduling graphs), graphs mapping combinatorial problems (queenx.y), and artificial graphs (mycielx, lex.y).

For the selection function F , with multiple random restarts a greedy, steepest-descent selection yields the best results in the case of SSC. In the cases of ISC and MC, we use different selection functions F in different steps of the annealing. Even though single results may exhibit variations, the best combination appears to be the use of a greedy selection when the goal is to reach

TABLE II
RESULTS FROM SSC- AND ISC-BASED ALGORITHMS (SSC-B AND ISC-B, RESPECTIVELY) FOR MINIMUM COLORING, AND RESULTS FROM MC. TIMES ARE IN SECONDS ($t[s]$) OR, WHEN LONGER THAN ONE HOUR, IN *Hour: Minute* FORMAT

Graph				SSC-B		ISC-B		MC	
name	$ V $	$ E $	χ	k	t [s]	k	t [s]	k	t [s]
DSJC1000.1	1000	99258	(21)	34.4	74.1	34.4	11.6	26.8	6.9
DSJC1000.5	1000	499652	(84)	142.7	887.7	142.7	125.3	111.5	91.7
DSJC1000.9	1000	898898	(226)	351.7	1:21	353.3	1102.4	284.2	394.4
DSJC250.1	250	6436	(8)	13.0	0.7	12.4	0.5	10.5	0.3
DSJC250.5	250	31336	(28)	45.0	5.3	43.9	3.2	36.5	1.9
DSJC250.9	250	55794	(72)	103.2	13.9	102.0	13.5	86.4	5.5
DSJC500.1	500	24916	(12)	20.4	5.6	20.4	2.1	16.0	1.2
DSJC500.5	500	125248	(48)	79.8	63.7	79.5	18.9	63.6	12.2
DSJC500.9	500	224874	(126)	189.1	222.1	192.0	101.6	156.4	41.8
DSJR500.1	500	7110	12	14.1	2.6	14.2	0.8	13.0	0.6
DSJR500.5	500	117724	(123)	161.8	84.9	162.3	38.0	133.5	17.6
flat1000_50.0	1000	245000	50	139.7	938.5	140.7	130.3	108.4	85.2
flat1000_76.0	1000	246708	76	141.0	906.5	140.1	118.6	109.7	100.9
flat300_20.0	300	21375	20	49.6	9.1	49.2	4.6	38.7	2.9
flat300_28.0	300	21695	28	50.5	9.2	49.4	4.9	40.3	2.7
fpsol2.i.1	496	11654	65	96.2	10.3	108.8	5.7	65.0	2.2
fpsol2.i.2	451	8691	30	90.7	5.9	113.1	5.2	30.0	1.4
inithx.i.1	864	18707	54	107.8	54.1	128.0	12.1	54.0	4.2
inithx.i.2	645	13979	31	97.4	23.8	120.4	10.6	31.0	2.6
le450_15a	450	8168	15	22.8	4.9	23.1	2.5	18.0	1.1
le450_15c	450	16680	15	33.1	11.2	33.6	6.0	25.9	2.1
le450_25a	450	8260	25	29.6	4.8	30.3	2.6	25.9	1.2
le450_25c	450	17343	25	39.4	10.0	39.7	5.6	31.0	2.4
le450_5a	450	5714	5	13.4	2.8	13.1	1.1	9.0	0.7
le450_5c	450	9803	5	17.3	4.2	17.4	1.7	5.8	0.7
miles1500	128	10396	73	75.9	1.0	75.9	1.4	73.0	0.9
miles250	128	774	8	8.8	0.1	8.5	0.1	8.0	0.1
miles750	128	4226	31	34.4	0.2	34.2	0.3	31.8	0.3
mulsol.i.1	197	3925	49	52.2	0.9	52.2	1.1	49.0	0.5
myciel5	47	236	6	6.0	0.1	6.0	0.1	6.0	0.1
myciel6	95	755	7	7.0	0.1	7.0	0.1	7.0	0.1
myciel7	191	2360	8	8.2	0.5	8.8	0.4	8.0	0.2
queen13_13	169	6656	13	19.4	0.5	19.1	0.7	16.9	0.4
queen14_14	196	8372	(18)	20.9	0.8	20.3	0.9	18.1	0.5
queen15_15	225	10360	(17)	22.2	1.0	21.8	1.1	19.4	0.6
queen16_16	256	12640	(18)	23.7	1.3	23.6	1.3	21.0	0.7
school1	385	19095	14	45.1	8.9	47.2	4.8	28.3	2.6
zeroin.i.1	211	4100	49	60.4	0.9	61.5	1.0	49.0	0.5

a feasible minimum ($\gamma = \gamma_L$), and a random selection when the goal is to move uphill and escape the local minimum ($\gamma = \gamma_H$). A randomized-greedy selection function works about as well in either step, but is slightly slower in execution.

1) *Spanning and Induced Subgraph k Coloring*: Table I reports averaged results over multiple runs of the SSC and the ISC programs. In both programs, the number of colors allowed is indicated in the k column. The value of k was set to χ when known. We did this to make the problems hard yet fully solvable, to permit evaluation of results. When χ was not known, k was set to the cardinality of the smallest known coloring we were able to find in the literature (values in parentheses). The total number of restarts was set to $N/10$ for the SSC program, which does not perform annealing. For the ISC program the total number of restarts was set to 20 for all graphs.

The results exhibit a large variation in solution quality. This is expected because the test graphs are significantly different from each other and they were created with “difficult to color” characteristics for the DIMACS challenge. However, in most instances, SSC colors more than 98% of the edges properly and ISC colors more than 85% of the nodes properly.

2) *Minimum Coloring*: In Table II we compare our three algorithms on the minimum coloring problem on the same graphs as in Table I. A minimum coloring can be found using SSC and ISC algorithms by adapting them to perform a binary search starting at $k = \Delta(G) + 1$. The names SSC-B and ISC-B indicate the binary search versions of SSC and ISC, respectively. The maximum number of restarts allowed at each stage of the search was $N/10$ for SSC-B and 20 for ISC-B, while the MC program was run with 10 restarts. The MC algorithm consistently performs better than SSC-B and ISC-B, even if allowed a smaller number of restarts.

3) *Comparison With Other Methods*: Finally, Table III compares MC with other algorithms reported in the literature (all timings are from the respective papers).

The recursive binary adaptation (RBA) algorithm proposed by Jagota [22] is the only other neural algorithm for which we have results on DIMACS benchmarks. All the other neural approaches to the minimum coloring mentioned in Section II either do not offer experimental results or their test graphs are not available. Timings for RBA are on a SUN Sparc 10.

“Coud” is the sequential coloring algorithm as modified by Coudert [4], probably the most efficient approach to date to

TABLE III
A COMPARISON OF MC WITH OTHER NEURAL (RBA) AND NON-NEURAL APPROACHES TO GRAPH COLORING. TIMES ARE IN SECONDS ($t[s]$) OR, WHEN LONGER THAN ONE HOUR, IN *HOURL: MINUTE* FORMAT

Graph		MC		RBA		Coud		lmXRLF	SWO		DSATUR	
name	χ	k	t [s]	k	t [s]	k	t [s]	k	k	t [s]	k	t [s]
DSJC125.5	(17)	21.3	0.4	26	42.0	-	-	18	18.3	1.6	22.9	0.03
DSJC250.5	(28)	36.5	1.9	43	478.0	-	-	30	31.9	8.3	37.3	0.12
DSJC500.5	(48)	63.6	12.2	72	1:47	-	-	50	56.3	40.9	65.7	0.46
DSJC1000.5	(84)	111.5	91.7	127	26:33	-	-	85	101.5	208.6	116.3	1.86
DSJR500.1	12	13.0	0.6	15	907.0	12	0.12	13	12.0	2.0	12.9	0.04
DSJR500.5	(123)	133.5	17.6	143	4:42	-	-	128	124.1	68.7	129.0	0.44
R125.1	5	5.0	0.1	5	6.0	5	0.02	-	5.0	0.2	5.0	0.00
R125.1c	46	46.2	1.0	51	96.0	46	0.13	-	46.0	5.1	46.4	0.05
R125.5	36	38.8	0.5	44	81.0	36	2.60	-	36.0	2.8	38.2	0.03
R250.1	8	8.0	0.2	9	103.0	8	0.05	-	8.0	0.5	8.0	0.01
R250.1c	64	65.8	5.2	76	1133.0	64	2.13	-	64.0	30.6	66.4	0.22
R250.5	65	70.4	2.5	79	1046.0	-	-	-	65.0	14.7	69.1	0.11
R1000.1	20	22.3	2.6	26	3:18	20	0.54	-	20.0	8.0	20.3	0.12
R1000.5	(238)	260.4	156.4	275	56:54	-	-	-	238.9	574.5	249.1	1.79
flat300_20_0	20	38.7	2.9	20	491.0	-	-	20	25.3	16.4	42.1	0.16
flat300_26_0	26	40.4	2.9	26	640.0	-	-	28	35.8	12.0	42.1	0.16
flat300_28_0	28	40.3	2.7	28	690.0	-	-	32	35.7	11.9	41.8	0.16
flat1000_50_0	50	108.4	85.2	50	8:31	-	-	50	100.0	203.9	115.2	1.82
flat1000_60_0	60	109.2	93.9	60	10:8	-	-	61	100.7	198.0	115.1	1.83
flat1000_76_0	76	109.7	100.9	76	13:17	-	-	85	100.6	208.4	114.5	1.83
le450_15a	15	18.0	1.1	22	1162.0	-	-	17	15.0	5.5	17.1	0.07
le450_15b	15	18.0	1.1	22	1168.0	-	-	17	15.0	6.1	16.9	0.07
le450_15c	15	25.9	2.1	30	1694.0	-	-	21	21.1	8.0	24.7	0.14
le450_15d	15	25.9	2.1	31	1766.0	-	-	21	21.2	7.8	24.6	0.14
musol.i.1	49	49.0	0.5	49	379.0	49	0.10	49	49.0	5.9	49.0	0.03
myciel5	6	6.0	0.1	-	-	6	4.17	-	-	-	6.0	0.00
myciel6	7	7.0	0.1	-	-	7	35:22	-	-	-	7.0	0.01
myciel7	8	8.0	0.2	-	-	-	-	8	-	-	8.0	0.02
queen8_8	9	10.6	0.1	-	-	9	38.69	-	-	-	11.6	0.01
queen9_9	10	11.9	0.1	-	-	10	5:12	-	-	-	12.8	0.02
queen15_15	(17)	19.4	0.6	-	-	-	-	17	-	-	20.7	0.08
queen16_16	(18)	21.0	0.7	-	-	-	-	18	-	-	22.1	0.09
school1	14	28.3	2.6	42	2178.0	14	0.41	14	14.0	8.4	18.1	0.15
school1_nsh	14	23.5	2.0	39	1640.0	14	0.25	14	14.0	7.2	21.3	0.11

graph coloring. It must be noted, however, that this algorithm works exceptionally well only on *1-perfect* graphs, i.e., in which χ is equal to the size of the maximum clique. On other graphs (like the Mycielski transformation based graphs) it is as slow as a sequential “branch and bound” coloring. “Coud,” however, is an exact algorithm, always yielding an optimal coloring. Timings for “Coud” are on a 60 MHz SUN SuperSparc.

“lmXRLF” is the *least constraining-most constrained extended RLF* algorithm by Kirovski and Potkonjak [28] that is based on the recursive largest first (RLF) heuristic by Leighton [30] to improve the simple sequential coloring. In this case we report only the coloring found, since they do not provide detailed timings but only state that none of the runs took longer than 1.5 hours and that all real-life examples were solved in less than a second on a SUN Sparc 5.

“SWO” is the “*Squeaky Wheel*” Optimization algorithm by Joslin and Clements [26], a general approach to optimization based on the iterative application of a greedy algorithm and a solution analyzer that changes the priority in which the elements are considered by the greedy algorithm in the next iteration. Times for “SWO” are from a 333 MHz Pentium Pro.

Finally, “DSATUR” is the well-known greedy routine by Brélaz [2]. We used the code provided by Joseph Culberson [5], compiled with the GNU gcc compiler and run on the same 466 MHz Alpha that we used to run our MC program on.

The nonneural algorithms specifically developed to solve the graph coloring problem tend to perform better than the neural ones. However, we see that no algorithm wins consistently, and that our implementation can still perform better than the other ones in some cases.

VII. CONCLUSION

We described a generalized quasi-Hopfield network approach to optimization that provides an algorithmic framework in which the problem-specific issues are embedded only in the energy function that the network evolution tries to minimize. We applied this approach to the solution of a family of NP-hard graph coloring problems, specifically the minimum coloring problem, the spanning subgraph k -coloring problem and the induced subgraph k -coloring problem.

The use of multivalued neurons allowed a network representation that is more natural and compact than in previous similar approaches. This reduced the network size from $(Nk)^2$ to N^2 connections asymptotically and eliminated the possibility of conflicts among neurons in the same cluster.

Experimental results showed that the performance of such a method compares favorably with other neural and nonneural approaches to the graph coloring problem. While acknowledging that the nonneural algorithms performed better on average on

this specific problem, our approach retains its characteristics of generality that might make it successfully applicable to other optimization problems. Finally, due to its simple, clean structure, this approach is also intrinsically easy to parallelize.

REFERENCES

- [1] M. O. Berger, "k-coloring vertices using a neural network with convergence to valid solutions," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 7, 1994, pp. 4514–4517.
- [2] D. Bréaz, "New methods to color the vertices of a graph," *Commun. ACM*, vol. 22, pp. 251–256, Apr. 1979.
- [3] M. Chams, A. Hertz, and D. de Werra, "Some experiments with simulated annealing for coloring graphs," *Europ. J. Operational Res.*, vol. 32, pp. 260–266, 1987.
- [4] O. Coudert, "Exact coloring of real-life graphs is easy," in *Proc. 34th Design Automat. Conf.*, New York, 1997, pp. 121–126.
- [5] Joseph Culberson's Coloring Page, J. Culberson. [Online]. Available: <http://web.cs.ualberta.ca/~joe/Coloring>
- [6] J. C. Culberson and F. Luo, "Exploring the k-colorable landscape with iterated greedy," in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. Providence, RI: Amer. Math. Soc., 1996, vol. 26, pp. 245–284.
- [7] E. D. Dahl, "Neural network algorithm for an NP-complete problem: Map and graph coloring," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 3, 1987, pp. 113–120.
- [8] G. De Micheli, *Synthesis and Optimization of Digital Circuits*. New York: McGraw Hill, 1994.
- [9] D. de Werra, "An introduction to timetabling," *Europ. J. Operational Res.*, vol. 19, pp. 151–162, 1985.
- [10] A. Di Blas, A. Jagota, and R. Hughey, "Parallel implementations of optimizing neural networks," in *Proc. ANNIE 2000 Conf.*, Nov. 2000, pp. 153–158.
- [11] P. J. Donnelly and D. J. A. Welsh, *The Antivoter Problem: Random 2-Colorings of Graphs*. New York: Academic, 1984, pp. 133–144. Graph Theory and Combinatorics.
- [12] C. Fleurent and J. A. Ferland, "Object-oriented implementation of heuristic search methods for graph coloring, maximum clique and satisfiability," in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. Providence, RI: Amer. Math. Soc., 1996, vol. 26, pp. 619–652.
- [13] D. Gajski, N. Dutt, A. Wu, and S. Lin, *High-Level Synthesis: Introduction to Chip and System Design*. Boston, MA: Kluwer, 1992.
- [14] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY: Freeman, 1979.
- [15] D. W. Gassen and J. D. Carothers, "Graph color minimization using neural networks," in *Proc. IEEE Int. Joint Conf. on Neural Networks*, 1993, pp. 1541–1544.
- [16] F. Glover, M. Parker, and J. Ryan, "Coloring by tabu branch and bound," in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. Providence, RI: Amer. Math. Soc., 1996, vol. 26, pp. 285–308.
- [17] A. Hertz and D. de Werra, "Using tabu search techniques for graph coloring," *Computing*, vol. 39, pp. 345–351, 1987.
- [18] Center for Discrete Mathematics and Theoretical Computer Science. [Online]. Available: <http://dimacs.rutgers.edu>
- [19] J. Hopfield and D. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.
- [20] —, "Computing with neural circuits: A model," *Science*, vol. 233, pp. 625–633, 1986.
- [21] A. Jagota, "Approximating maximum clique with a Hopfield network," *IEEE Trans. Neural Networks*, vol. 6, pp. 724–735, May 1995.
- [22] —, "An adaptive, multiple restarts neural network algorithm for graph coloring," *Europ. J. Operational Res.*, vol. 93, pp. 257–270, 1996.
- [23] A. Jagota, L. Sanchis, and R. Ganesan, "Approximately solving maximum clique using neural network and related heuristics," in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science: American Mathematical Society*, 1996, vol. 26, pp. 169–204.
- [24] D. S. Johnson, C. R. Aragon, L. A. Mcgeoch, and C. Schevon, "Optimization by simulated annealing: An experimental evaluation; Part II, graph coloring and number partitioning," *Operations Res.*, vol. 39, pp. 378–406, May–June 1991.
- [25] D. S. Johnson and M. A. Trick, Eds., *Clique, Coloring, and Satisfiability—Second DIMACS Implementation Challenge*. Providence, RI: Amer. Math. Soc., 1996.
- [26] D. E. Joslin and D. P. Clements, "Squeaky wheel' optimization," *J. Artificial Intell. Res.*, vol. 10, pp. 353–373, 1999.
- [27] D. Karger, R. Motwani, and M. Sudan, "Approximate graph coloring by semidefinite programming," *Proc. 35th IEEE Symp. Foundations of Computer Sci.*, pp. 2–43, 1994.

- [28] D. Kirowski and M. Potkonjak, "Efficient coloring of a large spectrum of graphs," in *Proc. 35th Design Automat. Conf.*, New York, 1998, pp. 427–432.
- [29] J. H. M. Korst and E. H. L. Aarts, "Combinatorial optimization on a Boltzmann machine," *J. Parallel Distributed Comput.*, vol. 6, pp. 331–357, 1989.
- [30] F. T. Leighton, "A graph coloring algorithm for large scheduling problems," *J. Res. Nat. Bureau Standards*, vol. 84, no. 6, pp. 489–503, 1979.
- [31] G. Lewandowski and A. Condon, "Experiments with parallel graph coloring heuristics and applications of graph coloring," in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. Providence, RI: Amer. Math. Soc., 1996, vol. 26, pp. 309–334.
- [32] M. Misra, "Parallel environments for implementing neural networks," *Neural Comput. Surveys*, vol. 1, pp. 46–80, 1997.
- [33] C. Morgenstern, "Distributed coloration neighborhood search," in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. Providence, RI: Amer. Math. Soc., 1996, vol. 26, pp. 335–358.
- [34] C. Peterson and B. Soderberg, "A new method for mapping optimization problems onto neural networks," *Int. J. Neural Syst.*, vol. 1, no. 1, pp. 3–22, 1989.
- [35] A. D. Petford and D. J. A. Welsh, "A randomised 3-coloring algorithm," *Discrete Math.*, vol. 74, pp. 253–261, 1989.
- [36] W. J. M. Philipsen and L. Stok, "Graph coloring using neural networks," in *IEEE Int. Symp. Circuits Syst.*, vol. 3, 1991, pp. 1597–1600.
- [37] S. Ramanathan and E. L. Lloyd, "Scheduling broadcasts in multihop radio networks," *IEEE/ACM Trans. Networking*, vol. 1, pp. 166–172, 1993.
- [38] E. C. Sewell, "An improved algorithm for exact graph coloring," in *DIMACS Series Discrete Math. Theoretical Comput. Sci.*: Amer. Math. Soc., 1996, vol. 26, pp. 359–373.
- [39] K. Smith and M. Palaniswami, "Static and dynamic channel assignment using neural networks," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 238–249, Feb. 1997.
- [40] Y. Takefuji and K. C. Lee, "Artificial neural networks for four-coloring map problems and k-colorability problems," *IEEE Trans. Circuits Syst. I*, vol. 38, pp. 326–333, Mar. 1991.
- [41] C. W. Wu, "Graph coloring via synchronization of coupled oscillators," *IEEE Trans. Circuits Syst. I*, vol. 45, pp. 974–978, Sept. 1998.



Andrea Di Blas (M'99) received the M.S. and Ph.D. degrees in electronic engineering from Politecnico di Torino, Italy, in 1994 and 2000.

He has been a Researcher at University of California, Santa Cruz, since 1999, where he is currently also a Lecturer in the Department of Computer Engineering. His research interests include methodologies for parallel processing, neural networks, optimization problems, and bioinformatics.



Arun Jagota received the Ph.D. degree in computer science from The State University of New York, Buffalo.

He has been a Visiting, Adjunct, or affiliated faculty member at the University of Memphis, the University of North Texas, the University of California, Santa Cruz, Santa Clara University, and San Jose State University. He has taught more than 35 different courses in computer science, and some in mathematics. He has authored more than 20 research papers in journals, and about the same

number in conference records. He conducts research in neural networks, in machine learning, and in bioinformatics.

Dr. Jagota is the Founding Editor of the journal *Neural Computing Surveys*.



Richard Hughey (S'83–M'85) received the B.A. degree in mathematics and the B.S. degree in engineering from Swarthmore College, Swarthmore, PA, and the Sc.M. and Ph.D. in computer science from Brown University, Providence, RI.

He is Associate Professor and Chair of the Department of Computer Engineering at the University of California, Santa Cruz. His research interests include parallel processing and bioinformatics.

Dr. Hughey is a member of the American Society for Engineering Education (ASEE), the International Society for Computational Biology (ISCB), and the IEEE Computer Society.