Relaxation Labeling Networks for the Maximum Clique Problem

Marcello Pelillo
Università "Ca' Foscari" di Venezia

Relaxation labeling networks are a class of parallel distributed computational models extremely popular in computer vision and pattern recognition. Despite their original heuristic derivation, they possess interesting dynamical properties and learning abilities, and exhibit also a certain biological plausibility. In this paper, it is shown how to take advantage of the properties of these models to approximately solve the maximum clique problem, a well-known intractable optimization problem which has practical applications in various fields. The approach is based on a result by Motzkin and Straus which naturally leads to formulate the problem in a manner that is readily mapped onto a relaxation labeling network. Extensive simulations have demonstrated the validity of the proposed model, both in terms of quality of solutions and speed.

Maximum clique problem, relaxation labeling processes, neural networks, optimization.

1 INTRODUCTION

Relaxation models of computation have recently attracted considerable interest within the neural network community not only because they appear to be closely related to the style of computation the human brain is supposed to implement [1], but also for their intrinsic abilities to solve difficult optimization problems [2, 3]. In the

[0]The author would like to thank E. R. Hancock for useful suggestions that helped to improve the presentation of the paper, A. Jagota for stimulating discussions about the Motzkin-Straus program, and S. Ricci for carrying out part of the simulations presented in the paper.

Correspondence and requests for reprints should be sent to Marcello Pelillo, Dipartimento di Matematica Applicata e Informatica, Università "Ca' Foscari" di Venezia, Via Torino 155, 30173 Venezia Mestre, Italy; e-mail: pelillo@di.unive.it.

field of computational vision, too, relaxation techniques have long been recognized as a viable way of solving various practical problems such as, for example, optical flow computation, edge detection, and stereo vision (see, e.g., [4]). One of the earliest and perhaps best-known examples of such cooperative algorithms was developed in the mid-1970s by Rosenfeld, Hummel, and Zucker [5] to solve certain labeling problems arising in the 3-D interpretation of ambiguous line drawings. Their *relaxation labeling* model (or variations thereof) has since then been the subject of extensive studies and has now become a widely accepted tool in the computer vision toolkit. Despite the fact that the algorithm was originally motivated by simple heuristic arguments, recent work has shown that it possesses interesting dynamical properties [6], learning capabilities [7], and turns out to be intriguingly related to certain mechanisms in the early stages of biological visual systems [8].

In this paper, it is shown how a simple yet powerful version of the original relaxation labeling model of Rosenfeld, Hummel, and Zucker [5] is capable of approximately solving the maximum clique problem (MCP), a well-known *NP*-hard optimization problem [9] which is computationally intractable even to approximate well [10]. The problem is frequently encountered in such diverse domains as cluster analysis, information retrieval, computer vision, and experimental design, to name just a few, and turns out to be intimately related to a number of other difficult graph problems. During the past few years, a wide variety of algorithms have been proposed to solve the MCP, and the literature on this subject has now grown considerably [11]. Despite its practical and theoretical relevance, however, we note that in the neural network domain relatively few attempts have been made to tackle the problem, in comparison to the huge amount of work done on, e.g., the traveling salesman problem (see [12, 13, 14, 15, 16] and references therein).

The approach described in this paper is based on a result due to Motzkin and Straus [17] which allows us to formulate the MCP as a continuous optimization problem with linear constraints, a problem that is then naturally mapped onto a relaxation labeling network. The Motzkin-Straus theorem has been exploited by Pardalos and Phillips [18] for developing a global optimization approach to solve the MCP and deriving bounds on the maximum clique size, and has also recently motivated the development of a powerful continuous based heuristic [19]. Like other clique finding neural network models (e.g., [14, 16]), the one proposed here requires a number of computational units that coincides with the number of vertices of the graph at hand, and no penalty term is needed. One additional feature that our algorithm shares with the inverted neural network of Grossman [16] is that the model employs binary 0/1 connections and is therefore especially suited for parallel hardware implementation. In contrast to Grossman's algorithm, however, ours performs a search in a continuous rather than discrete domain. We note that a number of authors have stressed the advantages of modeling discrete optimization problems via continuous methods [2, 19]. One distinctive feature of our relaxation network is that, unlike most clique finding neural models, it is completely devoid of operational parameters which typically require a lengthy, problem-dependent tuning phase, and this makes the model extremely simple to implement and use.

The outline of the paper is as follows. In Section 2, we introduce relaxation labeling networks and present some of their fundamental properties. Section 3 presents the Motzkin-Straus theorem, and discusses some theoretical results, most of which are formally derived in a companion paper [20]. In Section 4, the results of our experiments are given, and in Section 5 we draw our conclusions.

2 RELAXATION LABELING NETWORKS

The relaxation network we shall consider in this paper consists of n pairwise interconnected computational units. Let w_{ij} be the strength of the connection from unit u_i to unit u_j, and let W denote the $n \times n$ real-valued matrix having w_{ij} as its (i, j) entry. It will be assumed here that the weight matrix is symmetric[1] ($w_{ij} = w_{ji}$) and nonnegative ($w_{ij} \geq 0$). As to the latter point, we note that unipolar networks, like the one we are proposing here, turn out to be advantageous in many applications, especially when hardware implementation is a concern [21].

Let $u_i(t)$ represent the state of unit u_i at time t, and define the state of the network as a whole at time t to be the n-vector $\mathbf{u}(t) = (u_1(t), \cdots, u_n(t))^T$, where "$T$" denotes matrix transposition. The system operates as follows. It starts out with an initial state vector $\mathbf{u}(0)$ and iteratively and synchronously updates its own state according to the following discrete dynamical equation, which is but a special instance of the heuristic nonlinear relaxation scheme originally developed in [5]:

$$u_i(t+1) = u_i(t) q_i(t) \Bigg/ \sum_{j=1}^{n} u_j(t) q_j(t) \qquad i = 1 \ldots n \tag{1}$$

where

$$q_i(t) = \sum_{j=1}^{n} w_{ij} u_j(t) \qquad i = 1 \ldots n \tag{2}$$

is the net input to unit u_i at time t. The process evolves until $\mathbf{u}(t+1) = \mathbf{u}(t)$, which means that a fixed point has been reached.

Let $\mathcal{K}$ denote the standard simplex of $\Re^n$:

$$\mathcal{K} = \left\{ \mathbf{u} \in \Re^n \ : \ u_i \geq 0, \text{ all } i = 1 \ldots n, \text{ and } \sum_{i=1}^{n} u_i = 1 \right\}.$$

Because of the normalization factor present in equation (1) and the unipolarity condition, the network essentially performs a mapping of the domain $\mathcal{K}$ onto itself,

[1] See [6] for results concerning the dynamical properties of these models when the symmetry condition is relaxed.

provided that $\mathbf{u}(0) \in \mathcal{K}$. Levinson, Rabiner, and Sondhi [22], in a rather different context, offered a simple geometrical interpretation for transformations like (1). Let $\mathbf{u}$ be a point in $\mathcal{K}$, and let $\mathbf{q}$ denote the n-vector composed of the q_i's, as defined in (2). Moreover, let $\mathbf{z}$ be the n-vector whose ith component is given by the component-wise product between $\mathbf{u}$ and $\mathbf{q}$, i.e., $z_i = u_i q_i$. Then, it is readily seen that the vector obtained by applying the transformation (1) to $\mathbf{u}$ is simply the intersection of the vector $\mathbf{z}$ (or its extension) with the hyperplane defined by $\sum_i u_i - 1 = 0$.

As an aside, we note that output normalization has now become a common practice within the neural network community [23]; this can be regarded as a multi-input generalization of the more familiar logistic nonlinearity, and can be easily implemented onto physical circuitry [24]. It can also be considered as a form of "soft" competition among hypotheses, an approach that contrasts with the classical winner-take-all view [25].

Let us now examine the dynamical properties of the proposed relaxation scheme. Central to our analysis is an interesting result known as the Baum-Eagon inequality [26], which we present here in a rather simplified form.

Theorem 1 (Baum-Eagon) *Let $P(\mathbf{u})$ be a polynomial in the variables $\{u_i\}$ with nonnegative coefficients, and let $\mathbf{u}$ be a point of the domain $\mathcal{K}$. Define the mapping $\mathbf{z} = \mathcal{M}(\mathbf{u})$ as*

$$z_i = u_i \frac{\partial P(\mathbf{u})}{\partial u_i} \Bigg/ \sum_{j=1}^{n} u_j \frac{\partial P(\mathbf{u})}{\partial u_j} \qquad i = 1 \ldots n \, . \tag{3}$$

Then $P(\mathcal{M}(\mathbf{u})) > P(\mathbf{u})$, unless $\mathcal{M}(\mathbf{u}) = \mathbf{u}$ or, in other words, $\mathcal{M}$ is a growth transformation *[27] for the polynomial P.*

Indeed, Baum and Eagon's theorem was originally proven for the special case of homogeneous polynomials. In a subsequent paper, however, Baum and Sell [27] extended the original result to nonhomogeneous polynomials, and proved that the inequality also holds for all points lying on the segment connecting $\mathbf{u}$ and $\mathcal{M}(\mathbf{u})$. Additionally, they provided an analysis of the asymptotic behavior of the transformation $\mathcal{M}$ in the vicinity of local extrema. As noted by Baum and Sell [27], the mapping defined in (3) makes use of first derivatives only and yet is able to make finite steps while increasing P. This contrasts sharply with conventional gradient methods, for which an increase in the objective function is guaranteed only when infinitesimal steps are taken, and determining the optimal step size entails computing higher-order derivatives.

The Baum-Eagon inequality provides us with an effective iterative means for optimizing polynomial functions over a domain of probability values and, in fact, it has served as the basis for many statistical estimation procedures. More recently, its usefulness in the field of speech recognition has been proven extensively [22].

Now, a direct application of the Baum-Eagon inequality reveals that the mapping defined by formulas (1) and (2) acts as a growth transformation for the following quadratic polynomial

$$P(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T W \mathbf{u} \ , \tag{4}$$

which means that

$$P(\mathbf{u}(t+1)) > P(\mathbf{u}(t)) \tag{5}$$

for all $t \geq 0$, unless $\mathbf{u}(t+1) = \mathbf{u}(t)$. This property follows immediately from the fact that, when $W = W^T$, we get $\partial P(\mathbf{u})/\partial u_i = \sum_{j=1}^{n} w_{ij} u_j$ $(i = 1 \ldots n)$, which have exactly the same form as the q_i's in (2). Put another way, we can say that the negative of P is a strict Liapunov (or energy) function for the dynamical system described by equations (1) and (2). For the sake of completeness, we mention that the same result can also be readily obtained from an earlier (and less general) theorem of Blakley [28]. It may also be of some interest to note that the same recurrence relations defined in (1) and (2) were earlier independently employed as a model of evolution in population genetics [29].

3 CLIQUE DETECTION VIA RELAXATION

Let $G = (V, E)$ be a undirected graph, where $V = \{1, \cdots, n\}$ is the set of vertices, and $E \subseteq V \times V$ is the set of edges. A *clique* of G is a subset of V in which every pair of vertices is connected by an edge. A clique is called *maximal* if no strict superset of C is a clique. A *maximum* clique is a clique having largest cardinality (note that a maximal clique is not necessarily a maximum one). Hence, the maximum clique problem consists of finding a clique of maximum size of a graph G. In the following, given a set S of vertices in G, we will denote by $\mathbf{u}^s$ its characteristic vector, defined as $u_i^s = 1/|S|$ if $i \in S$ and $u_i^s = 0$ otherwise, where $|S|$ denotes the cardinality of S.

Given a graph G, consider the following quadratic program:

$$\begin{array}{ll} \text{maximize} & f(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T A \mathbf{u} \\ \text{subject to} & \mathbf{u} \in \mathcal{K} \end{array} \tag{6}$$

where $A = (a_{ij})$ is the adjacency matrix of G (i.e., $a_{ij} = 1$ if $(i, j) \in E$, and $a_{ij} = 0$ if $(i, j) \notin E$). A point $\mathbf{u}^*$ is said to be a global maximum of (6) if $f(\mathbf{u}^*) \geq f(\mathbf{u})$ for all $\mathbf{u} \in \mathcal{K}$. It is said to be a local maximum if there exists an $\epsilon > 0$ such that $f(\mathbf{u}^*) \geq f(\mathbf{u})$ for all $\mathbf{u} \in \mathcal{K}$ whose distance from $\mathbf{u}^*$ is less than ϵ, and if $f(\mathbf{u}^*) = f(\mathbf{u})$ implies $\mathbf{u}^* = \mathbf{u}$ then $\mathbf{u}^*$ is said to be a strict local maximum.

Motzkin and Straus [17] proved the following remarkable result which relates the global maximum of (6) to the maximum clique size of G; this turns out to be the basis of the work reported in this paper (see also [18] for proof).

Theorem 2 (Motzkin-Straus) *Let $\alpha = f(\mathbf{u}^*)$ be the global maximum of (6). Then, G has a maximum clique C of size $k = 1/(1-2\alpha)$. The global maximum $\mathbf{u}^*$ of (6) can be attained by setting $u_i^* = 1/k$ if $i \in C$ and $u_i^* = 0$ otherwise.*

Hence, the Motzkin-Straus result provides us with a method to solve the "evaluation" version of the MCP, i.e., given a graph determine the *size* of its maximum clique (not the clique itself): this can be accomplished by maximizing in $\mathcal{K}$ the polynomial f, and then applying Theorem 2. We note that this problem formulation is essentially equivalent to its original "optimization" counterpart. Straightforward recursive or iterative procedures are in fact available to determine the vertices comprising the maximum clique once we have a means to compute its size [9, 18]. However, the following characterization can easily be proven [20].

Theorem 3 *A subset C of vertices in G is a maximum clique if and only if its characteristic vector $\mathbf{u}^c$ is a global maximum of the Motzkin-Straus problem.*

The computational significance of the previous result is that if the global maximum $\mathbf{u}^*$ of f over $\mathcal{K}$ is attained at a point $\mathbf{u}^*$ of the form $u_i^* = 1/k$ if $i \in C$ and $u_i^* = 0$ if $i \notin C$, for some subset C of vertices, then we can say that C is a maximum clique of G. In this case, we need not resort to such iterative procedures to derive the vertices of the maximum clique, since they are directly encoded into $\mathbf{u}^*$ itself. Note that, in general, global maxima of (6) do not necessarily correspond to maximum cliques. This was earlier observed by Pardalos and Phillips [18], and has recently been formalized by Pelillo and Jagota [20], who showed that problem (6) can have in fact infinitely many "infeasible" solutions.

In exactly the same way as maximum cliques are characterized in terms of *global* maxima of the Motzkin-Straus problem (6), we find that maximal cliques can be characterized in terms of *local* maxima of (6). Specifically, the following result has recently been proven by Gibbons *et al.* [30], and a weaker version thereof has independently been derived in [20].

Proposition 1 *Let C be a maximal clique of G. Then its characteristic vector $\mathbf{u}^c$ is a local maximum of the Motzkin-Straus quadratic program.*

The converse has also been proven to hold under certain circumstances [20, 30].

Proposition 2 *Let C be a subset of vertices of a graph G, and let $\mathbf{u}^c$ be its characteristic vector. If $\mathbf{u}^c$ is a strict local maximum of the Motzkin-Straus problem, then C is a maximal clique.*

In [20, 30] it is shown that under the hypothesis of the previous proposition, C is in fact a *strictly* maximal clique, which means that no vertex external to C is connected with exactly $|C|-1$ vertices of C (note that by definition of maximal clique, no vertex external to C can be connected with more than $|C|-1$ vertices of C). It is however believed that the "strict" condition can be relaxed from the previous result, thereby obtaining the following

Conjecture 1 *A subset C of vertices of G is a maximal clique if and only if $\mathbf{u}^c$ is a local maximum of the Motzkin-Straus problem.*

Once that the MCP is formulated in terms of maximizing a quadratic polynomial over $\mathcal{K}$, the use of the previously defined relaxation model immediately suggests itself. Consider, in fact, a network with as many computational units as there are vertices in G, and define the (symmetric and nonnegative) weight matrix to be

$$W = A \ , \tag{7}$$

where A is the adjacency matrix of G. Doing so, the network, starting from an arbitrary initial state $\mathbf{u}(0)$, will iteratively maximize the function f defined in (6), and will eventually approach a fixed point $\mathbf{u}^*$ corresponding to a maximum of f. This solution can then be employed to get an estimate of the maximum clique size, as seen in Theorem 2. Additionally, if the converged solution $\mathbf{u}^*$ happens to be the characteristic vector of some subset of vertices of G, then we are even able to directly determine the clique itself from the nonzero components of $\mathbf{u}^*$. Note that, since the algorithm essentially performs a gradient ascent search in $\mathcal{K}$, we are not in principle guaranteed that it will eventually converge to a global maximum of (6). In the light of the preceding results, therefore, what we can say is that the process can find only maximal cliques of G, hoping they have a large size. This is a common feature of most heuristic clique finding procedures [11], and neural networks are no exception (e.g., [14, 16]).

Observe that, since the process cannot leave the boundary of $\mathcal{K}$, it is preferable that the relaxation process begins with an interior point (i.e., $u_i(0) > 0$ for all i). A reasonable choice is to make the process start out in the vicinity of the center of the simplex $\mathcal{K}$: this would avoid that the search be biased in favor of any particular vertex in G. However, more sophisticated initialization strategies may be employed which take into account the connectivity structure of the graph at hand.

4 EXPERIMENTAL RESULTS

In order to assess the effectiveness of the proposed clique finding model, extensive simulations were carried out over randomly generated graphs of various sizes and densities.[2] In the first part of this study, we considered graphs with a relatively small number of vertices, i.e., with up to 500 vertices and over 60,000 edges. This allowed us to compare our results with those obtained using well-known exact clique

[2] The density δ of a graph G with n vertices is defined to be the ratio between the number of edges in G and $\binom{n}{2}$.

finding algorithms. In particular, the Bron-Kerbosch (BK) backtracking-based procedure [31], and the partially enumerative algorithm developed by Carraghan and Pardalos (CP) [32] were employed for comparison. We found the BK algorithm attractive because it is able to generate *all* cliques of a graph, and this allowed us to better understand certain phenomena associated with our relaxation model. When the computational time of BK became prohibitively long, we resorted to the CP algorithm, which is considerably faster than BK, but finds just *one* maximum clique.

The sizes considered in this series of experiments were $n = 10$, 25, 50, 75, 100, 250, 500 and the densities were $\delta = 0.10$, 0.25, 0.50, 0.75, 0.90. Indeed, for 250- and 500-vertex graphs the densities 0.75 and 0.90 were excluded, because even the CP algorithm became excessively slow in these cases. For each pair (n, δ), 100 different graphs with n vertices and approximate density δ were obtained so that, overall, 3100 graphs were employed. For each of them, the corresponding relaxation network was constructed as described in the previous section and then run by starting the process from the vector $\mathbf{u}(0) = (1/n, \cdots, 1/n)^T$, which corresponds to the center of the domain $\mathcal{K}$. We tried also to make the relaxation search begin with noisy versions of this vector, but no significant difference in performance was observed. To get a high-precision final solution, the network was allowed to iterate until the (squared) distance between two successive states became smaller than 10^{-10}. Each of the 3100 graphs were also given as input to the BK/CP algorithms in order to determine its actual maximum clique size k and then compute the corresponding global maximum α, according to the Motzkin-Straus theorem. Specifically, the BK algorithm was run over all graphs whose size n was less than or equal to 100; the exception was the 100-vertex 0.90-density case where, for reasons of computational expediency, we resorted to the CP algorithm. The CP algorithm was also employed for all 250- and 500-vertex graphs.

To systematically evaluate the goodness of the solutions found by the network, the following recently proposed "quality" measure was adopted [33]:

$$Q = (f_{ave} - f_{relax}) / (f_{ave} - \alpha) \quad , \tag{8}$$

where f_{ave} is the average value of the function f calculated by collecting a sufficient number of random points in $\mathcal{K}$, f_{relax} is the value of f associated with the solution found by the relaxation network, and α (as usual) represents the global maximum of f over $\mathcal{K}$ calculated using the BK/CP algorithms. From the above definition, it is apparent that values of Q near to zero correspond to very poor solutions (i.e., $f_{ave} \approx f_{relax}$), whereas values of Q near to one correspond to very good solutions (i.e., $f_{relax} \approx \alpha$).

The results of the simulations are summarized in Table 1 which contains, for each n and δ, the mean value of Q averaged over the 100 trials. In Table 2, the average clique sizes found by both the relaxation and the BK/CP algorithms are shown. The entries here are of the form k_{relax}/k_{opt}, where k_{relax} and k_{opt} denote

Table 1: Average values of the Q-measure for various sizes and densities

n \ δ	0.10	0.25	0.50	0.75	0.90
10	0.99	0.99	1.00	1.00	1.00
25	0.99	0.99	0.99	1.00	1.00
50	0.99	0.99	0.99	1.00	1.00
75	0.97	1.00	0.96	1.00	0.99
100	0.92	0.97	0.90	1.00	1.00
250	0.93	0.92	0.81	—	—
500	0.92	0.92	0.80	—	—

the average maximum clique size found by the relaxation process and the BK/CP algorithms, respectively. As can be seen, the relaxation labeling network, despite its inability to escape from local optima, did a good job of determining the size of the maximum cliques, even for high values of n and δ. This suggests that global maxima of the objective function f have large basins of attraction. In Table 3, the median number of iterations performed by the network to get a solution, for all cases considered, is reported.

As we have noted in the previous section, there may be a number of infeasible maxima associated with the Motzkin-Straus program [20] and, in principle, nothing prevents the relaxation process at converging towards one such solution. It should be clear that infeasible solutions represent an undesirable feature as they do not permit us to recover the vertices comprising a clique directly from their structure. The use of the BK algorithm over graphs with at most 100 vertices allowed us to discover certain interesting regularities in the behavior of our algorithm. Specifically, we observed that when a graph contained either a unique maximum clique or two or more disjoint maximum cliques, the network systematically converged to a fixed point $\mathbf{u}^*$ having the form of a characteristic vector. Unfortunately, when two or more maximum cliques shared some vertices this did not happen. However, we found that in such cases the network typically approaches a solution where all nonzero components correspond to the nodes of these cliques and, additionally, the components associated with the shared vertices have maximum value (typically of the form $1/k$, k being the clique size). This allows one to determine the maximum clique with little additional effort. We note that these empirical findings are perfectly in accordance with certain theoretical results presented in a companion paper [20], where it is shown that, under certain circumstances, all points belonging to the convex hull of the characteristic vectors of maximum cliques are infeasible solutions of the Motzkin-Straus program.

In order to evaluate the scaling properties of the proposed algorithm, in the second part of our study we concentrated on larger problem dimensions. The number of vertices considered in this case were $n = 1000$, 1500, and 2000, whereas the only

Table 2: Average clique sizes found by relaxation (left) and the BK/CP algorithms (right)

n \ δ	0.10	0.25	0.50	0.75	0.90
10	2.10/2.10	2.70/2.70	3.57/3.57	4.93/4.93	6.07/6.07
25	2.85/2.85	3.79/3.79	5.56/5.56	8.82/8.82	12.49/12.49
50	3.08/3.08	4.50/4.51	7.19/7.21	12.31/12.36	20.21/20.22
75	3.32/3.50	5.06/5.08	8.11/8.35	14.47/15.14	24.17/24.96
100	3.36/3.93	5.16/5.51	8.48/9.10	16.31/16.87	27.71/29.22
250	3.82/4.51	5.50/6.84	9.41/11.41	—	—
500	4.09/5.01	6.02/7.70	10.28/13.14	—	—

density considered was $\delta = 0.50$. For each size n, ten graphs were generated randomly and then given as input to the relaxation network. In this case, the process was terminated when the squared distance between two successive states became smaller than 10^{-7}. Unlike the previous series of experiments, where conventional exact algorithms were capable of solving the problem within reasonable computational time, in this case no such procedure is known to exist. Hence, to gauge the quality of the solutions found we resorted to a well-known result due to Matula, which accurately predicts the size of the maximum clique in a random graph, when the number of vertices is sufficiently large [14]. Let

$$M(n,\delta) = 2\log_{1/\delta} n - 2\log_{1/\delta}\log_{1/\delta} n + 2\log_{1/\delta}\frac{e}{2} + 1\ . \tag{9}$$

Matula proved that, as $n \to \infty$, the size of the maximum clique in an n-vertex δ-density random graph is either $\lfloor M(n,\delta)\rfloor$ or $\lceil M(n,\delta)\rceil$ with probability tending to 1, where $\lfloor x\rfloor$ denotes the largest integer less than or equal to x, and $\lceil x\rceil$ denotes the smallest integer greater than or equal to x. Interestingly, it was also shown that the size of the smallest maximal clique is expected to be $M(n,\delta)/2$ [14]. The results of our simulations are reported in Table 4, where the average clique size found by the relaxation process (denoted by k_{relax}), the expected maximum clique size $M(n, 0.5)$, and the expected size of the smallest maximal clique $M(n, 0.5)/2$ are shown, together with the median number of iterations performed by the relaxation process. The results are encouraging. The clique sizes found by the relaxation network are in fact significantly larger than the estimated minimum, thereby contradicting what is known as the Jerrum conjecture [14], which states that in a large 0.5-density random graph it may be hard to find a clique whose size is even a bit larger than that of the smallest maximal clique. A conclusion similar to ours has also been drawn by Jagota in [14]. It is interesting to note how the number of iterations of the relaxation process was approximately constant, that is independent on the number of vertices in the graph.

It may be interesting to compare our results with those obtained by others.

Table 3: Median number of relaxation iterations to get a solution

δ / n	0.10	0.25	0.50	0.75	0.90
10	54	54	56	99	119
25	36	64	118	175	224
50	53	84	153	268	367
75	59	98	160	284	513
100	82	112	187	369	552
250	96	150	223	—	—
500	102	160	262	—	—

Jagota [14] reports, among others, results over random graphs of size 100 and densities $\delta = 0.50$ and 0.90, averaged over 50 trials, for a number of neural network heuristics. The best results over the 0.50-density graphs were obtained by a mean field annealing (MFA) network and a stochastic steepest descent (SSD) network: they were able to generate cliques of (average) size 8.5 and 8.6, respectively, a result that is nearly identical to the one obtained by our model. On the 0.90-density graphs, the MFA algorithm arrived at an average clique size of 30.02, while the SSD produced cliques of size 28.76 (on the average). On our 100-vertex 0.90-density graphs we were instead able to get an average clique size of 27.71. It should be stressed, however, that the MFA algorithm not only is computationally expensive but requires also a number of parameters that are difficult to tune properly, and typically depend on the particular distribution and size of graphs [15]. In [14], Jagota reports that his MFA network required more than one hour to find a clique on 400-vertex graphs, when run on a semi-loaded SparcStation. We report that our algorithm took about 30 minutes on a SparcStation 10 to converge, when run on a 2000-vertex graph. Recently, the computational time of MFA was dramatically reduced by both implementing the algorithm on a parallel 32-processor Connection Machine and employing a two-temperature annealing schedule [15]. The problem of determining a good problem-independent initial temperature, however, is still an unresolved one. In addition, we note that the results reported in Jagota's paper for the SSD algorithm represent the best solution found among 100 different runs of the algorithm, over each graph. By contrast, our relaxation algorithm was run only once for each graph. If we compare our results with those obtained by Jagota using a continuous Hopfield dynamics (CHD), which is more close in spirit to the algorithm proposed here (i.e., it is continuous and requires no annealing) we find that our algorithm outperforms CHD over the 0-50 density graphs (i.e., 8.48 vs. 7.44), whereas the two algorithms obtained equivalent results over the 0.90-density case (i.e., 27.71 vs. 27.92)—again, we note that the CHD algorithm, in contrast to ours, requires a lengthy tuning phase to properly set a number of parameters. This seems to confirm an observation of Pardalos and Xue [11] who argued that the

Table 4: Results over large 0.5-density graphs

n	k_{relax}	$M(n, 0.5)$	$M(n, 0.5)/2$	Iterations
1000	11.10	15.18	7.59	196
1500	11.70	16.19	8.09	193.5
2000	12.00	16.91	8.45	197

choice of a proper formulation is of crucial importance in solving the MCP.

Lin and Lee [12] also reported experiments on random graphs, and we note that their results over 10-, 50-, 100-, and 250-vertex random graphs, with density 0.10, 0.50, and 0.90, are essentially equivalent to those obtained by our model. Lin and Lee's network, however, consists of $2(n+1)$ computational units, and employs real-valued connection weights.

Finally, our algorithm seems to be superior to the one developed by Funabiki *et al.* [13]. Specifically, on ten 100-vertex random graphs with density 0.25, 0.50, and 0.75, they found an average clique size of 4.2, 8.0, and 14.1, respectively, while our averages (over 100 graphs) are 5.16, 8.48, and 16.31. Their results on ten 500-vertex graphs with density 0.25 and 0.50 were 6.2 and 9.4, respectively, while ours are 6.02 and 10.28. Finally, on ten 1000-vertex 0.5-density graphs they obtained an average clique size of 10.4; our average is 11.10.

5 CONCLUSIONS

We have shown how the classical problem of finding the maximum clique of an undirected graph can be approximately solved by means of a simple yet powerful relaxation labeling network. This is a parallel distributed computational model that is widely popular within the machine vision domain, and has recently proven to possess interesting dynamical properties and learning abilities. Based on a result due to Motzkin and Straus, we have seen how the MCP can naturally be mapped onto a relaxation labeling network which is capable of maximizing quadratic polynomials over a certain linearly constrained domain. The model is especially suited for parallel implementation, and is attractive for its operational simplicity, since no operational parameters need to be determined. Extensive simulations over randomly generated graphs with up to 2000 vertices and about one million edges have demonstrated the effectiveness of the proposed approach both in terms of quality of solutions and speed.

Before concluding the paper, we note that the performance of our relaxation model can be significantly improved by employing some annealing strategy, so as to help the process escape from poor local optima. In this case, however, we would be faced with the non-trivial problem of determining a good annealing schedule, a choice which typically depends on the distribution and size of graphs. Moreover, running time would increase considerably. One problem associated with the proposed model relates to the existence of infeasible maxima in the Motzkin-Straus program which have been found both theoretically and experimentally. Infeasible solutions should be avoided because they do not allow us to determine the nodes comprising the clique directly from the structure of the final solution. One possible approach to solve this problem is to introduce some penalty term into the Motzkin-Straus objective function which would discourage infeasible solutions, thereby favoring solutions having the form of characteristic vectors. It is not completely clear, however, what kind of penalty term could serve our purpose.

References

[1] T. J. Sejnowski. Open questions about computation in cerebral cortex. In J. McClelland and D. E. Rumelhart (Eds.), *Parallel Distributed Processing, Vol. 2*, pp. 372–389. MIT Press, Cambridge, MA, 1986.

[2] J. J. Hopfield and D. W. Tank. "Neural" computation of decisions in optimization problems. *Biol. Cybern.*, 52:141–152, 1995.

[3] G. E. Hinton and T. J. Sejnowski. Optimal perceptual inference. In *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, pp. 448–453, Washington, DC, 1983.

[4] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[5] A. Rosenfeld, R. A. Hummel, and S. W. Zucker. Scene labeling by relaxation operations. *IEEE Trans. Syst. Man Cybern.*, 6(6):420–433, 1976.

[6] M. Pelillo. On the dynamics of relaxation labeling processes. In *Proc. IEEE Int. Conf. Neural Networks*, pp. 1006–1011, Orlando, FL, 1994.

[7] M. Pelillo and M. Refice. Learning compatibility coefficients for relaxation labeling processes. *IEEE Trans. Pattern Anal. Machine Intell.*, 16(9):933–945, 1994.

[8] S. W. Zucker, A. Dobbins, and L. Iverson. Two stages of curve detection suggest two styles of visual computation. *Neural Computation*, 1:68–81, 1989.

[9] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[10] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proc. 32nd Ann. Symp. Found. Comput. Sci.*, pp. 2–12, San Juan, Puerto Rico, 1991.

[11] P. M. Pardalos and J. Xue. The maximum clique problem. *J. Global Optim.*, 4:301–328, 1994.

[12] F. Lin and K. Lee. A parallel computation network for the maximum clique problem. In *Proc. 1st Int. Conf. Fuzzy Theory Tech.*, Baton Rouge, Louisiana, 1992.

[13] N. Funabiki, Y. Takefuji, and K. C. Lee. A neural network model for finding a near-maximum clique. *J. Parallel Distrib. Comput.*, 14(3):340–344, 1992.

[14] A. Jagota. Approximating maximum clique with a Hopfield network. *IEEE Trans. Neural Networks*, 6(3):724–735, 1995.

[15] A. Jagota, L. Sanchis, and R. Ganesan. Approximately solving maximum clique using neural networks and related heuristics. In D. S. Johnson and M. Trick (Eds.), *Second DIMACS Implementation Challenge.* in press, 1995.

[16] T. Grossman. Applying the INN model to the MaxClique problem. In D. S. Johnson and M. Trick (Eds.), *Second DIMACS Implementation Challenge.* in press, 1995.

[17] T. S. Motzkin and E. G. Straus. Maxima for graphs and a new proof of a theorem of Turán. *Canad. J. Math.*, 17:533–540, 1965.

[18] P. M. Pardalos and A. T. Phillips. A global optimization approach for solving the maximum clique problem. *Int. J. Computer Math.*, 33:209–216, 1990.

[19] L. E. Gibbons, D. W. Hearn, and P. M. Pardalos. A continuous based heuristic for the maximum clique problem. In D. S. Johnson and M. Trick (Eds.), *Second DIMACS Implementation Challenge.* in press, 1995.

[20] M. Pelillo and A. Jagota. Feasible and infeasible maxima in a quadratic program for maximum clique. *J. Artif. Neural Networks*, this issue, 1995.

[21] J. S. Denker. Neural networks refinements and extensions. In J. S. Denker (Ed.), *Neural Networks for Computing*, pp. 121–128. American Institute of Physics, New York, 1986.

[22] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi. An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition. *Bell Syst. Tech. J.*, 62(4):1035–1074, 1983.

[23] C. Peterson and B. Söderberg. A new method for mapping optimization problems onto neural networks. *Int. J. Neural Syst.*, 1(1):3–22, 1989.

[24] J. S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In F. Fogelman-Soulié and J. Hérault (Eds.), *Neurocomputing: Algorithms, Architectures, and Complexity*, pp. 227–236. Springer-Verlag, Berlin, 1990.

[25] S. J. Nowlan. Maximum likelihood competitive learning. In D. S. Touretzky (Ed.), *Advances in Neural Information Processing Systems 2*, pp. 574–582. Morgan Kaufmann, San Mateo, CA, 1990.

[26] L. E. Baum and J. A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bull. Amer. Math. Soc.*, 73:360–363, 1967.

[27] L. E. Baum and G. R. Sell. Growth transformations for functions on manifolds. *Pacif. J. Math.*, 27(2):211–227, 1968.

[28] G. R. Blakley. Homogeneous nonnegative symmetric quadratic transformations. *Bull. Amer. Math. Soc.*, 70:712–715, 1964.

[29] W. J. Ewens. *Mathematical Population Genetics*. Springer-Verlag, Berlin, 1979.

[30] L. E. Gibbons, D. W. Hearn, P. M. Pardalos, and M. V. Ramana. Continuous characterization of the maximum clique problem. *Math. Oper. Res.*, submitted for publication, 1995.

[31] C. Bron and J. Kerbosch. Finding all cliques of an undirected graph. *Comm. ACM*, 16(9):575–577, 1973.

[32] R. Carraghan and P. M. Pardalos. An exact algorithm for the maximum clique problem. *Oper. Res. Lett.*, 9:375–382, 1990.

[33] P. W. Protzel. Comparative performance measure for neural networks solving optimization problems. In *Proc. Int. J. Conf. Neural Networks*, vol. II, pp. 523–526, Washington, DC, 1990.