Annals of Operations Research 48(1994)433–461

G-networks: a unifying model for neural and queueing networks

Erol Gelenbe
Department of Electrical Engineering, Duke University, Durham, NC 27706, USA

We survey results concerning a new stochastic network we have developed [1–7], which was initially motivated by neural network modelling [1], or – as we called it – by queueing networks with positive and negative customers [2, 3]. Indeed, it is well known that signals in neural networks are formed by impulses or action potentials, traveling much like customers in a queueing network. We call this model a G-network because it serves as a unifying basis for diverse areas of stochastic modelling in queueing networks, computer networks, computer system performance and neural networks. In its simplest version, "negative" and "positive" signals or customers circulate among a finite set of units, modelling inhibitory and excitatory signals of a neural network, or "negative and positive customers" of a queueing network. Signals can arrive either from other units or from the outside world. Positive signals are accumulated at the input of each unit, and constitute its signal potential. The state of each unit or neuron is its signal potential (which is equivalent to the queue length), while the network state is the vector of signal potentials at each neuron. If its potential is positive, a unit or neuron fires, and sends out signals to the other neurons or to the outside world. As it does so, its signal potential is depleted. In the Markovian case, this model has product form, i.e. the steady-state probability distribution of its potential vector is the product of the marginal probabilities of the potential at each neuron. The signal flow equations of the network, which describe the rate at which positive or negative signals arrive to each neuron, are non-linear. We discuss the relationship between this model and the usual connectionist (formal) model of neural networks, and present applications to combinatorial optimization and to image texture processing. Extensions of the model to the case of "multiple signal classes", and to "networks with triggered customer motion" are presented. We also examine the general stability conditions which guarantee that the network has a well-defined steady-state behaviour.

1. Introduction

In this introduction, we deal with the simplest instance of the G-network model. The presentation begins with terminology from neural networks. The queueing network analogy will become apparent as we proceed.

Consider a network of n neurons in which *positive and negative* signals circulate. Each neuron accumulates signals as they arrive, and can fire if its total

signal count at a given instant of time is positive. Firing then occurs at random according to an exponential distribution of constant rate, the neuron sends signals out to other neurons or to the outside of the network.

Positive and negative signals have different roles in the network; positive signals represent excitation, while negative signals represent inhibition. A negative signal *reduces by 1* the potential of the neuron to which it arrives (i.e. it "cancels" an existing signal) or has no effect on the signal potential if it is already zero, while an arriving positive signal *adds 1* to the neuron potential. The potential at a neuron is constituted only by positive signals which have accumulated, which have not yet been cancelled by negative signals, and which have not yet been sent out by the neuron as it fires.

Figure 1 is a simplified representation of biophysical neural behaviour (see for instance [9, 10, 12]). In this figure, at time $t = 0$ a neuron is excited; at time T (typically T may be of the order of 50 milliseconds) it fires a train of impulses along its axone. Each of these impulses is practically of identical amplitude (represented in our random model by 1). Some time later (say around $t = T + \tau$), the neuron may fire another train of impulses, as a result of the same excitation. Even when the neuron is not excited, it may send out impulses at random. Our model represents firing of an excited neuron by a train of impulses or unit signals; the $+1$ or -1 value is tied to the interpretation (excitation or inhibition) given to the arriving signal at the receiving neuron. However, the fact that all neural impulses or "action potentials" have a quasi-constant amplitude is well known.

1.1. THE MATHEMATICAL MODEL

Let us now turn to the mathematical model. Signals can either arrive to a neuron from the outside of the network (exogenous signals) or from other neurons. Each time a neuron fires, a signal leaves it, depleting the total input potential of the neuron. A signal which leaves neuron i heads for neuron j with probability $p^+(i, j)$ as a positive (or normal) signal, or as a negative signal with probability $p^-(i, j)$, or it departs from the network with probability $d(i)$. Let $p(i, j) = p^+(i, j) + p^-(i, j)$; it is the transition probability of a Markov chain

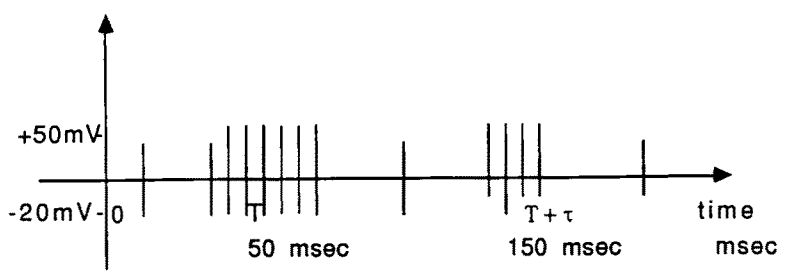

Fig. 1. Representation of biophysical neuron output signal after excitation at time $t = 0$.

representing the movement of signals between neurons. Clearly, we have $\sum_j p(i,j) + d(i) = 1$ for $1 \leq i \leq n$.

A neuron is capable of firing and emitting signals if its potential is strictly positive. We assume that exogenous signals arrive to each neuron in Poisson streams of positive or negative signals. This is equivalent to a network of n queues, in which positive and negative customers circulate. Positive customers correspond to the usual customers of queueing systems, while a negative customer eliminates a positive customer at the queue to which it arrives. Queue length corresponds to a neuron's signal potential. A physical analogy is also to consider matter and anti-matter.

In [1, 2] we show that the purely Markovian version of this network, with positive signals which arrive to the ith neuron according to a Poisson process of rate $\Lambda(i)$, negative customers which arrive to the ith neuron according to a Poisson process of rate $\lambda(i)$, i.i.d. exponential neuron firing times with rates $r(1), \ldots, r(n)$, and Markovian movements of signals between neurons, has a product form solution. That is, the open network's stationary probability distribution can be written as the product of the marginal probabilities of the state of each neuron. Thus, *in steady-state the network's neurons are seemingly independent, though they are in fact coupled* via the signals which move from one neuron to the other in the network. On the other hand, as with previous models of neural networks [9], the equations which yield the rate of signal arrival and hence the rate of firing of each neuron are non-linear. This distinguishes the mathematical model from previously known queueing networks.

1.1.1. A model for the study of artificial neural networks

The model we propose has a certain number of interesting features as a tool for representing artificial neural networks:

- it is easy to simulate, since each neuron is simply represented by a counter;
- it is closely related to the standard connexionist model [9] and it is possible to go from one model to the other easily;
- it represents neuron potential and therefore the level of excitation as an integer, rather than as a binary variable, which leads to more detailed information on system state.

1.1.2. Interest of G-networks as an extension of queueing networks

Negative customers can be viewed as representing a decision to remove work from a queue. A typical application would be in packet switching networks where positive customers model ordinary packets, while negative customers model acknowledgement packets. A queueing network with negative and positive customers can also be viewed as a network of communicating semaphores in a

parallel operating system, where each queue represents a semaphore. G-networks with triggered customer movement [7], discussed in section 7, can be used to represent computer networks in which routing decisions occur dynamically.

1.1.3. Further extensions of G-networks

Starting from the model with negative and positive customers we introduced in [1, 2], several extensions have been considered. In [3], single server networks with renewal arrivals and various service rules have been studied. In [4], we have discussed a multiple class extension of these models which is reviewed in section 6 of the present paper. In [5], the general stability conditions for these networks, including multiple classes, have been examined. In [6], we have introduced the case where negative customers can carry out batch customer removal; this was independently obtained by Henderson et al. [14] with the assumption that batch sizes depend both on the origin of the negative customer and on the queue at which removal takes place. In [7], the concept of triggered customer movement in G-networks is introduced: here, negative customers are allowed to displace normal customers from one queue to another, and in section 7 we present these results with the possibility of batch customer removal. In [18], the model has been extended to the case of "negative and positive neurons" and applied to associate memory: here queues (or neurons) are allowed to have negative queue length.

1.2. PRODUCT FORM SOLUTION OF G-NETWORKS

Each neuron i of the network is represented at any time t by its input signal potential $k_i(t)$. This is equivalent to the ith queue's length in a queueing network. The model's most important property is that the steady-state probability distribution of network state can always be expressed as the product of the probabilities of the states of each neuron. Thus, in steady-state, the network is seemingly composed of independent neurons, though this is obviously not the case. Let $k(t)$ be the vector of signal potentials at time t, and $k = (k_1, \ldots, k_n)$ be a particular value of the vector. We are obviously interested in the quantity $p(k, t) = \text{Prob}\,[k(t) = k]$, and more particularly in $p(k) = \lim_{t\to\infty} \text{Prob}\,[k(t) = k]$ which denotes the stationary probability distribution, when it exists.

THEOREM 1 [1, 2]

Let

$$q_i \equiv \lambda^+(i)/[r(i) + \lambda^-(i)], \tag{1}$$

where the $\lambda^+(i)$, $\lambda^-(i)$ for $i = 1, \ldots, n$ satisfy the following system of non-linear

simultaneous equations:

$$\lambda^+(i) = \sum_j q_j r(j) p^+(j,i) + \Lambda(i), \quad \lambda^-(i) = \sum_j q_j r(j) p^-(j,i) + \lambda(i). \tag{2}$$

If a unique non-negative solution $\{\lambda^+(i), \lambda^-(i)\}$ exists to eqs. (1), (2) such that each $q_i < 1$, then:

$$p(k) = \prod_{i=1}^{n} [1 - q_i] q_i^{k_i}. \tag{3}$$

The proof can be found in [1]. Since $\{k(t) : t \geq 0\}$ is a continuous time Markov chain, it satisfies the usual Chapman–Kolmogorov equations; thus in steady-state $p(k)$ must satisfy the following global balance equations:

$$\begin{aligned} p(k) \sum_i & [\Lambda(i) + (\lambda(i) + r(i))\mathbf{1}[k_i > 0]] \\ &= \sum_i \Bigg[p(k_i^+) r(i) d(i) + p(k_i^-) \Lambda(i) \mathbf{1}[k_i > 0] + p(k_i^+) \lambda(i) \\ &\quad + \sum_j \{ p(k_{ij}^{+-}) r(i) p^+(i,j) \mathbf{1}[k_j > 0] + p(k_{ij}^{++}) r(i) p^-(i,j) \\ &\quad + p(k_i^+) r(i) p^-(i,j) \mathbf{1}[k_j = 0] \} \Bigg], \end{aligned} \tag{4}$$

where the vectors used in (4) are defined as follows:

$$\begin{aligned} k_i^+ &= (k_1, \ldots, k_i + 1, \ldots, k_n), \\ k_i^- &= (k_1, \ldots, k_i - 1, \ldots, k_n), \\ k_{ij}^{+-} &= (k_1, \ldots, k_i + 1, \ldots, k_j - 1, \ldots, k_n), \\ k_{ij}^{++} &= (k_1, \ldots, k_i + 1, \ldots, k_j + 1, \ldots, k_n). \end{aligned}$$

Here $\mathbf{1}[X]$ is the usual characteristic function which takes the value 1 if X is true and 0 otherwise. Theorem 1 is proved by showing that (3) satisfies this system of equations. The computational simplicity of this result is illustrated by the useful consequence indicated below.

COROLLARY 1.1

The probability that neuron i is firing in steady-state is simply given by q_i and the average neuron potential in steady-state is simply $A_i = q_i/[1 - q_i]$.

By theorem 1 we are guaranteed a stationary solution of product form provided the non-linear signal flow eqs. (1), (2) have a non-negative solution. The following result can be easily established.

THEOREM 2

If the solution to (1), (2) exists with $0 < q_i < 1$, then it is unique.

Proof

Since $\{k(t) : t \geq 0\}$ is an irreducible and aperiodic Markov chain, if a positive stationary solution $p(k)$ exists, then it is unique. By theorem 1, if the $0 < q_i < 1$ solution to (1), (2) exists for $i = 1, \ldots, n$, then $p(k)$ is given by (3) and is clearly positive for all k. Suppose now that for some i there are two different q_i, q_i' satisfying (1), (2). But this implies that for all k_i, $\lim_{t\to\infty} P[k_i(t) = 0]$ has two different values $[1 - q_i]$ and $[1 - q_i']$, which contradicts the uniqueness of $p(k)$; hence the result. □

Let us now turn to the existence and uniqueness of the solutions $\lambda^+(i), \lambda^-(i)$, $1 \leq i \leq n$, to eqs. (1), (2) which represent the average arrival rate of positive and negative signals to each neuron in the network.

We shall say that a network is *damped* if $p^+(j,i) \geq 0$, $p^-(j,i) \geq 0$ with the following property: $r(i) + \lambda(i) > \Lambda(i) + \sum_j r(j)p^+(j,i)$ for all $i = 1, \ldots, n$. The following result provides sufficient conditions for stability which are in general easy to verify. Necessary and sufficient conditions will be discussed at the end of this paper.

THEOREM 3 [2]

If the network is damped, then the customer flow equations (1), (2) always have a unique solution with $q_i < 1$.

Proof

The proof is by construction of the n-dimensional vector homotopy [9] function $H(q, x)$ for a real number $0 \leq x < 1$. Let us define the following n-vectors:

$$q = (q_1, \ldots, q_n), \quad F(q) = (F_1(q), \ldots, F_n(q)),$$

where

$$F_i(q) = \left[\sum_j q_j r(j) p^+(j,i) + \Lambda(i)\right] \Bigg/ \left[\sum_j q_j r(j) p^-(j,i) + \lambda(i) + r(i)\right].$$

Clearly, the equation we are interested in is $q = F(q)$ which, when it has a solution in $D = [0,1]^n$, yields the appropriate values of the q_i for theorem 1 to be applicable in order to compute the stationary solution of the network. Notice that $F(q) : \mathbb{R}^n \to \mathbb{R}^n$. Notice also that $F(q) \in C^2$. Consider the mapping $F(q) : D \to \mathbb{R}^n$. Clearly, we are interested in the interior points of D since we seek solutions $0 < q_i < 1$. Write $D = D^0 \cup \partial D$ where ∂D stands for the boundary of D, and D^0 is the set of interior points. Let $y = (y_1, \ldots, y_n)$ where

$$y_i = \left[\sum_j r(j)p^+(j,i) + \Lambda(i)\right] \Big/ [\lambda(i) + r(i)].$$

By assumption, $y_i < 1$ for all $i = 1, \ldots, n$. Now define

$$H(q,x) = (1-x)(q-y) + x(q - F(q)), \quad 0 \leq x < 1.$$

Clearly $H(q,0) = q - y$ and $H(q,1) = q - F(q)$. Consider the set

$$H^{-1} = \{q : q \in D, H(q,x) = 0 \text{ and } 0 \leq x < 1\}.$$

We can show that H^{-1} and ∂D have an empty intersection, i.e. as x is varied from 0 towards 1 the solution of $H(q,x)$, if it exists, does not touch the boundary of D. To do this assume the contrary; this implies that for some $x = x^*$ there exists some $q = q^*$ for which $H(q^*, x^*) = 0$ and such that $q_i^* = 0$ or 1. If $q_i^* = 0$ we can write

$$-(1-x^*)y_i - x^* F_i(q^*) = 0, \quad \text{or} \quad x^*/(1-x^*) = -y_i/F_i(q^*) < 0 \quad \Rightarrow \quad x^* < 0,$$

which contradicts the assumption about x. If $q_i^* = 1$, then

$$-(1-x^*)(1-y_i) - x^*(1 - F_i(q^*)) = 0, \quad \text{or}$$

$$x^*/(1-x^*) = -(1-y_i)/(1 - F_i(q^*)) < 0 \quad \Rightarrow \quad x^* < 0,$$

because $(1-y_i) > 0$ and $0 < F_i(q^*) < y_i$ so that $(1 - F_i(q^*)) > 0$, again contradicting the assumption about x. Thus $H(q,x) = 0$ cannot have a solution on the boundary ∂D for any $0 \leq x < 1$. As a consequence, applying theorem 3.3.1 of [9] (which is a Leray–Schauder form of the fixed-point theorem), it follows that $F(q) = q$ has at least one solution in D^0; it is therefore a unique solution as a consequence of theorem 2.

2. Analogy with classical connexionist neural networks

A connexionist network [9] is a set of n deterministic neurons each of which computes its *state* $y(i)$ using a sigmoid function $y(i) = f(x(i))$ where

$x(i) = (\sum_j w_{ji} y(j) - \theta_i)$ is the *input signal* composed of the weighted sums of the states $y(j)$ of the other neurons of the network; the w_{ji} are the weights and θ_i is the threshold. In its simplest form $f(\ )$ is the unit step function. The set of weights and the set of thresholds completely characterise the network.

An analogy between this usual model of neural networks and the model discussed in this paper can be easily constructed. Each deterministic neuron is replaced by a neuron of the G-network. The threshold of neuron i is represented by a flow of negative signals to the neuron so that $\lambda(i) = \theta_i$.

Consider the non-output neuron i; it is represented by the random neuron i whose parameters are chosen as follows:

$$d(i) = 0, \quad r(i)p^+(i,j) = w_{ij} \text{ if } w_{ij} > 0 \text{ and } r(i)p^-(i,j) = |w_{ij}| \text{ if } w_{ij} < 0.$$

Summing over all j, the firing rate $r(i)$ of the non-output "random" neuron i is chosen:

$$r(i) = \sum_j |w_{ij}|.$$

Finally, for each output random neuron $id(i) = 1$, and assign some appropriate value to $r(i)$. To introduce the external parameters or inputs to the neural network we use the arrival rates of positive signals $\Lambda(i)$. Assume that the external signals to the formal neural network are binary. If the input signal to neuron i is 0 or if neuron i is not connected to an external signal we, set $\Lambda(i) = 0$. If the external signal to neuron i has the value 1, we set $\Lambda(i) = \Lambda$. Here Λ can be chosen so as to obtain the desired effect at the output neurons.

All the parameters of the random network are chosen from those of the formal network, except for the firing rates of the output neurons, and the input rate of positive signals. The state $Y = (y_1, \ldots, y_n)$ of the formal neural network, where y_i is 0 or 1, is simulated by the probabilities of the random neurons. Consider $\lim_{t\to\infty} P[k_i(t) > 0]$ which (as a consequence of corollary 1.1) is simply q_i; we associate y_i to q_i. Thus we have for some "cut-point" $1 - \alpha$,

$$[y_i = 0] \Leftrightarrow q_i < 1 - \alpha; \quad [y_i = 1] \Leftrightarrow q_i \geq 1 - \alpha.$$

A more detailed way of differentiating between highly excited and relatively unexcited neurons is via the average neuron potential $A_i = q_i/[1 - q_i]$. In an arbitrary neural network, i.e. one which does not have the feedforward structure, this procedure could also be used for establishing the random network. We could also use the $d(i)$ at each neuron i in order to represent the loss currents which are known to exist in biological neural systems [10], and take $r(i)d(i)$ to be the rate of loss of electric potential at a neuron if we wish to include this effect in the model.

EXAMPLE 1

A simple example often given to illustrate the behaviour of formal neural networks is the network for the XOR (exclusive OR) function [9]. We will present the equivalent random model representation for this network. In fig. 2 we show a formal neural network which receives two binary inputs x_1, x_2 and which produces $y(x_1, x_2)$, the Boolean function XOR. It is composed of four neurons; each neuron is numbered (1 to 4) and the synaptic weights are indicated on arcs between neurons. The threshold of each neuron is 0.

In fig. 3 we show the G-network analog corresponding to fig. 2. According to the rules we have given for constructing the random analog, we have:

- $\lambda(i) = 0$ for $i = 1, \ldots, 4$ because all thresholds are 0;
- $r(1) = r(2) = 2$, $r(3) = 1.1$, $r(4) = r$, as yet undetermined;
- $p^+(1,3) = p^+(2,3) = 0.5$, $p^-(1,4) = p^-(2,4) = 0.5$, $p^+(3,4) = 1$;
- $d(1) = d(2) = d(3) = 0$, $d(4) = 1$.

Recall that according to the rules proposed above, we choose a value $L(i) = \Lambda$ to represent $x_i = 1$, and $\Lambda(i) = 0$ to represent $x_i = 0$. Set Λ large enough to saturate neurons 1 and 2, i.e. any $\Lambda > 2$. q_4 is the analog of the output y of the

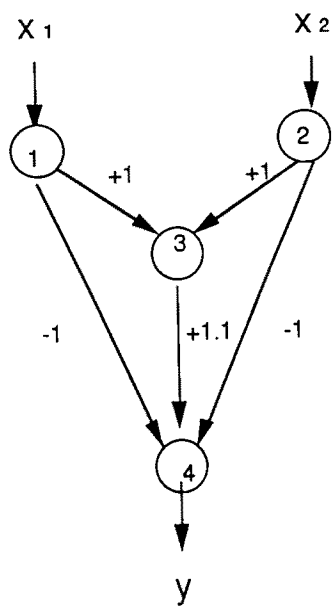

Fig. 2. A connexionist network for the Boolean XOR function.

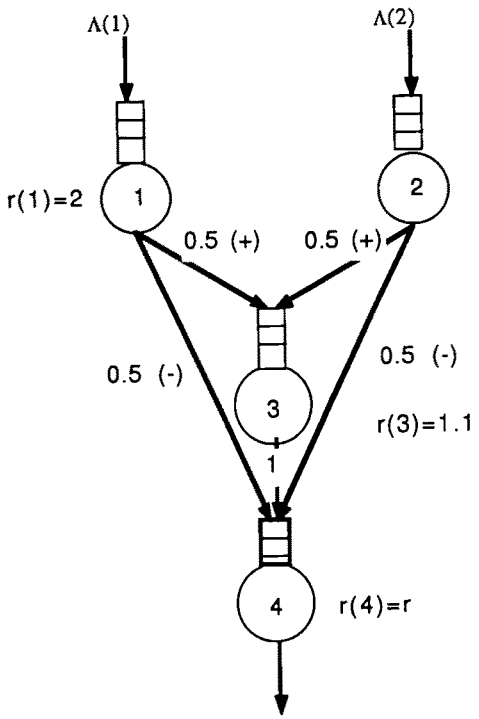

Fig. 3. The G-network analog of the formal neural network shown in fig. 2; here $p^+(1,3) = p^+(2,3) = 0.5$, $p^+(3,4) = 1$, and $p^-(1,4) = p^-(2,4) = 0.5$.

connexionist network of fig. 2. Notice that:

$$q_4 = \begin{cases} 0, & \text{if } \Lambda(1) = L(2) = 0, \\ 1.1/[r+2], & \text{if } \Lambda(1) = \Lambda(2) = \Lambda > 2, \\ 1/[r+1], & \text{if } \Lambda(1) = \Lambda,\ \Lambda(2) = 0 \text{ or vice-versa.} \end{cases}$$

Setting $\alpha = 0.85$ and $r = 0.1$ we see that we obtain the XOR function with this network, since we have $q_4 = 0.909$ when $\Lambda(1) = \Lambda$, $\Lambda(2) = 0$ or vice-versa, and $q_4 = 0.5238$ if $\Lambda(1) = \Lambda(2) = \Lambda$ for any $\Lambda > 2$. In fact we may choose any $1 - \alpha$ such that $1.1/[r+2] < 1 - \alpha < 1/[r+1]$. The capacity of the network to represent the XOR function becomes even more apparent if we consider the average potential of neuron 4; we then have $A_4 = 11.222$ when $\Lambda(1) = 2$, $\Lambda(2) = 0$ or vice-versa, and $A_4 = 1.0999$ if $\Lambda(1) = \Lambda(2) = \Lambda$ for any $\Lambda > 2$. Thus, the average neuron potential can also be used very effectively to discriminate between the output states of the neural network since we have a ratio greater than 10 between the average potential of neuron 4 of the input equivalent to (0,1) or (1,0) and the input equivalent to (1,1).

3. A simple hardware implementation of G-networks

In addition to being an analytical tool for the study of artificial neural networks, G-networks can also be implemented in hardware, or simulated in software, as shown in fig. 4.

Here, each neuron is simulated using a Counter and a Random Number Generator (RNG). The latter is used to simulate the exponential delays associated with the signal emission process, as well as to handle the signal routing probabilities. The interconnection network receives signals, which can be viewed as short packets of data containing the signal's polarity and its destination, and routes the signal to the appropriate output line connected to the neuron to which it is addressed. Alternatively, all routing information and signal polarity (including the random generation of the destination) can be handled by the interconnection network. This network can be programmable so as to represent different networks. External signal sources can be implemented simply as permanently activated counters which constantly generate signals as their output. Obviously, as with any highly parallel machine implementation, the complexity of the system is concealed in the interconnection network.

4. Use of the G-network model for a neural network approach to combinatorial optimization

Combinatorial problems arise in a routine manner in many applications, and many of the most applicable problems are NP-hard so that efficient algorithms for

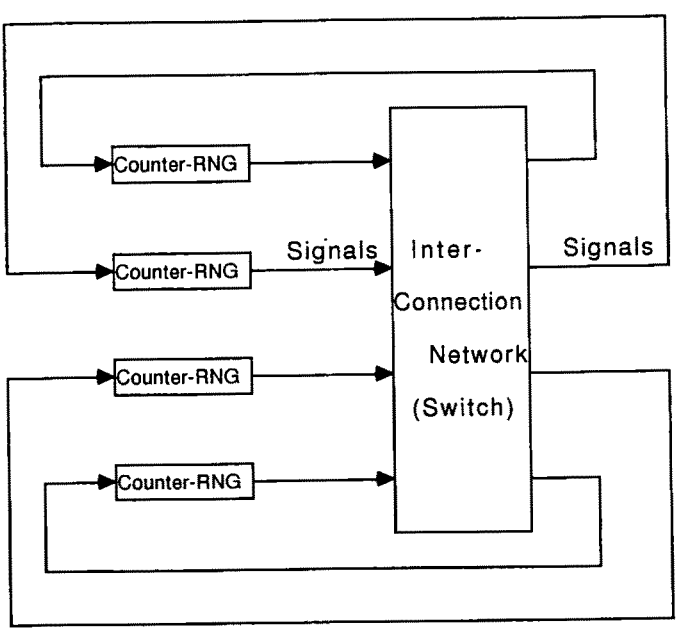

Fig. 4. A possible hardware implementation of a random network.

their exact solution are computationally intractable when the problem size becomes realistically large. Artificial neural networks have been proposed in the past as tools for providing heuristic solutions to such problems [13] and this is still an active area of research. One of the major issues involved is to maintain the computational cost of the neural network heuristic at an acceptably low level; this is difficult to do with "classical" neural network models, since the solution method used for the neural network heuristic is in general a Monte Carlo simulation. The G-network provides an alternative to this approach since it can use an analytical solution [8].

The purpose of this section is to illustrate the application of the G-network to the heuristic solution of an NP-hard problem: the "minimum graph covering problem" which appears in diverse applications such as plant lay-out or image processing.

The formal problem can be stated as follows. Let G be a graph with K nodes $\mathcal{K} = \{1, \ldots, K\}$ and edges denoted by (u, v), where u, v are nodes of the graph. A *cover* of G is a subset $\mathcal{C}$ of $\mathcal{K}$ such that for each edge (u, v) in G, either u or v are in $\mathcal{C}$. A *minimum cover* of G is a set $\mathcal{C}^*$ such that the number of nodes in $\mathcal{C}^*$ is no larger than the number of nodes in any cover $\mathcal{C}$ of G: $|\mathcal{C}^*| \leq |\mathcal{C}|$.

In intuitive terms, if an image is represented by a set of interconnected straight lines, then the minimum cover will yield a compact representation of the image with a minimum number of components. In the transposition of the image to the graph representation, the straight lines of the image can be represented by vertices of G, while lines intersect at an angle would give rise to an edge in G.

This problem can also be formulated so that each node of the graph G can carry a weight, and this weight is included in the "size" of each cover $\mathcal{C}$. We summarize and compare here the solution of the minimum graph covering problem using four main approaches:

- The exact solution E; in this case all covers are enumerated and a minimum cover is selected. For graphs chosen at random, we have rapidly discovered that this approach was too time consuming to implement on a workstation when the number of nodes of G exceeded K.
- A simulated "neural network" based on simulated annealing (SA) [16].
- The heuristic solution, using a well-known approach, the Greedy Algorithm GA. The basic idea here is to first select the node of G which has the highest degree (number of neighbours), and to include it in $\mathcal{C}^+$, the cover being generated; the node and all its adjacent edges and terminal nodes are then removed from the graph and the procedure is repeated until all nodes have been removed, or placed in $\mathcal{C}^+$.
- Finally, the heuristic solution using the G-network, based on our work [8]. This is discussed below.

4.1. G-NETWORK CONSTRUCTION FOR THE GRAPH COVERING PROBLEM [8]

The network used to solve the problem is set up in the following manner. For each node i of G, we construct two neurons $N(i)$ and $n(i)$. The objective is to have $N(i) = 1$ and $n(i) = 0$ if $i \in \mathscr{C}^*$, and $N(i) = 0$ and $n(i) = 1$ in the opposite case. The network parameters are chosen as follows:

$\Lambda(N(i)) = D(i)$, the degree of node i,

$\Lambda(n(i)) = 1$ for all i,

$r(N(i)) = 2K$,

$r(n(i)) = D(i)$,

$p^-(N(i), n(i)) = 1$ for all i,

$p^+(n(i), N(j)) = 1/D(i)$ if $j \in \mathscr{N}(i)$,

where we denote by $\mathscr{N}(i)$ the set of neighbours of node i. Notice that this network is inspired from the greedy algorithm heuristic, since each $N(i)$ is excited by an external signal which is proportional to $D(i)$. However, the effect of the greedy algorithm is compensated by the role of the complementary neurons $n(i)$ which are inhibited by the corresponding $N(i)$, but which in turn excite all of the $N(j)$ such that node j is a neighbour of node i. The purpose of this is to increase the probability of including in the minimal cover the neighbours of a node which is not being included in the minimal cover. There are no exogenous negative signals, i.e. $\lambda(N(i)) = \lambda(n(i)) = 0$ for all i.

It can be easily seen that, because we have chosen $r(N(i)) = 2K$, the excitation probability of node $N(i)$ will satisfy $q(N(i)) \leq 1$. Similarly, the excitation probability of $n(i)$ will satisfy $q(n(i)) \leq 1/(q(N(i)) + D(i)) \leq 1$ for any node i which is not isolated (i.e. which has at least one neighbour, since otherwise it does not need to be considered in the cover). So the solution of eq. (1) exists. It is obtained in practice by an iterative numerical solution of the equations:

$$q(N(i)) = \left[D(i) + \sum_{j \in \mathscr{N}(i)} q(n(j)) \right] / 2K,$$

$$q(n(i)) = 1/[D(i) + 2Kq(N(i))].$$

In fact, we need only solve the K equations (one per node) of the form:

$$q(n(i)) = 1 \Big/ \left[2D(i) + \sum_{j \in \mathscr{N}(i)} q(n(j)) \right].$$

Table 1

	$\pi = 0.5$			$\pi = 0.25$			$\pi = 0.125$		
$K = 20$ Method	mini [%]	exc	T [s]	mini [%]	exc	T [s]	mini [%]	exc	T [s]
SA	100	0	43	88	0.12	35	96	0.04	27
GR	100	0	0	92	0.08	0	100	0	0
RN	100	0	3	96	0.04	0	100	0	0

The algorithm we have used for obtaining a "quasi-minimal" cover $\mathscr{C}^-$ is:

Step 1. Solve for the $q(N(i))$, $q(n(i))$, $i = 1, \ldots, K$ from the above model.

Step 2. Place in $\mathscr{C}^-$ the node, say i^*, for which $q(N(i))$ is largest, i.e. closest to 1.

Step 3. Remove i^* and all its neighbours from G, and reinitialize the above model without i^*.

Step 4. Return to step 1 if there is more than one node in the remaining graph; otherwise include it into $\mathscr{C}^-$ and end the procedure.

Notice that step 1 of the algorithm, which is the most time-consuming part, can be accelerated and executed in parallel with up to $2K$ processors.

4.2. EXPERIMENTAL COMPARISON

Let us now compare results obtained with the three methods described above, using randomly generated graphs. We compare both the quality of the results obtained, and the total computation time of the method using the same computer workstation.

A family of random graphs is defined by two parameters (K, π), where K is the number of nodes and π is the probability that there is an arc between any two nodes of the graph. For $K = 20$ and 50 (see tables 1, 2) we tabulate results and indicate: the fraction of cases where the method found the minimum cover (**mini**), the average number of nodes in excess of the minimum cover (**exc**), and the

Table 2

	$\pi = 0.5$			$\pi = 0.25$			$\pi = 0.125$		
$K = 50$ Method	mini [%]	exc	T [s]	mini [%]	exc	T [s]	mini [%]	exc	T [s]
SA	72	0.28	43	56	0.60	1.25	56	0.48	90
GR	56	0.48	0	48	0.56	0	76	0.24	0
RN	52	0.52	32	60	0.4	20	80	0.2	9

Table 3

	$\pi = 0.5$			$\pi = 0.25$			$\pi = 0.125$		
$K = 100$ Method	mini [%]	exc	T [s]	mini [%]	exc	T [s]	mini [%]	exc	T [s]
SA	68	0.36	600	52	0.72	140	32	1.32	260
GR	52	0.56	3	56	0.64	1	56	0.64	1
RN	60	0.52	220	48	0.56	120	56	0.52	70

computation time T. For $K = 100$ (see table 3), the exhaustive search of the minimum cover for each graph was impossible to carry out. Thus for $K = 100$, **exc** stands for the average number of nodes in excess of the smallest cover found among the four methods tested. For each (K, π), the number of graphs chosen at random is 25, and the values shown are average values over this set.

We notice that for large π, Simulated Annealing (SA) provides better results; however, the computational cost is generally incompatible with "fast" or real-time applications. For smaller values of π, which correspond to more realistic image processing applications since the graphs considered are less dense, the modified Greedy Algorithm (GR) and the Random Network (RN) algorithm provide substantially better results than SA. GR is the lowest cost algorithm in terms of computation time, but is slightly less effective than the RN algorithm.

At this point of the evaluation, we dispose of a reasonable comparison of these various methods. It shows that the approximate solution of combinatorial optimization problems based on the G-network is a good direction to explore.

5. G-networks for image texture generation [17]

The generation of artificial textures is a useful function in image synthesis systems. The purpose of this section is to describe a novel application of G-networks [17]: the generation of various image textures having different characteristics. The generation process is analytical and the textures which are obtained have features similar to those generated by the MRF model such as granularity and inclination. An eight parameter model, based on choosing the local interaction parameters between neighbouring neurons in the plane, is proposed. Numerical iterations of the "traffic" or field equations of the neural network model, starting with a randomly generated grey-level image, are shown to produce textures having different desirable features. The experimental evaluation shows that the random network provides good results, at a computational cost substantially less than that of other approaches such as Markov random fields. Modelling textures with Markov random fields (MRF) has been of a great interest in the last decade [20, 21]. Many natural textures which come up in applications such as satellite image

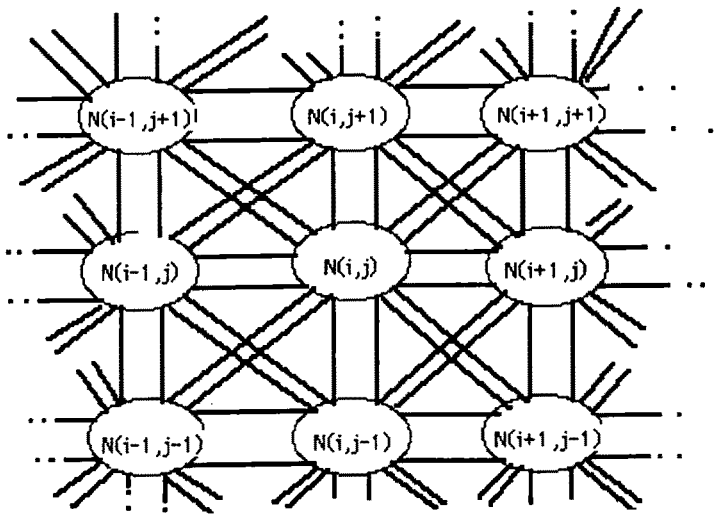

Fig. 5. Topology of the random neural network used for texture generation.

data analysis or biomedical image analysis may be described by these models, and they are known to be useful in classification and segmentation problems. However, in general, the operations performed during texture generation are all very time consuming because they are essentially based on Monte Carlo simulations, which may be prohibitively long in many real applications. Thus there have been attempts to parallelize these algorithms [22] so as to reduce the computational cost. However, parallel MRF algorithms still require a substantial amount of computer time.

The G-network which we propose in order to generate artificial textures associates a neuron $N(i, j)$ to each point or picture element ("pixel") (i, j) in the plane. The state $f(i, j)$ of $N(i, j)$ can be interpreted as the grey-level value of the pixel at (i, j). The topology of the random network which we propose for texture generation is shown in fig. 5. We restrict our attention to the generation of grey-level images. We see that each neuron in the network will be, in general, connected to at most 8 neighbours. In many instances, it will suffice that each neuron be in fact connected to as few as 4 other neurons.

We use the notation of fig. 6 (with numbering of neurons from 0 to 7 in counterclockwise order around a given neuron) for the positions in the network, where x denotes any (i, j).

x_3	x_2	x_1
x_4	x	x_0
x_5	x_6	x_7

Fig. 6. Simplified notation for neuron positions.

For instance, x_0 denotes $(i+1, j)$. The parameters of the G-network will be chosen as follows:

(a) $p^+(x_d, x) = p^+(x, x_d) = P_d$ and $p^-(x_d, x) = p^-(x, x_d) = Q_d$ for any $d = 0, \ldots, 7$; i.e. all connections are symmetrical, be they excitatory or inhibitory. In general either P_d or Q_d is non-zero, but not both; i.e. between any two neurons the connection is either excitatory or inhibitory but not both.

(b) $r(x) = r$, $\Lambda(x) = c$, where r and c are constants, and $\lambda(x) = 0$, for any neuron $N(x)$ in the network.

(c) To simplify notation, set $w_d = rP_d w_d = -rQ_d$. In order to obtain directional symmetry we set $w_0 = w_4$, $w_2 = w_6$, $w_3 = w_7$, $w_1 = w_5$.

Notice that $r = \sum_d q(x_d)|w_d|$. Using eqs. (1), (2) we have:

$$\lambda^+(x) = \sum_d q(x_d) r P_d + c, \quad \lambda^-(x) = \sum_d q(x_d) r Q_d,$$

where $q(x_d)$ denotes the stationary probability of the state of neuron x_d, and for any neuron in position or pixel x,

$$q(x) = \lambda^+(x)/(\lambda^-(x) + r).$$

We associate the grey-value function $f(x)$ of a pixel x, with the q values:

$$f(i, j) = f(x) = q(x) \text{ if } q(x) \leq 1 \quad \text{and} \quad f(x) = 1 \text{ if } q(x) > 1.$$

To deal with binary pictures, we simply set $f(x) = 1$ if $q(x) > 0.5$, and $f(x) = 0$ otherwise.

Consider now the choice of the parameter c; a large value of c will lead to a brighter image, while a small value of c will lead to a darker image. The choice can be made as follows. Because the network parameters are homogeneous equations we will have a single fixed point $q(x)$ for any x; call its value q, and write the above equations as:

$$q = [\alpha q + c]/[r + bq] \quad \text{or} \quad bq^2(r - \alpha)q - c = 0,$$

yielding

$$2bq = (r - \alpha) + ((r - \alpha)^2 + 4bc)^{1/2}.$$

q is then the average grey level of the image which will be generated. Thus c must be chosen so as to bring q to the desired value. In other words, if we are given a desirable average grey level q, then we choose c as follows: $c = q(bq + r - \alpha)$.

5.1. THE TEXTURE GENERATION ALGORITHM

The texture generation algorithm is as follows. First choose c from a given value of q, and all of the directional weights w_d. Then:

Step 1. Generate at random a bitmap value of 0 or 1 for each pixel $x = (i, j)$ and assign it to the variable $q^0(x)$ for each x.

Step 2. Starting with $i = 0$ up to $i = N$ (the stopping condition) iterate on equations

$$\lambda^{(i+1)+}(x) = \sum_d q^i(x_d) r P_d + c,$$

$$\lambda^{(i+1)-}(x) = \sum_d q^i(x_d) r Q_d,$$

and compute $q^N(x) = \lambda^{N+}(x)/(\lambda^{N-}(x) + r)$.

Step 3. Compute the average grey level $G = [\sum_x q^N(x)]/[\text{size of picture}]$.

Step 4. Assign the grey level of each pixel as follows: $f(x) = 1$ if $q^N(x) > G$, $f(x) = 0$ if $q^N(x) \leq G$.

The directional weights w_d are the most important parameters for the textures which are obtained. This is illustrated in the results shown below. N, the number of iterations used, will also have an important effect. The role of the directional weights on the textures which are generated is illustrated in the examples shown in the following figures:

(a) If only one of the directional weights is given (fig. 7), anisotropic line textures will be generated.

(b) For the directional weights having the same signs:

- if horizontal and vertical directional weights are given, ordered checkboard-like patterns will be obtained;

Fig. 7. w_3, $w_7 = 1$; all others 0.

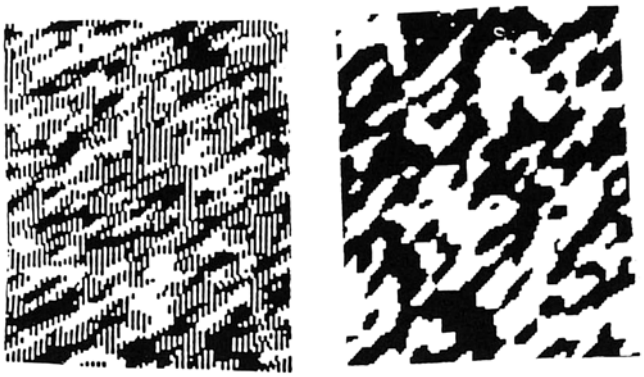

Fig. 8. (a, left) $w_0 = w_4 = w_1 = w_5 = 1$; others 0. (b, right) $w_0 = w_4 = w_2 = w_6 = w_1 = w_5 = 1$; others 0.

- if horizontal or vertical directional weights are given along with one of the diagonal directional weights (fig. 8a) an anisotropic effect with vertical or horizontal inhibition is observed;
- if both diagonal directional weights are given, isotropic vertical, horizontal and diagonal inhibitions may be seen;
- the cases for three and four directional weights can be extrapolated from the combinations of the above configurations (figs. 8b and 9).

(c) For directional weights having different signs (fig. 10):

- if horizontal and vertical directional weights are given, anisotropic inhibition of ordered vertical-horizontal patterns will be obtained;
- if horizontal or vertical directional weights are given along with one of the diagonal directional weights, anisotropic diagonal inhibition on an anisotropic line texture background is observed;
- if both diagonal directional weights are given, isotropic inhibition of ordered diagonal patterns may be seen.

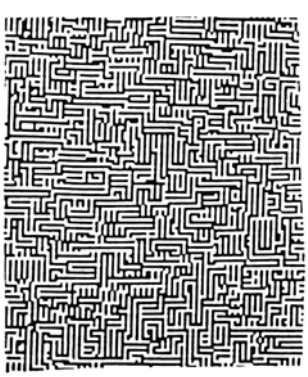

Fig. 9. $w_3 = w_7 = 1$, $w_6 = w_2 = -1$; others 0.

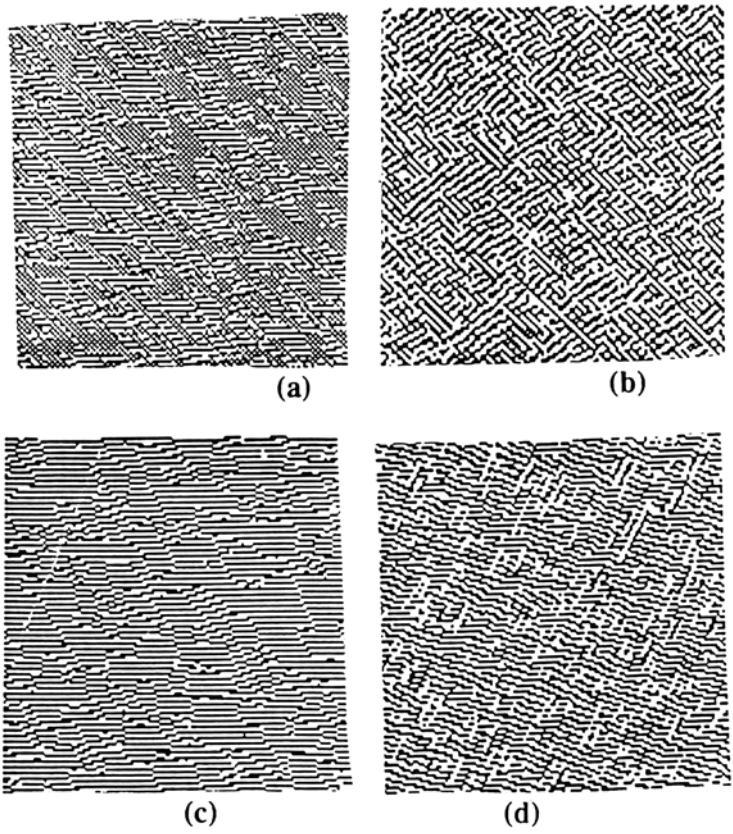

Fig. 10. Example textures generated with the G-network. (a) $w_0 = w_4 = -1$, $w_3 = w_7 = 1$. (b) $w_0 = w_4 = w_1 = w_5 = -1$, $w_3 = w_7 = 1$. (c) $w_0 = w_4 = -1$, $w_2 = w_6 = w_3 = w_7 = 1$ and all others 0. (d) $w_0 = w_4 = w_1 = w_5 = -1$, $w_2 = w_6 = w_3 = w_7 = 1$.

6. Multiple class G-networks [4]

Consider a network which generalises the basic model presented in [1, 2]. It is composed of n neurons and receives exogenous positive (excitatory) and negative (inhibitory) signals or customers, as well as endogenous signals exchanged by the neurons. As in [1, 2], excitatory or inhibitory signals are sent by excited neurons to other neurons in the network, or to the outside world. A neuron is excited if its potential is positive; it then fires at exponentially distributed intervals sending excitatory signals of *different classes*, or inhibitory signals of a single class, to other neurons or to the outside of the network. This model has been introduced in [4]. It represents a neural network which processes several streams of information in parallel, or a queueing network with multiple class positive, and single class negative customers. Of course, multiple class models are well known in queueing theory. Since positive signals may belong to several *classes*, the potential at a

neuron is represented by the vector $\boldsymbol{k}_i = (k_{i1}, \ldots, k_{iC})$ where k_{ic} is the value of the "class c potential" of neuron i, or its "excitation level in terms of class c signals". The total potential of neuron i is $k_i = \sum_{c=1,C} k_{ic}$.

Exogenous positive signals of class c arrive to neuron i in a Poisson stream of rate $\Lambda(i,c)$, while exogenous negative signals arrive to it according to a Poisson process of rate $\lambda(i)$. When a positive signal of class c arrives to a neuron, it merely increases k_{ic} by 1. Since negative signals belong to a single class, when a negative signal arrives to neuron i, if $k_i > 0$ the potential is reduced by 1, and the class of the potential to be reduced is chosen at random so that with probability k_{ic}/k_i it is of class c. As with single class networks discussed above, a negative signal arriving to a neuron whose potential is zero has no effect. When its potential is positive ($k_i > 0$), neuron i can fire; the potential depleted is of class c with probability k_{ic}/k_i, in which case the neuron fires at rate $r(i,c) > 0$, and sends to neuron j a class ξ positive signal with probability $(k_{ic}/k_i)p^+(i,c;j,\xi)$, or a negative signal with probability $(k_{ic}/k_i)p^-(i,c;j)$. The probability that the depleted signal is sent out of the network, or that it is "lost" is $(k_{ic}/k_i)d(i,c)$. The $\{p^+(i,c;j,\xi), p^-(i,c;j), d(i,c)\}$ are the transition probabilities of a Markov chain with state-space $\{1,\ldots,n\} \times \{1,\ldots,C\} \times \{+,-\}$ representing the movement of signals in the network, and for (i,c), $1 \leq i \leq n$, $1 \leq c \leq C$:

$$\sum_{(j,\xi)} p^+(i,c;j,\xi) + \sum_j p^-(i,c;j) + d(i,c) = 1.$$

The complete state of the network is represented by the vector (of vectors) $\boldsymbol{k} = (\boldsymbol{k}_1, \ldots, \boldsymbol{k}_n)$. Under the above assumptions, the process $\{\boldsymbol{k}(t), t \geq 0\}$ is Markovian, and we shall denote by $p(\boldsymbol{k},t) \equiv P[\boldsymbol{k}(t) = \boldsymbol{k}]$ the probability distribution of its state. Its behaviour is then described by the Chapman–Kolmogorov equations:

$$\begin{aligned}
\mathrm{d}p(\boldsymbol{k},t)/\mathrm{d}t = & -p(\boldsymbol{k},t)\sum_{(i,c)}[\Lambda(i,c) + (\lambda(i) + r(i,c))(k_{ic}/k_i)] \\
& + \sum_{(i,c)}\Big\{ p(\boldsymbol{k} + e_{ic}, t)r(i,c)((k_{ic}+1)/(k_i+1))d(i,c) \\
& + p(\boldsymbol{k} - e_{ic}, t)\Lambda(i,c)\mathbf{1}[k_{ic} > 0] + p(\boldsymbol{k} + e_{ic}, t)\lambda(i)((k_{ic}+1)/(k_i+1)) \\
& + \sum_{(j,\xi)}(p(\boldsymbol{k} + e_{ic} - e_{j\xi}, t)r(i,c)((k_{ic}+1)/(k_i+1))p^+(i,c;j,\xi)\mathbf{1}[k_{j\xi} > 0] \\
& + p(\boldsymbol{k} + e_{ic}, t)r(i,c)((k_{ic}+1)/(k_i+1))p^-(i,c;j)\mathbf{1}[k_j = 0] \\
& + p(\boldsymbol{k} + e_{ic} + e_{j\xi}, t)r(i,c)((k_{ic}+1)/(k_i+1)) \\
& \times ((k_{jc}+1)/(k_j+1))p^-(i,c;j))\Big\},
\end{aligned}$$

where we have used the notation:

$$\boldsymbol{k} + e_{ic} = (\boldsymbol{k}_1, \ldots, (k_{i1}, \ldots, k_{ic} + 1, \ldots, k_{iC}), \ldots, \boldsymbol{k}_n),$$

$$\boldsymbol{k} + e_{jc} = (\boldsymbol{k}_1, \ldots, (k_{j1}, \ldots, k_{jc} + 1, \ldots, k_{jC}), \ldots, \boldsymbol{k}_n),$$

$$\boldsymbol{k} + e_{ic} - e_{j\xi} = (\boldsymbol{k}_1, \ldots, (k_{i1}, \ldots, k_i + 1, \ldots, k_{iC}), \ldots,$$
$$(k_{j1}, \ldots, k_\xi - 1, \ldots, k_{iC}), \ldots, \boldsymbol{k}_n),$$

$$\boldsymbol{k} + e_{ic} + e_{j\xi} = (\boldsymbol{k}_1, \ldots, (k_{i1}, \ldots k_i + 1, \ldots, k_{iC}), \ldots,$$
$$(k_{j1}, \ldots, k_\xi + 1, \ldots, k_{iC}), \ldots, \boldsymbol{k}_n).$$

These vectors are defined only if their elements are non-negative.

THEOREM 4 [4]

Let $\boldsymbol{k}(t)$ be the vector representing the state of the neural network at time t, and let $\{q_{ic}\}$ with $0 < \sum_{(i,c)} q_{ic} < 1$, be the solution of the system of non-linear equations:

$$q_{ic} = \lambda + (i,c)/[r(i,c) + \lambda^-(i)],$$

$$\lambda^+(i,c) = \sum_{(j,\xi)} q_j \xi r(j,\xi) p^+(j,\xi;i,c) + \Lambda(i,c),$$

$$\lambda^-(i) = \sum_{(j,\xi)} q_j \xi r(j,\xi) p^-(j,\xi;i) + \lambda(i),$$

then the stationary solution $p(\boldsymbol{k}) \equiv \lim_{t\to\infty} P[\boldsymbol{k}(t) = \boldsymbol{k}]$ exists and is given by:

$$p(\boldsymbol{k}) = \prod_{i=1}^{n} (k_i!) G_i \prod_{c=1}^{C} [(q_{ic})^{k_{ic}}/k_{ic}!],$$

where the G_i are appropriate normalising constants.

7. G-networks extended to networks with triggered customer movement

Consider an open network of queues with n servers which have mutually independent i.i.d. exponential service times of rates $r(1), \ldots, r(n)$. Two types of entities circulate in the network: "customers" and "signals". External arrivals to the ith queue of the network can either be customers which arrive according to a

Poisson process of rate $\Lambda(i)$, or signals which constitute a Poisson arrival process of rate $\lambda(i)$. An arriving customer *adds 1* to the queue length. The queue length is constituted only by customers and their service is carried out in the usual manner. A customer leaves queue i (after finishing service), and heads for queue j with probability $p^+(i,j)$ as a customer, or as a *signal* with probability $p^-(i,j)$, or it will depart from the network with probability $d(i) = 1 - \sum_j p(i,j)$, $1 \leq i \leq n$. The matrix P, with elements $p(i,j) = p^+(i,j) + p^-(i,j)$ is the transition matrix of a Markov chain representing the movement of customers and signals. A signal arriving to an empty queue will have no effect, and will just disappear; if queue i is non-empty then one of the following two events occur:

- The arriving signal triggers the instantaneous passage of a customer from queue i to some other queue j with probability $q(i,j)$; Q denotes the corresponding n by n transition probability matrix.
- Or, with probability $D(i) = 1 - \sum_j q(i,j)$, it forces a batch of customers of random size to leave the network. Let the length of queue i be k_i at the instant of arrival of the trigger; if $k_i \geq B_i$, its length is reduced by B_i; if $k_i < B_i$, the queue length becomes zero. The distribution of batch size B_i is general and given by $P[B_i = s] = \pi_{is}$, $s \geq 1$.

Thus a signal acts as an external trigger which instantaneously moves a customer from one queue to the other, or a batch of customers to the outside world. We assume that $p(i,j)$ and $q(i,j)$ are the transition probabilities of transient Markov chains, so that each customer is guaranteed to leave the network with probability one. Let $k = (k_1, \ldots, k_n)$ be any n-vector of non-negative integers. Define the following vectors:

$$k_i^{+s} = (k_1, \ldots, k_i + s, \ldots, k_n), \qquad s \geq 1,$$

$$k_i^{-} = (k_1, \ldots, k_i - 1, \ldots, k_n),$$

$$k_{ij}^{+-} = (k_1, \ldots, k_i + 1, \ldots, k_j - 1, \ldots, k_n), \quad s \geq 1,$$

$$k_{ij}^{+s+} = (k_1, \ldots, k_i + 1, \ldots, k_j + s, \ldots, k_n),$$

$$k_{ijm}^{++-} = (k_1, \ldots, k_i + 1, \ldots, k_j + 1, \ldots, k_m - 1, \ldots, k_n).$$

We denote by $\{k(t) : t \geq 0\}$ the continuous time Markov chain representing the state of the network, where $k(t) = (k_1(t), \ldots, k_n(t))$ and $k_i(t)$ is the number of customers in queue i at time t. $\{k(t) : t \geq 0\}$ satisfies a system of Chapman–Kolmogorov equations, and its steady-state distribution $p(k) \equiv \lim_{t \to \infty} P[k(t) = k]$, if it exists, satisfies the following system of balance

equations:

$$p(k)\sum_i [\Lambda(i)+(\lambda(i)+r(i))\mathbf{1}[k_i>0]]$$

$$=\sum_i \Bigg[p(k_i^+)r(i)d(i)+p(k_i^-)\Lambda(i)\mathbf{1}[k_i>0]+\lambda(i)D(i)\sum_{s=1}^{\infty}\pi_{is}p(k_i^{+s})$$

$$+\lambda(i)D(i)\sum_{s=1}^{\infty}\pi_{is}\sum_{v=0}^{s-1}p(k^{+v_i})\mathbf{1}[k_i=0]$$

$$+\sum_j \Big\{ p(k_{ij}^{+-})[r(i)p^+(i,j)+\lambda(i)q(i,j)]\mathbf{1}[k_j>0]$$

$$+\sum_{s=1}^{\infty}\pi_{is}p(k_{ij}^{++s})r(i)p^-(i,j)D(j)$$

$$+\sum_{s=1}^{\infty}\pi_{is}\sum_{v=1}^{s-1}p(k_{ij}^{++v})r(i)p^-(i,j)D(j)\mathbf{1}[k_j=0]$$

$$+p(k_i^+)r(i)p^-(i,j)\mathbf{1}[k_j=0]+\sum_m p(k_{ijm}^{+++-})r(i)p^-(i,j)q(j,m)\mathbf{1}[k_m>0]\Big\}\Bigg], \tag{5}$$

where $\mathbf{1}[X]=1$ if X is true and 0 otherwise. The last four terms on the right-hand side of (5) cover the effect of signals which originate within the network: the first two of these cover the case where the arriving signal provokes the departure towards the outside of the network of a batch of customers, the next covers the case where the signal arrives to an empty queue, and the fourth deals with the case where the signal arriving from queue i to queue j causes the instantaneous motion of a positive customer from queue j to queue m, whose queue length therefore increases by 1.

The model with triggered customer movement was introduced in [7]. Here we generalise it to the case of batch customer removal introduced in [6]. Batch customer removal in the context of networks with negative and positive customers has been independently developed in [14].

Notice that the "negative customer" of [2] corresponds to the case of a triggered external departure of a single normal customer (in this section with probability $p^-(i,j)D(j)$).

Let $f_i(x)=[1-\sum_{s=1}^{\infty}\pi_{is}x^s]/[1-x]$ for $x\in\mathbb{R}$, and consider the following system of non-linear equations:

$$\lambda^-(i)=\sum_i r(j)q_j p^-(j,i)+\lambda(i),$$

$$\lambda^+(i) = \sum_j r(j)q_j\left[p^+(j,i) + \sum_m p^-(j,m)q_m q(m,i)\right] + \sum_j \lambda(j)q_j q(j,i) + \Lambda(i), \tag{6}$$

where $q_i \equiv \lambda^+(i)/[r(j) + \lambda^-(i)D(i)f_i(q_i)]$, $i,j,m = 1,\ldots,n$. Notice that $\lambda^+(i)$ may be written as

$$\lambda^+(i) = \sum_j r(j)q_j p^+(j,i) + \Lambda(i) + \sum_j \lambda^-(j)q_j q(j,i).$$

THEOREM 5

If a non-negative solution $\{\lambda^+(i), \lambda^-(i)\}$ exists to eqs. (6) such that each $\lambda^+(i) < r(i) + \lambda^-(i)D(i)f_i(q_i)$ for $i = 1,\ldots,n$, then:

$$p(k) = \prod_{i=1}^{n} [1 - q_i]q_i^{ki}. \tag{7}$$

A key question about the model concerns the existence of a solution to the non-linear eqs. (6). This can be treated in a manner similar to the approach taken in [5], though the matter is now more complicated.

THEOREM 6

If the matrix $[P^+ + Q]$ is substochastic and transient (i.e. it does not contain any ergodic classes), the solution to (6), $\{\lambda^+(i), \lambda^-(i)\}$, $i = 1,\ldots,n$, always exists.

Proof

The proof is based on Brouwer's theorem. Let us write (6) as:

$$\lambda^+(i) = \sum_j \lambda^+(j)g(j)p^+(j,i) + \Lambda(i) + \sum_j \lambda^+(j)h(j)q(j,i),$$

$$\lambda^-(i) = \sum_j \lambda^+(i)g(j)p^-(j,i) + \lambda(i),$$

where

$$g(i) \equiv r(i)/[r(i) + \lambda^-(i)D(i)f_i(q_i)],$$

$$h(i) \equiv \lambda^-(i)D(i)f_i(q_i)/r(i) + \lambda^-(i)D(i)f_i(q_i)], \quad i = 1,\ldots,n.$$

Notice that $g(i)$ and $h(i) = 1 - g(i)$ are continuous functions of $r(i) \geq 0$ and of $\lambda^-(i) \geq 0$, with $0 \leq g(i), h(i) \leq 1$ for all i. Define the following vectors:

$$\boldsymbol{\lambda}^+ \text{ with elements } \lambda^+(i),$$

$$\boldsymbol{\lambda}^- \text{ with elements } \lambda^-(i),$$

$$\boldsymbol{\Lambda} \text{ with elements } \Lambda(i),$$

$$\boldsymbol{\lambda} \text{ with elements } \lambda(i).$$

Let G be the diagonal matrix with elements $g(i)$. Then:

$$\boldsymbol{\lambda}^+ = \boldsymbol{\lambda}^+[GP^+ + (I - G)Q] + \boldsymbol{\Lambda}, \quad \boldsymbol{\lambda}^- = \boldsymbol{\lambda}^+ GP^- + \boldsymbol{\lambda},$$

or

$$\boldsymbol{\lambda}^+(I - [GP^+ + (I - G)Q]) = \boldsymbol{\Lambda}, \quad \boldsymbol{\lambda}^- = \lambda^+ GP^- + \boldsymbol{\lambda}.$$

The series $\sum_{n=0}^{\infty}[GP^+ + (I - G)Q]^n$ is geometrically convergent, because $G \leq I$ and $[P^+ + Q]$ is substochastic and does not contain any ergodic classes [15, p. 43 ff]. Hence

$$\boldsymbol{\lambda}^+ = \boldsymbol{\Lambda} \sum_{n=0}^{\infty} [GP^+ + (I - G)Q]^n,$$

so that

$$\boldsymbol{\lambda}^- - \boldsymbol{\lambda} = \boldsymbol{\Lambda} \sum_{n=0}^{\infty} [GP^+ + (I - G)Q]^n GP^-$$

Let $\boldsymbol{y} = \boldsymbol{\lambda}^- - \boldsymbol{\lambda}$. Call the vector function $F(\boldsymbol{y}) = \boldsymbol{\Lambda} \sum_{n=0}^{\infty}[GP^+ + (I - G)Q]^n GP^-$ where the dependence of F on $\boldsymbol{y}$ comes from G, which in turn depends on $\boldsymbol{\lambda}^-$. Notice that $F: [\mathbf{0}, F(\mathbf{0})] \to [\mathbf{0}, F(\mathbf{0})]$, and that it is continuous. Therefore, by Brouwer's fixed-point theorem, $\boldsymbol{y} = G(\boldsymbol{y})$ has a fixed-point $\boldsymbol{y}^*$. This fixed point will in turn yield the solution of (6):

$$\boldsymbol{\lambda}^-(\boldsymbol{y}^*) = \boldsymbol{\lambda} + \boldsymbol{y}^*, \quad \boldsymbol{\lambda}^+(\boldsymbol{y}^*) = \boldsymbol{\Lambda} \sum_{n=0}^{\infty} (F(\boldsymbol{y}^*)P^+)^n,$$

completing the proof. □

The result concerning existence of product form solutions for this class of

networks is now as follows. Let us denote $\rho_i \equiv \lambda^+(i)/[r(i) + \lambda^-(i)D(i)f_i(q_i)]$. Notice that by setting $\boldsymbol{y}^*$ in the values of $\lambda^+(i)$ and $\lambda^-(i)$, we obtain $\rho_i(\boldsymbol{y}^*)$.

THEOREM 7

Consider a G-network for which $\rho(\boldsymbol{k}) \equiv \lim_{t\to\infty} P[\boldsymbol{k}(t) = \boldsymbol{k}]$, if it exists, has the form:

$$p(\boldsymbol{k}) = \prod_{i=1}^{n} g_i(\boldsymbol{k}_i),$$

where each $g_i(\boldsymbol{k}_i)$ depends only on the k_i and the ρ_i, for $i = 1, \ldots, n$. Furthermore, assume that, for each $k_i \geq 0$, $\Gamma(k_i) \equiv \sum_{k_i}[g_i(k_i)]$ is such that for each $i = 1, \ldots, n$, $\{\sum_{k_i \geq 0} \Gamma(k_i)\}$ converges if $\rho_i(\boldsymbol{y}^*) < 1$, and diverges if $\rho_i(\boldsymbol{y}^*) \geq 1$. Then the stationary distribution $\rho(k) > 0$ for all k, of the G-network exists if $\rho_i(\boldsymbol{y}^*) < 1$, for all i. If $\rho_i(\boldsymbol{y}^*) \geq 1$, the stationary solution does not exist.

Proof

The stationary probability distribution $p(k)$ of the network satisfies the appropriate global balance equations, and the customer flow eqs. (1) always have a solution. Under the assumptions concerning $\Gamma(k_i)$, it is clear that the solution $p(k) > 0$ will exist if $\rho_i(\boldsymbol{y}^*) < 1$, for all i, and that it will not exist if $\rho_i(\boldsymbol{y}^*) \geq 1$ for some i. □

8. Conclusions

Starting with the queueing networks with positive and negative customers which we introduced in [1, 2], in this paper we have surveyed a new class of stochastic networks which we have discovered, the G-networks, which can be viewed as a unifying model for artificial neural networks and for queueing networks with enhanced and novel primitives. They have product form. Their relationship with standard neural network models is discussed and illustrated via examples, as well as two applications: to combinatorial optimisation in NP-hard problems [8] and to image texture generation [17].

Although they have product form solutions, these networks have an interesting and unusual property: their signal flow (or customer flow) equations are non-linear, while similar equations for product form networks which were known so far are linear. This leads to non-trivial questions regarding existence and uniqueness of their solutions and stability conditions. Sufficient stability conditions using the concept of "hyperstability" [2] are proved using non-linear analysis. Brouwer's fixed-point theorem is used to derive a general stability condition for G-networks.

The product form solution for multiple "signal" or customer class networks is established. We also generalise the model to "triggered customer movement" [7] with batch customer removal [6],

In [19], the G-network has been applied to the recognition of noisy digits and characters. Currently it is being applied to the travelling salesman problem, to optimum graph partitioning, to learning and compression of images, and to the segmentation of non-homogeneous image textures.

Acknowledgements

This work was accomplished with the pluriannual support of C3 CNRS, the French National Program in Distributed Computing, and with a grant from the French Ministry of Research (MRT) covering the years 1991–92.

References

[1] E. Gelenbe, Random neural networks with negative and positive signals and product form solution, Neural Comp. 1 (1989) 502–510.

[2] E. Gelenbe, Queueing networks with negative and positive customers and product form solution, J. App. Prob. 28 (1991) 656–663.

[3] E. Gelenbe, P. Glynn and K. Sigman, Queues with negative arrivals, J. App. Prob. 28 (1991) 245–250.

[4] J.M. Fourneau and E. Gelenbe, G-networks with multiple classes of signals, *Proc. ORSA Computer Science Technical Committee Conf.*, Williamsburg, VA (Pergamon Press, 1992).

[5] E. Gelenbe and R. Schassberger, Stability of G-networks, to appear in: *Probability and its Applications in the Engineering and Informational Sciences* (1992).

[6] E. Gelenbe, G-networks: Negative customers with batch removal, submitted to Perf. Eval. (1992).

[7] E. Gelenbe, G-networks with triggered customer movement, to appear in J. App. Prob. (1993).

[8] E. Gelenbe and F. Batty, Minimum graph covering with the random neural network model, *Proc. ORSA Computer Science Technical Committee Conf.*, Williamsburg, VA (Pergamon Press, 1992).

[9] D.E. Rumelhart, J.L. McClelland and the PDP Research Group, *Parallel Distributed Processing*, Vols. 1 and 2 (Bradford Books and MIT Press, Cambridge, MA, 1986).

[10] E.C. Kandel and J. H. Schwartz, *Principles of Neural Science* (Elsevier, Amsterdam, 1985).

[11] C.D. Garcia and W.I. Zangwill, *Pathways to Solutions, Fixed Points, and Equilibria* (Prentice–Hall, Englewood Cliffs, NJ, 1981).

[12] J.T. Sejnowski, Skeleton fields in the brain, in: *Parallel Models of Associative Memory*, eds. G.E. Hinton and J.A. Anderson (Lawrence Erlbaum Associates, Hillsdale, NJ, 1981).

[13] J.J. Hopfield and D.W. Tank, Neural computation in combinatorial optimization problems, Biol. Cyber. 52 (1985) 141–152.

[14] W. Henderson, B.S. Northcote and P.G. Taylor, Geometric equilibrium for queues with interactive batch departures, private communication (August 1992).

[15] J.G. Kemeny and J.L. Snell, *Finite Markov Chains* (Van Nostrand, Princeton, NJ, 1965).

[16] L. Hérault and J.J. Niez, Neural networks and combinatorial optimization: a study of NP-complete graph problems, in: *Neural Networks: Advances and Applications*, ed. E. Gelenbe (Elsevier, North-Holland, Amsterdam, 1991).

[17] V. Atalay, E. Gelenbe and N. Yalabik, Image texture generation with the random neural network model, *Int. Conf. on Artificial Neural Networks (ICANN-91)*, Helsinki (June 1991).
[18] E. Gelenbe, A. Stafylopatis and A. Likas, Associative memory operation of the random neural network model, *Int. Conf. on Artificial Neural Networks (ICANN-91)*, Helsinki (June 1991).
[19] M. Mokhtari, Application of the random neural network model to the recognition of typed images, to appear in Int. J. Pattern Rec. Art. Int. (IJPRAI).
[20] G. Cross and A. Jain, Markov random field texture models, IEEE Trans. PAMI-5 (Jan. 1983).
[21] S. Lakshmanan and H. Derin, Simultaneous parameter estimation and segmentation of Gibbs random fields using simulated annealing, IEEE Trans. PAMI-11 (August 1989).
[22] L. Onural and M.I. Gürelli, Generation and parameter estimation of Markov random field textures by highly parallel networks, in: *From Pixels to Features 2, ESPRIT BRA Workshop on Parallelism in Image Processing*, Bonas, France (August 1990).