A Saturated Linear Dynamical Network for Approximating Maximum Clique

Ferhan Pekergin, Ömer Morgül, and Cüneyt Güzeliş

***Abstract*—We use a saturated linear gradient dynamical network for finding an approximate solution to the maximum clique problem. We show that for almost all initial conditions, any solution of the network defined on a closed hypercube reaches one of the vertices of the hypercube, and any such vertex corresponds to a maximal clique. We examine the performance of the method on a set of random graphs and compare the results with those of some existing methods. The proposed model presents a simple continuous, yet powerful, solution in approximating maximum clique, which may outperform many relatively complex methods, e.g., Hopfield-type neural network based methods and conventional heuristics.**

***Index Terms*— Combinatorial optimization, gradient systems, max clique, neural networks.**

I. Introduction

GRADIENT dynamical systems are described by a set of differential equations in a state equation form whose vector field is produced by the gradient of a scalar function called energy. These systems do not have complex dynamics such as oscillation, so that any bounded solution of them converges to one of the equilibrium points which are indeed extrema of the associated energy function [1]. On the other hand, so called gradient-like systems (e.g., Hopfield-type neural networks) are, in fact, not gradient systems although they have the same kind of dynamics. As a consequence of their dynamical properties, gradient and gradient-like systems have been widely used as natural models for solving unconstrained minimization problems by considering the cost function as the energy (see, e.g., [2]). Constrained minimization problems can also be solved in the same way, by adding to the cost some penalty function terms representing constraint violations, hence transforming the problem into an unconstrained one. Many continuous optimization methods, and almost all analog neural network architectures designed for real-time optimization, are based on the explained gradient systems approach. The application area of gradient system based methods covers a very broad class of optimization problems, including nonlinear, discrete and combinatorial (even of NP-hard) ones, such as the traveling salesman problem.

This paper employs a saturated linear (gradient) dynamical network for approximating the maximum clique problem which can be formulated as a special kind of quadratic 0–1 programming problem. The quadratic cost function, which is indefinite, has been taken as the energy for the network. The solutions produced by unstable linear dynamics are saturated on the hypersurface of closed unit hypercube so that saturated mode solutions eventually satisfy 0–1 constraints as reaching one of the vertices of the hypercube. We show that any such vertex is a valid solution to the problem, i.e., it defines a clique and furthermore provides the maximality. Such a network model was proposed previously in [3] with a different parameter setting for the design of an associative memory where stored patterns correspond to stable vertices. In the models of [3] and the present paper, bipolarity (or unipolarity) condition on the optimization variables is provided by the saturation mechanism described above. Such a handling of discrete variable condition is quite unusual in the literature where basically two techniques are used. In the first approach [4], [2], penalty terms ($x_i(1 - x_i)$ for unipolar or $1 - x_i^2$ for bipolar variables) are added to the cost function and then discrete constraints are relaxed into continuous ones as linear inequalities defining a hypercube. As discussed in [2] for quadratic cost functions, the resulting cost can be made concave in this way, and its continuous local minima take place at the hypercube vertices, but some of these continuous minima may not correspond to discrete local minima of the original problem. The second technique that most analog neural network models exploit is to introduce extra continuous variables such that original optimization variables take discrete values as a function of these new variables. Section IV of this paper shows how such an approach can be used for the considered maximum clique problem.

The problem of finding a maximum clique to which we propose an approximate solution is an NP-hard discrete optimization problem [5]. It is computationally intractable even to approximate with certain absolute performance bounds [6]. Several practical problems arising in a diverse field, e.g., pattern recognition, computer vision, information processing etc., and also a number of graph theoretic problems can be transformed into maximum clique problem. It is, therefore, of interest to develop methods for finding exact and also approximate solution to it. The methods available in the literature fall into two categories: i) Global methods ensuring a maximum clique to be found (see [7] for a review), ii) Local methods capable of finding maximal cliques providing approximations in some degrees. Recently, several neural network models which are mostly of Hopfield type [2], have

Manuscript received January 29, 1998; revised June 15, 1998. This paper was recommended by Associate Editor D. Liu.

F. Pekergin is with the Laboratoire d'Informatique de Paris-Nord, Université Paris-Nord, 93430-Villetaneuse, France.

Ö. Morgül is with the Department of Electrical Engineering, Bilkent University, 06533-Bilkent, Ankara, Turkey.

C. Güzeliş is with the Faculty of Electrical and Electronics Engineering, Istanbul Technical University, Maslak 80626, Istanbul, Turkey.

Publisher Item Identifier S 1057-7122(99)04746-7.

been proposed (see, [8], [9], and related references therein). All of the existing neural network based methods and the method of this paper could be considered in the second category. The saturated linear gradient network used here is closely related to the continuous time Hopfield network. As stated in [3], this model can be implemented by an analog electronic circuit in a configuration the same with those of RC opamps implementations of Hopfield networks but with using ideal (lossless) integrators with saturation instead of lossy integrators and nonlinear amplifiers. The model considered in this paper is linear on any k-dimensional ($k = 1, 2, \cdots, n$) faces of the hypercube $[0,1]^n$. This makes the model mathematically and computationally tractable. Despite of its simplicity, as observed from the computer experiments it has a better approximation performance than the existing Hopfield type neural networks and conventional heuristic methods.

This paper is organized as follows. In Section II we present a formulation of the maximum clique problem. In Section III we show that almost all solutions of saturated linear dynamical network reach some vertices of the unit hypercube and any vertex reached corresponds to a maximal clique. In Section IV we give Grossberg type version of the proposed model which is described by state equations with continuous right hand side as opposed to the original version. In Section V we present the results of some computer simulations and give a performance comparison with some other methods. Finally we give some concluding remarks.

II. Formulation of Maximum Clique Problem

In the following, the definitions of maximum clique and maximum independent set which indeed corresponds to maximum clique in the complement graph will be given. Due to its simplicity in formulation, only maximum independent set problem will be considered and maximum clique problem will then be treated based on an optimization formulation obtained for maximum independent set. All statements which will be made for independent sets could be considered as the statements for cliques in the complement graph. Although proofs for some facts given in this section are available in the literature (see, e.g., [10]), they are also repeated here in order to make the paper self-contained. On the other hand, to avoid confusion, note that throughout the paper vertex will be used alternatingly for hypercube corners and graph nodes. Any considered graph will be assumed to have no loop, no more than one edge associated to a vertex pair and have at least one edge.

Let $G = (V, E)$ be an undirected graph, where V is the set of vertices and $E \subset V \times V$ is the set of edges. A subset $S \subset V$ of vertices is called a clique if for every pair of vertices in S, there is an edge in E, i.e., the subgraph induced by S is complete. A maximal clique S is a clique whose proper extensions are not cliques, i.e., for any S', if $S \subset S'$ and $S \neq S'$ then S' is not a clique. A maximum clique of G is a clique whose cardinality is maximum. Note that, by definition a maximum clique is also maximal, but the converse is not necessarily true.

An equivalent characterization of maximum clique problem can be given in terms of independent sets of vertices. Let $G = (V, E)$ be an undirected graph. A subset $S \subset V$ of vertices is called an independent set of G if its vertices are pairwise nonadjacent, that is if vertices $v_i, v_j \in S, i \neq j$, then $(v_i, v_j) \notin E$. A maximal independent set of G is a subset S of vertices whose proper extensions are not independent sets, i.e., for any S', if $S \subset S'$ and $S' \neq S$, then S' is not an independent set. A maximum independent set of G is an independent set whose cardinality is maximum. Note that, by definition a maximum independent set is also maximal, but the converse is not necessarily true.

It is well-known that the maximum clique problem for a graph is equivalent to the maximum independent set problem for the complement of the graph. More precisely, let $G = (V, E)$ be an undirected graph and let $\overline{G} = (V, \overline{E})$ be its complement, i.e., $\overline{E} = V \times V \setminus E$. Then, it is well known that S is a maximal (maximum) clique of G if and only if S is a maximal (maximum) independent set of $\overline{G}$ (see, e.g., [10]).

The problems presented above are known to be NP-complete [5], and even their absolute approximations are NP-hard [6]. Hence, simple algorithms which yield good suboptimal solutions in reasonable amount of time may be quite useful for practical problems related to these problems.

Maximal independent set problem (or equivalently maximal clique problem) can be stated as a quadratic 0-1 programming problem in various ways, see e.g., [7], [10], [9]. Also, by using some well known results, the problem can be transformed into a continuous and concave minimization problem for a quadratic cost function on the unit hypercube $[0,1]^n$, where $n = |V|$ (see e.g., [4], [2]). Once the problem is formulated as a continuous and constrained optimization problem, various standard neural optimization schemes could be used to approximate its solution (e.g., see [2]). Most of these schemes utilize standard gradient descent type optimization technique. Consequently, they might converge to a local minimum, and this minimum may not correspond to a valid solution, i.e., a maximal clique or maximal independent set. Moreover, in most of these schemes there are various parameters which must be adjusted beforehand. In most of the cases, these parameters affect the solution crucially, and the setting of them are often based on a trial-and-error procedure, which is not a trivial task.

We formulate the problem as a quadratic 0-1 optimization problem as follows. Let $G = (V, E)$ be an undirected graph. Let $n = |V|$ be the number of vertices, and let $v_i \in V$, $i = 1, 2, \cdots, n$ denote the vertices. Let $A \in \{0,1\}^{n \times n}$ be the adjacency matrix of G, i.e., for $i, j = 1, 2, \cdots, n$, $a_{i,j} = a_{j,i} = 1$ if and only if $(v_i, v_j) \in E$. Note that A is a symmetric matrix, and $a_{i,i} = 0$ for $i = 1, 2, \cdots, n$.

We first state the following simple fact:

Fact 1: $A \in \{0,1\}^{n \times n}$ is an indefinite matrix.

Proof: Since $A = A^T$ all eigenvalues of A are real numbers, where the superscript T denotes the transpose. Moreover, $trace\{A\} = 0$, hence so is the sum of eigenvalues of A. Since the associated graph has at least one edge, then A has rank not less than 1. This implies that A necessarily has both negative and positive eigenvalues, hence is indefinite. □

Fact 2 shows that the adjacency matrix A is closely related to the characterization of independent sets. Let $S \subset V$ be a subset of vertices and let $x^S \in \{0,1\}^n$ be its characteristic vector, i.e., $x_i^S = 1$ if and only if $v_i \in S$ and $x_i^S = 0$ if and only if $v_i \notin S$ for $i = 1, 2, \cdots, n$.

Fact 2: S is an independent set if and only if its characteristic vector x^S satisfies the quadratic equation $(x^S)^T A x^S = 0$.

Proof: First note that $(x)^T Ax \geq 0$ for any $x \in \{0,1\}^n$. Let S be an independent set and let $x^S \in \{0,1\}^n$ be its characteristic vector. If $x_i^S = 1$ and $x_j^S = 1$ for any $i, j = 1, 2, \cdots, n$ such that $j \neq i$, then $a_{i,j} = 0$ which easily follows from the definitions of independent set and adjacency matrix. Then, necessarily we have $(Ax^S)_i = 0$ for an i so that $x_i^S = 1$, for otherwise there must be at least one $j = 1, 2, \cdots, n$ such that $a_{i,j} = 1$ and $x_j^S = 1$, which is a contradiction. It then follows $(x^S)^T A x^S = 0$. Conversely, let $(x^S)^T A x^S = 0$ for some $x^S \in \{0,1\}^n$, and let $S \subset V$ be the subset of vertices whose characteristic vector is given by x^S. Let us have $x_i^S = 1$ and $(Ax^S)_i = 0$. If $x_j^S = 1$ for some j, then necessarily we have $a_{i,j} = 0$, which means that $v_i, v_j \in S$ and $(v_i, v_j) \notin E$. Hence, S is an independent set. □

We note that Fact 2 does not characterize maximal independent sets. A standard characterization of maximum independent set problem for a $G = (V, E)$ as a quadratic 0-1 optimization problem is given as follows:

$$\min f(x) := x^T A x - e^T x, \qquad x \in \{0,1\}^n, \tag{1}$$

where, $A \in \{0,1\}^{n \times n}$ is the adjacency matrix and $e = (1, 1, \cdots, 1)^T \in \boldsymbol{R}^n$.

Fact 3: Any $x^* \in \{0,1\}^n$ is a (discrete) global minimum of $f(x)$ given by (1) if and only if the set S such that $x^S = x^*$ is a maximum independent set for G.

Proof: If $x^* \in \{0,1\}^n$ is a global minimum, then necessarily we have

$$(x^*)^T A x^* = \sum_{\substack{(v_i, v_j) \in E \\ i > j}} 2 x_i^* x_j^* = 0.$$

Otherwise, we can find another vector $x^{**} \in \{0,1\}^n$ yielding a smaller cost. Hence x^* corresponds to a maximum independent set by Fact 2 and the assumptions. This proves the necessity. The sufficiency follows from Fact 2. □

The correspondence described by Fact 3 is true also in local sense. A point $x^* \in \{0,1\}^n$ is called a (discrete) local minimum of $f(x)$ if $f(x*) \leq f(x)$ for any $x \in \{0,1\}^n$ adjacent to x^*, i.e., $\Sigma_{i=1}^n |x_i - x_i^*| \leq 1$.

Fact 4: A point $x^* \in \{0,1\}^n$ is a discrete local minimum of $f(x)$ if and only if $x^S = x^*$ defines a maximal independent set for G.

Proof: Suppose that $x^* \in \{0,1\}^n$ is a discrete minimum but does not define a maximal independent set. It is clear from the proof of Fact 3 that when the independency or the maximality is violated by a characteristic vector x^S, the point $x^* = x^S$ cannot be a minimum. This proves the necessity, the sufficiency is clear by the definitions. □

III. Saturated Linear Dynamical Network

Starting from (1), let us define the following energy function:

$$V = x^T A x - e^T x \qquad x \in [0,1]^n \tag{2}$$

and based on (2), let us define the following gradient descent dynamics:

$$\dot{x} = -\tfrac{1}{2}\nabla V = \tfrac{1}{2}e - Ax. \tag{3}$$

Assume that $x \in \boldsymbol{R}^n$ and let us take the derivative of (2) with respect to time along the solutions of (3). Then we obtain:

$$\dot{V} = -\tfrac{1}{2}\|\nabla V\|^2 = -2\left\|Ax - \tfrac{1}{2}e\right\|^2, \tag{4}$$

hence the energy is a nonincreasing function of time along the solutions of (3). Moreover, by Fact 1, A is an indefinite matrix. Hence, even if (3) has an equilibrium point, it is necessarily of saddle type. Therefore, (3) represents an unstable dynamics, and unless (3) has an equilibrium point and $x(0)$ belongs to the stable manifold of that equilibrium point, the solutions of (3) are unbounded. Since A has both positive and negative eigenvalues, the union of such stable manifolds always has measure zero. It follows that for almost all initial conditions $x(0)$, the solutions of (3) are unstable and hence escape from any bounded region in $\boldsymbol{R}^n$ in finite time. This does not contradict (4), which states that the quadratic form (2) decreases along the solutions of (3), but (2) itself is an indefinite quadratic form hence is not bounded from below.

The unstable dynamics given by (3) is not useful if not restricted to the unit hypercube. To force the solutions to stay inside the unit hypercube, including the boundaries, we modify (3) as follows:

$$\dot{x}_i = \begin{cases} \tfrac{1}{2} - (Ax)_i, & \text{if } 0 \leq x_i \leq 1 \text{ except for the} \\ & \quad\text{following cases} \\ 0, & \text{if } x_i = 1 \text{ and } \tfrac{1}{2} - (Ax)_i \geq 0 \\ 0, & \text{if } x_i = 0 \text{ and } \tfrac{1}{2} - (Ax)_i \leq 0. \end{cases} \tag{5}$$

Note that although the right hand side of the differential equation given by (5) is discontinuous, for any initial condition $x(0) \in [0,1]^n$, there exists a unique solution to (5). Moreover, this solution is continuous with respect to time, but not differentiable.

The rationale behind using the discontinuous dynamics given by (5) is to restrict the dynamics given by (3) to the unit hypercube. We will show below that the quadratic form given by (2) continuously decreases along the solutions of (5), and due to the unstable behavior of (5) it eventually reaches to a vertex of the hypercube in finite time and stays there. We will also show that the converged vertex always corresponds to a maximal independent set.

Theorem 1: For all initial conditions $x(0) \in [0,1]^n$, the quadratic form given by (2) is a nonincreasing function of time along the solutions of (5).

Proof: Let us consider three cases: when the trajectories of (5) are inside the unit hypercube, when the trajectories are on a hypersurface $x_i = 0$ or $x_i = 1$.

Case 1: When $x(t) \in (0,1)^n$, the dynamics is given by (3); and (4) shows that V decreases along the solutions. There may be an equilibrium point of (3) in $(0,1)^n$, and $x(0)$ may be on the stable manifold of this equilibrium point. In such cases, the solutions converge to this equilibrium point. However, union of such stable manifolds has measure zero, and for almost all initial conditions $x(0) \in (0,1)^n$, the solutions of (3) grow and eventually hit the boundary of unit hypercube in finite time.

Case 2: Assume that $x_i(T) = 0$ for some time $T \geq 0$. Let $A_r \in \boldsymbol{R}^{m\times m}$ denote the matrix obtained by deleting the ith row and ith column of A where $m = n-1$. Let $b_r = \frac{1}{2}e_r \in \boldsymbol{R}^m$, where $e_r = (1,1,\cdots,1)^T$, and let $x_r = (x_1, x_2, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n)^T \in \boldsymbol{R}^m$. Then, (5) reduces to

$$\begin{aligned}\dot{x}_r &= b_r - A_r x_r, \qquad 0 < x_j < 1 \quad j = 1,2,\cdots,n,\\ &\quad j \neq i, \qquad (6)\\ x_i &= 0. \qquad (7)\end{aligned}$$

For this case, the quadratic form (2) becomes:

$$V_r = x_r^T A_r x_r - 2b_r^T x_r. \qquad (8)$$

Note that $V = V_r$ for $x_i = 0$. By differentiating (8) along (6), we get

$$\dot{V}_r = -2\|A_r x_r - b_r\|^2 \qquad (9)$$

which shows that quadratic form (2) continue to decrease on the hypersurface $x_i = 0$. Note that A_r is the adjacency matrix of the subgraph obtained by eliminating ith vertex of graph G. If the resulting subgraph has no edge, then A_r becomes a zero matrix. In this case, the reduced system (6) cannot have an equilibrium point, so the solutions cannot stay inside the hypersurface. When the subgraph has at least one edge, A_r becomes indefinite, thus representing an unstable dynamic. Now, there may be an equilibrium point of (6) on the hypersurface $x_i = 0$ and $x_r(T)$ may be on the stable manifold of such an equilibrium point. In such cases, the solution converges to this equilibrium point. However, the union of such stable manifolds has measure zero in $[0,1]^m$, and for almost all $x_r(T)$ the solutions of (6) grow and eventually hit the boundaries of the hypersurface defined by $x_i = 0$ or escape from this hypersurface, i.e., $x_i > 0$ for some $t \geq T$ (which means we fall into the first case).

Case 3: Assume that $x_i(T) = 1$ for some time $T \geq 0$. Let $A_r \in \boldsymbol{R}^{m\times m}$ and $x_r \in \boldsymbol{R}^m$ be as defined in Case 2. Let us define $\hat{b}_r \in \boldsymbol{R}^m$ as follows:

$$(\hat{b}_r)_j = -\tfrac{1}{2}, \quad \text{if } a_{i,j} = 1, \qquad (10)$$

$$(\hat{b}_r)_j = \tfrac{1}{2}, \quad \text{if } a_{i,j} = 0, j \neq i. \qquad (11)$$

Then (5) reduces to

$$\begin{aligned}\dot{x}_r &= \hat{b}_r - A_r x_r \qquad 0 < x_j < 1 \quad j = 1,2,\cdots,n,\\ &\quad j \neq i, \qquad (12)\\ x_i &= 1. \qquad (13)\end{aligned}$$

For $x_i = 1$ the quadratic form (2) becomes

$$\hat{V}_r = x_r^T A_r x_r - 2\hat{b}_r^T x_r - 1. \qquad (14)$$

Note that $V = \hat{V}_r$ for $x_i = 1$. By differentiating (14) along the solutions of (12) we obtain

$$\dot{\hat{V}}_r = -2\|A_r x_r - \hat{b}_r\|^2. \qquad (15)$$

By using the same argument given in Case 2 just after (9), it follows that the quadratic form (2) continually decreases on the hypersurface $x_i = 1$, and for almost all $x_r(T) \in [0,1]^m$, the trajectories eventually hit the boundary of the hypersurface $x_i = 1$. As opposed to Case 2, in this case the trajectory cannot leave the hypersurface $x_i = 1$ (see Remark 1 below).

For any $x(0) \in [0,1]^n$, as the corresponding trajectory evolves in time it will hit the boundary of unit hypercube and at least one of the Cases 2 and 3 will be valid. By consecutive application of these cases, we conclude that the quadratic form continually decreases along the solutions of (5). □

Remark 1: Consider the case $x_i = 1$ and $1/2 - (Ax)_i \geq 0$ for some t. Since $x_i = 1$, then $1/2 - (Ax)_j < 0$ for all j such that $a_{i,j} = 1$. Therefore, as time evolves, any x_j with $a_{i,j} = 1$ cannot increase, hence, $1/2 - (Ax)_i = \frac{1}{2} - \Sigma_{j=1}^n a_{i,j}x_j$ remains nonnegative. This means that any trajectory that hits the hypersurface $x_i = 1$ cannot leave it. □

By using Theorem 1 we can prove the following convergence result.

Corollary 1: For almost all $x(0) \in [0,1]^n$ in finite time, the trajectories of (5) reach one of the vertices of the unit hypercube and stay there thereafter.

Proof: By Theorem 1, the quadratic function V decreases along the trajectories of (5). Since the solutions of (5) are bounded and, since V is bounded below on the unit hypercube, it follows that V, as a function of time, converges to a constant V_o. Let $L = \{x \in [0,1]^n | V(x) = V_o\}$ denote the level set of V. Then, by continuity of $x(\cdot)$ with respect to time and $V(\cdot)$ with respect to x, it follows that $x(t)$ converges to L as t evolves. Since (3) inside the hypercube and (6) and (12) on the hypersurfaces represent unstable dynamics, the trajectories escape from the equilibrium point for almost all initial conditions. Hence, without loss of generality we may assume that L does not contain an equilibrium point of (3), (6), or (12). Note that since the unit hypercube is a compact set, it follows that L is not empty. Let $x \in L$ such that $x(t) \to x$. Note that if the right-hand side of (5) were a Lipschitz function of x, then a trajectory which converges to an equilibrium point would not reach it in a finite time. However, since in our case the right-hand side of (5) is discontinuous, this conclusion does not hold in general and, as argued in Corollary 1, any trajectory reaches to a vertex in finite time. If $0 > x_i > 1$ for some $i = 1,2,\cdots,n$ then by Theorem 1 energy continues to decrease. Hence, we have $V(t) < V_o$ $\forall t > T$, which is a contradiction. Hence, necessarily x is a vertex and $x(t) = x$ $\forall t \geq T$. □

Next, we will show that the vertex $x \in \{0,1\}^n$ to which (5) converges actually corresponds to a maximal independent set.

Theorem 2: Let $x \in \{0,1\}^n$ be a point to which (5) converges for some initial conditions $x(0) \in [0,1]^n$. Let S be the set of vertices of the graph $G = (V, E)$ such that

$$S = \{v_i \in V | x_i = 1, \qquad i = 1,2,\cdots,n\}.$$

Then, S is a maximal independent set.

Proof: First, we will show that S is an independent set. Let $x_i = 1$. Then, from (5) it follows that $\frac{1}{2} - (Ax)_i \geq 0$. However, since $x \in \{0,1\}^n$, $(Ax)_i$ is necessarily a nonnegative integer. Hence, we have $(Ax)_i = 0$. Therefore, we have $x^T Ax = \Sigma_{i=1}^n x_i(Ax)_i = 0$. By Fact 2, S is an independent set.

Now, let S' be another set of vertices such that $S \subset S'$ and $S \neq S'$. Then, we have a vertex $v_j \in V$ such that $v_j \in S'$ but $v_j \notin S$. Hence, by (5) we must have $x_j = 0$ and $\frac{1}{2} - (Ax)_j \leq 0$. Let $x' \in \{0,1\}^n$ be the characteristic vector of S', i.e., $(x')_j = 1$ if and only if $v_j \in S'$ for $j = 1, 2, \cdots, n$. Then, $(x')_j = 1$ and since $S \subset S'$, we must have $(Ax')_j \geq (Ax)_j$. Since we have $(Ax)_j \geq \frac{1}{2}$, this implies that $(Ax')_j \geq \frac{1}{2}$, hence, $(x')^T Ax' \neq 0$. By Fact 2, S' cannot be an independent set. ☐

The next corollary follows from Theorem 2 and Fact 4.

Corollary 2: Let $x \in \{0,1\}^n$ be a point where (5) converges for some initial conditions $x(0) \in [0,1]^n$. Then, x is a discrete local minimum of (1). ☐

Remark 2: According to Theorem 1, for all initial conditions $x(0) \in [0,1]^n$, the quadratic form given by (2) is a nonincreasing function of time along the solutions of (5). In fact, it decreases monotonically except when $x(0)$ or $x(T)$ for some $T > 0$, corresponding to the instance when the trajectory hits a hypersurface, is an equilibrium point of the associated dynamics. However, such equilibrium points are necessarily of the saddle type and, consequently, the union of their stable manifolds is not dense in $[0,1]^n$. Therefore, for almost all initial conditions $x(0) \in [0,1]^n$, the quadratic form given by (2) continually decreases along the solutions of (5) and reaches a vertex in finite time. This point was also confirmed in our simulations, in which we choose the initial conditions randomly and, in each case, the corresponding trajectory reached a vertex. If a trajectory does not reach a vertex, which is a highly unlikely event, then we can make an arbitrary small change on $x(0)$ and the new trajectory will almost always converge to a vertex. ☐

IV. Relation with a Grossberg Neural Network

The dynamical system (5) for maximum independent set problem is related to a Grossberg-type neural network [2] as follows. Let the function $h(\cdot)$: $\boldsymbol{R} \to \boldsymbol{R}$ and $s(\cdot)$: $\boldsymbol{R}^n \to \boldsymbol{R}^n$ be defined as

$$h(y) = \begin{cases} 1, & \text{if } y \geq 1 \\ y, & \text{if } 0 \leq y \leq 1 \\ 0, & \text{if } y \leq 0 \end{cases} \tag{16}$$

$$s(x) = [h(x_1)\; h(x_2) \cdots h(x_n)]^T. \tag{17}$$

Consider the following dynamical system:

$$\dot{x} = \tfrac{1}{2}e - As(x). \tag{18}$$

Clearly, (18) can be considered as a Grossberg-type neural network. To understand the behavior of (18) let us assume that $x(0) \in (0,1)^n$. Then, $s(x) = x$ and the dynamics (18) becomes equivalent to (3). Therefore, the solutions of (18) increase and eventually hit the boundary of unit hypercube. If $x_i \leq 0$, then $(s(x))_i = 0$ and the dynamics associated with the rest of the variables reduces to (6). Also, if $x_i \geq 1$, then $(s(x))_i = 1$ and the dynamics associated with the rest of the variables reduce to (12). The difference between the behavior of (18) and (5) is the following. In (5) the solutions are restricted to the unit hypercube. However, in (18) the solution $x(t)$ continues to grow even if it reaches the boundary of the hypercube, but $s(x(t))$ eventually becomes bounded as converging to a vertex. This property could be summarized as follows.

Fact 5: For almost all $x(0) \in [0,1]^n$, the solutions $x(t)$ of (18) diverge to infinity. Moreover, there exists a $T \geq 0$ such that $s(x(t)) \in \{0,1\}^n$, and $s(x(t)) = s(x(T))$ for all $t \geq T$. Let $y^* = s(x(T))$ and let S denote the set of vertices corresponding to y^*. Then S is a maximal independent set of the graph G.

Proof: These results can be proven by using the ideas given in Theorem 1, Corollary 1 and Theorem 2, and thus are omitted here. ☐

Remark 3: Note that in (18), $x(t)$ remains unbounded and $s(x(t))$ converges to a vertex, which corresponds to a maximal independent set. Since (18) is unstable, $s(x(t))$ reaches a vertex in a finite time. Since $x(t)$ increases proportional to t asymptotically, at time T when $s(x(t)) \in \{0,1\}^n$ and $s(x(t)) = s(x(T))$ $t \geq T$, the solution $x(t)$ will not be too large at the time T. The dynamical system (18) should be reset afterwards to avoid further increase. Since the solutions of (18) continue to increase (in norm) and the solutions of (5) are restricted to unit hypercube in (5) then, in general, the vertices to which these two models converge might be different.

Also note that (18) could be implemented as an analog electronic circuit. However, as stated above, due to the unstable behavior of $x(t)$ this circuit should be turned off when a vertex is reached. ☐

V. Numerical Results

In this section, we illustrate the performance of the saturated linear dynamical network (SLDN) on random graphs of various vertex sizes and densities. As a primary performance measure, we consider average maximal clique sizes found by our method in several experiments. Herein, the average is taken over the test graphs generated with the same characteristics, i.e., the vertex sizes and densities. We consider also averages computed for the same test sets but taking into account only the best results obtained by five (and also ten) independent runs of our algorithm on each graph with randomly chosen initial conditions. The first performance measure provides an indirect comparison of our results with those reported in the literature for some methods which have been applied to random graphs with the same characteristics. The second measure is related to the ability of the method to find different search directions when it is started by different initial points. Considering the best solution among many ones is a natural performance improvement technique, used frequently for such local search methods. However, not every local method has this kind of improvement possibility.

We performed a set of direct comparisons between our model and a continuous Hopfield network proposed by Jagota in [8]. The reason for this comparison is the following: our model and the one proposed in [8] are both continuous and gradient-like systems. They are very similar also in some other respects. On the other hand, this continuous Hopfield network is compared in [8] with several other algorithms applied to the maximum clique problem. The information given in [8] about the performance of the methods available in the literature

TABLE I
AVERAGE CLIQUES SIZES FOUND FOR 100-, 400-, AND 1000-VERTEX GRAPHS WITH DENSITIES OF 0.25, 0.50, AND 0.75

		Overall Average		Av. over Bests among 5 Runs		Av. over Bests among 10 Runs	
$\|V\|$	Density	SLDN	CHD	SLDN	CHD	SLDN	CHD
100	0.25	4.83	4.48	5.21	4.58	5.30	4.62
	0.50	8.07	7.38	8.47	7.59	8.60	7.66
	0.75	15.05	13.87	15.63	14.24	15.76	14.40
400	0.25	5.70	5.53	6.34	5.96	6.56	6.08
	0.50	9.91	9.24	10.70	10.03	10.96	10.30
	0.75	20.44	18.79	21.8	19.93	22.24	20.28
1000	0.25	6.17	6.03	6.95	6.50	7.10	6.60
	0.50	10.93	10.25	12.05	11.10	12.50	11.4
	0.75	23.19	21.26	24.45	22.45	24.8	23.0

TABLE II
PERCENTAGE OF THE BETTER (B), EQUAL (E) AND WORSE (W) QUALITY SOLUTIONS OF THE SLDN WITH RESPECT TO THOSE OF CHD FOR 100-, 400-, AND 1000-VERTEX GRAPHS WITH DENSITIES OF 0.25, 0.50, AND 0.75

		Over all runs			Best among 5 runs			Best among 10 runs		
$\|V\|$	Density	B	E	W	B	E	W	B	E	W
100	0.25	40.8	47.4	11.8	55	43	2	56	44	0
	0.50	59.6	29	11.4	70	26	4	70	28	2
	0.75	67.8	23	9.2	73	21	6	74	22	4
400	0.25	35.4	41.4	23.2	41	52	7	52	42	6
	0.50	54.2	32.4	13.4	55	39	6	56	38	6
	0.75	74.4	15	10.6	87	7	6	92	4	4
1000	0.25	37	38	25	55	35	10	50	50	0
	0.50	56	28	16	70	20	10	80	10	10
	0.75	78	14	8	90	5	5	80	10	10

provides us with an evaluation of the performance quality and efficiency of our simple method in comparison with relatively more complex ones.

Our SLDN, defined by the differential equations in (5), was implemented by using the Forward–Euler algorithm. The step size was set to the minimum of $\{0.01, 1/\text{maximum degree}\}$. The initial state was chosen as a random vector with arbitrary direction in the vicinity of the origin $0 \leq x_i(0) \leq 0.01$ for all i.

Continuous Hopfield dynamics (CHD) was implemented exactly as described in [8]. Jagota suggests to bound the iteration number with the cardinality of vertex set. We observed from the experiments that the given discretization scheme for CHD may not yield convergence after $|V|$ iterations to a hypercube vertex. Thirteen such cases, which yield to nonmaximal cliques, were observed over 600 runs on large graphs with high densities (i.e., $|V| \geq 400$ and density of 0.75). To obtain a complete comparison on the whole set we allowed the CHD algorithm to continue the iterations until convergence, and we used the cardinality of the maximal clique so found.

In the simulations, we generated random graphs of 100, 400, and 1000 vertices with 0.25, 0.50, and 0.75 densities. The test sets of 100- and 400-vertex graphs include 50 instances, while the 1000-vertex graph sets have ten instances for each density. As the SLDN and CHD algorithms are run ten times on each instance, our comparisons report on a total of 3300 solutions for each method. The scheme used to generate random graphs may be summarized as follows. Initially take E as empty set. Then, for all $i, j \in V$ with $i < j$ include an edge to E with a probability equal to the density.

Table I summarizes the simulation results as giving the average clique sizes found by SLDN and CHD. For every size

TABLE III
DISTRIBUTION OF SOLUTIONS FOUND BY SLDN AND CHD FOR 0.25 DENSITY RANDOM GRAPHS

			Clique Size					
$\|V\|$	Trial No.	Method	3	4	5	6	7	8
100	1	SLDN		28.2	60.2	11.6		
		CHD	3.4	45.8	50.4	0.4		
	5	SLDN		3	73	24		
		CHD	2	39	58	1		
	10	SLDN			70	30		
		CHD	2	36	60	2		
400	1	SLDN	0.2	1.2	36.4	52.4	9.8	
		CHD		3.6	42.2	51.4	2.8	
	5	SLDN			1	64	35	
		CHD			11	82	7	
	10	SLDN				44	56	
		CHD			2	88	10	
1000	1	SLDN			16	52	31	1
		CHD			16	65	19	
	5	SLDN				10	85	5
		CHD				50	50	
	10	SLDN					90	10
		CHD				40	60	

TABLE IV
DISTRIBUTION OF SOLUTIONS FOUND BY SLDN AND CHD FOR 0.50 DENSITY RANDOM GRAPHS

			Clique Size								
$\|V\|$	Trial No.	Method	6	7	8	9	10	11	12	13	14
100	1	SLDN	0.6	22.8	51	23.6	2				
		CHD	13.8	46.4	27.2	12.6					
	5	SLDN		1	52	44	3				
		CHD	6	41	34	19					
	10	SLDN			46	50	4				
		CHD	6	36	38	20					
400	1	SLDN		0.2	3	25.4	50.4	19	1.8	0.1	
		CHD		1	17.2	43.8	28.4	7.2	0.2		
	5	SLDN				1	38	52	8	1	
		CHD			1	20	55	23	1		
	10	SLDN					22	62	14	2	
		CHD				10	52	36	2		
1000	1	SLDN				5	26	46	18	4	1
		CHD				17	47	30	6		
	5	SLDN						25	50	20	5
		CHD					20	50	30		
	10	SLDN						10	40	40	10
		CHD					10	40	50		

and density, SLDN finds larger maximal cliques on average than CHD. When we consider overall results, SLDN cliques are generally 6–8.5% larger. SLDN performances are fairly good on low-density graphs with 2.98 and 2.26% larger cliques than those of CHD for 400- and 1000-vertex graphs of 0.25 density. However, the cliques found are at least 7% larger in the case of 0.75 density graphs. When we consider the best results among five or ten runs for each instance, we observe that SLDN yields 6–12% larger cliques in all cases. Another

TABLE V
DISTRIBUTION OF SOLUTIONS FOUND BY SLDN AND CHD FOR 0.75 DENSITY RANDOM GRAPHS

$\|V\|$	Method & Trial No.	Clique Size															
		11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
100	SLDN 1	0.2	1.2	9.8	21	29.6	25.2	9.2	3.8								
	CHD 1	1.4	10.8	25.2	32.6	19.6	10.4										
	SLDN 5			2	16	29	33	13	7								
	CHD 5		5	21	33	25	16										
	SLDN 10			2	10	28	38	12	5								
	CHD 10		4	20	34	26	16										
400	SLDN 1							0.6	4.8	18.2	27.8	30	13.4	4.4	0.6	0.2	
	CHD 1					0.2	3	12.2	25.6	31.2	18	8.1	1.6				
	SLDN 5										7	33	38	18	3	1	
	CHD 5							2	5	27	37	22	7				
	SLDN 10										4	14	46	28	6	2	
	CHD 10									24	36	28	12				
1000	SLDN 1											9	17	37	21	15	1
	CHD 1									9	14	35	31	7	3	1	
	SLDN 5													5	50	40	5
	CHD 5											20	40	20	15	5	
	SLDN 10														30	60	10
	CHD 10											10	30	20	30	10	

interesting remark is on the improvement of the solution quality with multiple running of the algorithms. For up to ten trials, the improvement rate of the SLDN is higher than that of the CHD. This shows the ability of the SLDN to capture new directions without losing the quality of the solutions.

Table II compares SLDN with CHD from another point of view. That is the probability to find a solution of better, equal, or worse quality obtained by SLDN with respect to CHD. When the algorithms are run only once, in the less favorable case SLDN finds a better result for 35.4% of the runs and a worse result in 23.2%. In the interesting case of 1000-vertex high-density graphs, the percentage of the better and worse results are, respectively, 78% and 8%. When the trials are repeated, the results become more favorable to SLDN with, at most, 10% worse and 50–80% better quality solutions. That confirms our previous remark about the diversity of the good search directions captured by the SLDN.

Average and relative performances are frequently used as comparison tools, but the distribution of the solution quality provides more detailed statistical data about the behaviors of the methods. Tables III–V give the percentages of clique sizes found by SLDN and CHD. Distributions given in these tables present a low dispersion for the SLDN results and a good improvement rate for repeated trials.

The performance of the CHD algorithm reported here and in [8] is very similar for the graphs characterized with the same characteristics. In the two cases common to both experiments, the average clique sizes found by the CHD in [8] are, respectively, 7.44 and 9.16 for 100- and 400-vertex graphs of 0.5 density, versus 7.38 and 9.24 in our experiments. Therefore, we consider the test sets used in [8] and in our study to be *statistically equivalent*. Thus, we can compare the performance of the SLDN to those of the other methods considered in [8]. In our comparisons, we choose only the best four algorithms among nine ones (derived from five different methods) presented in [8]. Two of them, ρ annealing and mean field annealing (MFA), use an annealing scheme while the others, which are different versions of the stochastic steepest descent (SSD) algorithm, perform a randomized search. The first version of SSD starts with an empty solution set and progressively includes vertices, while the second one starts

TABLE VI
INDIRECT COMPARISONS AMONG SLDN AND DIFFERENT METHODS

$\|V\|$	ρ-annealing	MFA	$SSD(\emptyset,\|V\|)$	$SSD(V,\|V\|)$	SLDN-1	SLDN-5	SLDN-10
100	8.06	8.50	8.36	8.60	8.07	8.47	8.60
400	10.34	10.36	10.80	11.04	9.91	10.70	10.96

(a)

$\|V\|$	Density	RLN	SLDN-1	SLDN-5	SLDN-10
100	0.25	5.16	4.83	5.21	5.30
	0.50	8.48	8.07	8.47	8.60
	0.75	16.31	15.05	15.63	15.76
1000	0.50	11.10	10.93	12.05	12.50

(b)

with V as a solution and acts in the opposite sense. The performances of these algorithms, denoted as $\mathrm{SSD}(\emptyset, |V|)$ and $\mathrm{SSD}(V, |V|)$, are calculated by taking the best solution obtained in $|V|$ runs on each instance.

In Table VI(a), we show the average clique sizes found by the methods studied in [8] and by the SLDN executed one, five, and ten times for 0.5 density random graphs. After only five trials the SLDN provides cliques as large as the MFA algorithm of [8] does. When we consider the best results found within ten trials of SLDN, only the $\mathrm{SSD}(V, |V|)$ accomplishes better than our approach for 400-vertex graphs, but this is the best solution found by 400 runs of the SSD algorithm. In the case of 100-vertex graphs, SLDN attains, within ten trials, the same quality as the results provided by 100 runs of $\mathrm{SSD}(V, |V|)$.

The last algorithm that we compare with SLDN is the relaxation labeling network (RLN) of Pelillo [9]. This algorithm, which is based on a different formulation of the maximum clique problem, yields generally very good results so we include it in our comparison. However, the RLN may converge to infeasible solutions. It lacks a direct comparison because an accurate comparison on the same test set may require many more computational efforts. Therefore, we summarize by Table VI(b) the RLN's performance reported in [9], as compared to that of the SLDN for parameter sets common to both methods. Since the RLN starts always with the same initial state, it does not have the possibility of improving the solution quality by repeating trials with different initial states. Table VI(b) shows that when RLN converges to a feasible solution, it provides larger maximal cliques as compared to

TABLE VII
RATIO OF THE AVERAGE ITERATION NUMBER TO $|V|$ FOR SLDN ALGORITHM EXECUTED ON RANDOM GRAPHS

	Density		
$\lvert V\rvert$	0.25	0.50	0.75
100	3.55	2.76	1.77
400	3.60	2.70	1.61
1000	3.50	2.71	1.64

the SLDN, which is run only once. However, the best results found by SLDN in five trials are as good as those of RLN, except for the 0.75 density case, where RLN is clearly better. With the knowledge of that, the RLN's results are close to optimal and the results of the SLDN obtained by ten trials can be considered as very good approximations. It should be noted that SLDN has very good relative performance (with five and ten trials) for the case of 1000-vertex graphs with 0.5 density. Finally, we recall that RLN may converge to an infeasible solution which is not a clique, whereas SLDN always provides a maximal clique. This fact constitutes a significant advantage of the SLDN over the RLN algorithm.

Above we presented an evaluation in order to show the solution quality of SLDN. In the sequel, we will discuss some computational features of the digital computer implementation of SLDN. In fact, the results given on the iteration number are meaningful also in the case of a possible hardware implementation of SLDN as an analog electronic circuit. In order to give a rough estimation on the execution time, Table VII is formed to give the ratio of the average iteration number to the vertex set cardinality $|V|$. The average amount of computation required by SLDN implemented on a digital computer can be simply obtained from the iteration number by multiplying it with the operation number performed in each iteration step. As it can be seen from Table VII, the iteration number appears to be proportional to the vertex set size. The mentioned ratio increases as the density decreases, due to the independent set formulation of the maximum clique problem. Finally, we remark that the normalized variance of the iteration number, i.e., the variance of iteration number over squared mean for the random graph sets is found between 0.044 and 0.065 in our experiments. This normalized variance, indeed, gives an upper bound on the probability, as the iteration number for a run is no less than the double of average iteration number.

From the above discussions, it is clear that SLDN not only offers a good approximation to the maximum clique problem, but also has nice computational properties.

VI. CONCLUSION

In this paper we proposed a continuous-time dynamical system to approximate the maximum clique for undirected graphs. The maximum clique problem is related to some real life problems and is known to be NP complete. Hence, simple algorithms which yield acceptable solutions sufficiently fast are quite important for such related practical problems. The dynamical system proposed here is a gradient-based system and is constrained to the unit hypercube. We proved that for almost all initial conditions in the unit hypercube, the trajectories will converge to a vertex of the hypercube in finite time T and remain there for $t \geq T$, and any such vertex corresponds to a maximal (but not necessarily a maximum) clique. We also presented some simulation results for random graphs and compared our results with some existing methods. We note that in most of the cases, maximal clique obtained by our method may be considered as a reasonable approximation of maximum clique and is obtained in a reasonable amount of time.

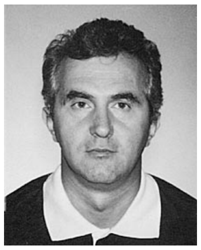
The proposed method can directly be extended to weighted maximum clique and to weighted forms of other equivalent problems. Since the model used has binary connection weights and requires only simple saturation nonlinearity, it is more suitable for VLSI implementation than some other networks applied to the same problem. The linearity of the model on all linear subregions, namely faces of the unit hypercube, gives the possibility of resorting linear analysis techniques, which might lead further improvements on the approximation to maximum clique and also might provide a useful framework in seeking solutions for some related theoretical problems.

REFERENCES

[1] M. W. Hirsch and S. Smale, *Differential Equations, Dynamical Systems, and Linear Algebra*. San Diego, CA: Academic, 1974.
[2] A. Cichocki and R. Unbehauben, *Neural Networks for Optimization and Signal Processing*. New York: Wiley, 1993.
[3] J.-H. Li, A. N. Michel, and W. Porod, "Analysis and synthesis of a class of neural networks: Linear systems operating on a closed hypercube," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1405–1422, Nov. 1989.
[4] P. M. Pardalos and F. B. Rosen, "Constrained global optimization: Algorithms and applications," in *Lecture Notes in Computer Science*. Berlin: Springer-Verlag 1987.
[5] M. R. Garey and S. J. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman 1993.
[6] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy, "Approximating clique is almost NP-complete," in *Proc. 32nd Ann. Symp. Found. Computer Science*, San Juan, Puerto Rico, 1991, pp. 2–12.
[7] P. Pardalos and J. Xue, "The maximum clique problem," *J. Global Optimization*, vol. 4, pp. 301–328, 1994.
[8] A. Jagota, "Approximating maximum clique with a Hopfield network," *IEEE Trans. Neural Networks*, vol. 6, pp. 724–735, May 1995.
[9] M. Pelillo, "Relaxation labeling networks for the maximum clique problem," *J. Artificial Neural Networks*, vol. 2, no. 4, pp. 313–328, 1995.
[10] P. M. Pardalos and G. P. Rodgers, "A branch and bound algorithm for the maximum clique problem," *Computers Ops. Res.*, vol. 19, no. 5, pp. 363–375, 1992.

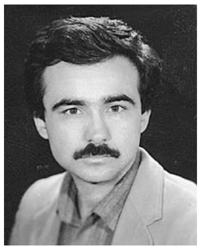
Ferhan Pekergin received the B.Sc. and M.Sc. degrees in electrical engineering from the Technical University of İstanbul, İstanbul, Turkey, in 1980 and 1982, respectively, and the Ph.D. degree in computer science from the Université René Descartes (Paris V), Paris, France, in 1992.

He was with the Technical University of İstanbul and the Université Paris-Sud, Paris, France, as a Teaching and Research Assistant. In 1992 he joined the Institut Universitaire de Technologie de Villetaneuse, the Université Paris-Nord, Paris, France, where he is currently Maître de Conférences. His research interest are in the areas of modeling and performance evaluation of communication networks, approximation techniques, neural networks, and combinatorial optimizations.



Ömer Morgül was born in İstanbul, Turkey, in 1959. He received the B.Sc. and the M.Sc. degrees in electrical engineering from the Technical University of Istanbul, Turkey, in 1980 and 1982, respectively, and the Ph.D. degree in electrical engineering from the University of California, Berkeley, in 1989.

Since September 1989, he has been with the Department of Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey, where he is now an Associate Professor. His research interests are in the area of systems and control theory, including the boundary control of flexible systems, nonlinear systems, chaotic electronic circuits, and neural networks.



Cüneyt Güzeliş received the B.Sc., M.Sc., and Ph.D. degrees from İstanbul Technical University, Istanbul, Turkey, in 1981, 1984, and 1989, respectively.

In 1982 he joined İstanbul Technical University as an assistant and is now a Professor. He was a Visiting Researcher and Lecturer in the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley from April 1989 to April 1991, and was a Visiting Professor in the Information Laboratory, University of Paris-Nord in September 1996 and June 1997. His research interests include nonlinear circuits and systems and neural networks and their signal processing applications.