Neural Comput & Applic (1993)1:124–146

Neural
Computing
& Applications

Artificial Neural Net Applications in Telecommunication Systems

T. Fritsch, M. Mittler and P. Tran-Gia

Institute of Computer Science, University of Würzburg, Am Hubland, D-8700 Würzburg, Germany

The artificial neural net development has had something of a renaissance in the last decade with an impressive range of application areas. From the viewpoint of telecommunication networks and systems, an increasing number of studies can be observed in recent literature dealing with proposed applications of neural nets in telecommunication environments, such as connection admission control in broadband networks, the control of high-speed inter-connection networks, channel allocation in cellular mobile systems, adaptive routing, etc. These proposed applications largely use three main neural net classes: feed-forward nets with backpropagation learning, Hopfield feedback nets, and self-organising neural nets. In this paper, we first give an overview of neural net classes and their main properties, and then present a review of applications in telecommunication systems, where attention is devoted to numerical aspects such as the convergence property and learning speed of the proposed neural nets.

Keywords: Hopfield; Backpropagation; Self-organising feature maps; Routing; Switching fabric; Admission control; Mobile communication; Interconnection networks; ATM; Packet radio networks; Channel assignment; Satellite scheduling

1. Introduction

Applications of artificial neural nets are *en vogue*. Their ability to generalize and optimize more quickly than some conventional algorithms has been observed in different areas of research such as speech recognition, financial forecasting, image data compression and noise reduction in signal processing. Neural nets take advantage of the redundancy incorporated in their distributed processing structures. Moreover, their ability to form some kind of internal representation of the supplied input vectors makes them very useful for recognising and distinguishing complex patterns.

In general, a neural net performs a functional approximation, whose accuracy only depends on the way in which the internal representation is built up. This affects the use of different transfer functions of the artificial neurons employed, different net structures or learning paradigms. Each application, whose subject can be formulated as a pattern recognition, pattern completion or pattern distinction problem, can use neural nets. In some areas, neural nets promise to be more efficient than conventional algorithms, especially in signal processing and speech recognition. In contrast, it is astounding that only a few publications deal with the use of neural nets in distributed computer systems.

In recent years, an increasing number of studies have used artificial neural nets in telecommunication systems, computer networks and data transmission technologies. For example, potential application fields are network planning, connection admission control, optimal path finding, tuning of wide area network parameters, error recognition and intruder detection, network reliability improvement, traffic estimation, congestion control, coding, etc.

In this paper, an overview of potential uses of neural nets in problems arising in telecommunication systems and networks will be given. We have two main purposes: first, to discuss proposed major neural net structures, and second, to describe the

Original manuscript received 29 March 1993

Correspondence and offprint requests to: T. Fritsch, Institute of Computer Science, University of Würzburg, Am Hubland, D-8700 Würzburg, Germany.

applications and, in particular, those steps from the problem statement and its mapping to a neural net, and its parameterization as well as numerical evaluation.

In a number of papers dealing with neural net applications in communication systems, numerical aspects are not often treated carefully. In some studies, it turns out that the proposed neural net can only be applied for a rather restricted set of parameters; in others, the convergence issues of the neural nets are not taken into account. Thus, we will focus on the numerical tractability of the application and the performance of the neural net compared with conventional solutions, where the parameterization problem will be discussed.

In the sections to follow we will concentrate on the application of neural nets in a number of specific applications, i.e. routing, admission control, switching and mobile communications. Applications in the following areas will not be described in detail in this paper: coding theory [1–6], data transmission [7–10] and signal processing related to telecommunications [11–17]. We first introduce some basic neural network structures, i.e. Hopfield nets, feedforward nets with backpropagation learning, and self organising feature maps. These are neural net structures which can be found in those applications described in section 3 of this paper.

2. Basic Neural Net Structures

2.1. Neuron Model

In the literature, some basic neural net structures are often used in proposed applications in telecommunication systems and networks: the classes of (i) feedback-oriented Hopfield nets, (ii) feedforward nets with backpropagation learning, and (iii) self-organising feature maps.

In general, the neuron model is the basic processing element of an artificial neural net. According to the commonly used model, a neuron performs a nonlinear mapping of a set of inputs x_i into an output v_j. The output v_j is computed as a weighted sum of inputs according to a transfer function, e.g. the sigmoidal function such as:

$$v_j = \frac{1}{1 + e^{\frac{-x_j}{x_0}}} \quad \text{with} \quad x_j = \sum_i w_{ij} v_i \,. \tag{1}$$

Other forms of the transfer function are:

$$v_j = \text{sign}(x_j) \quad \text{or} \quad v_j = \tanh(x_j) \,. \tag{2}$$

The term w_{ij} stands for the weight of the connection from neuron i to neuron j. Furthermore, for numerical reasons, a bias is often added to each x_j. The models of neurons described in the Eq. (1) and Eq. (2) are used for the two classes (i) and (ii) of the neural nets discussed here, while class (iii) uses vectors rather than neuron models as basic processing elements.

2.2. Hopfield Nets

Hopfield nets [18,19] form a well-known network structure class which is often proposed for use as a content-addressable memory or as an optimizer, e.g. in scheduling problems like the Travelling Salesman Problem.

A Hopfield net is an unlayered, fully connected net, the basic structure of which is shown in Fig. 1. The neurons are described in the previous subsection. In the following, we will consider neurons with a sigmoidal transfer function. The strength of the connections from a neuron i to a neuron j is described by the component w_{ij} of the weight matrix W. According to the working principle of the neuron model described above, the input signals to neuron j are summed and then transformed by the transfer function. As depicted in Fig. 1, the total input to neuron j is

$$x_j = \sum_{i \neq j} w_{ij} v_i + u_j \,. \tag{3}$$

The external input signal u_j adjusts the level of excitability of the network. This external bias is normally set to a constant value for all neurons.

In the following mathematical treatment of Hop-

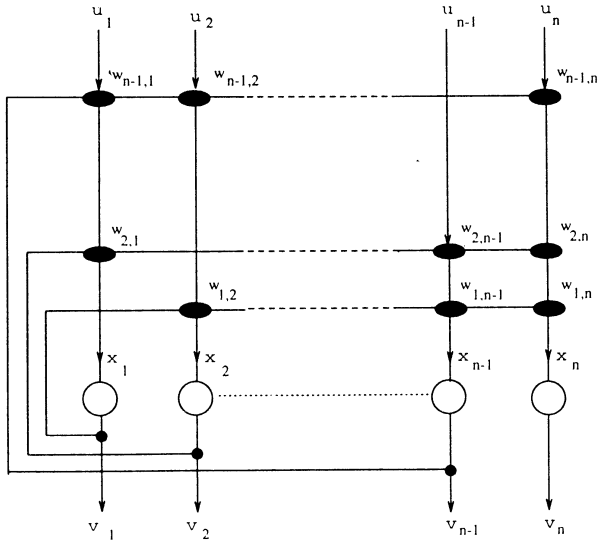

Fig. 1. Basic structure of a Hopfield net.

field nets, we will take the continuous-time version which leads to a set of differential equations. First, the equation indicating the time evolution of the net seen by a given neuron j is

$$\frac{dx_j}{dt} = \frac{-x_j}{\tau} + \sum_{i=1} w_{ij}v_i + u_j \ , \quad \tau = RC \tag{4}$$

$$v_i = g(x_i) \ , \quad g(x_i) = \frac{1}{2}\left(1 + \tanh\left(\frac{x_i}{x_0}\right)\right) . \tag{5}$$

The value τ determines the contribution of the values of x_j from time step $t-1$ to time step t. The variable x_0 in Eq. (5) modifies the slope of the hyperbolic tangent curve of the transfer function. Usually, the state of the Hopfield net can be described by means of an energy function such as

$$E = -\frac{1}{2}\sum_{i=1}^{N} v_i \sum_{j=1}^{N} w_{ij}v_j - \sum_{i=1}^{N} v_i u_i \ . \tag{6}$$

For the system of N neurons it guarantees that the system will settle down in a stable state, even though the solutions might not be the appropriate ones. Starting the net with initial random values of x_j and given weights w_{ij}, the net should converge to a stable state. During the iteration, the system's energy is minimised due to the negative sign of the energy function (Eq. (6)). Therefore, evaluating the final state of a Hopfield net corresponds to finding a minimum of this energy function, which may only be a local one.

To prove convergence of the Hopfield neural net in the continuous case, a system of N nonlinear differential equations (Eq. (2)) has to be solved. Since in the normal case no closed form of the solutions can be found, Hopfield takes advantage of the Liapunov function, which guarantees an asymptotic behaviour of the solutions to the differential equations [18,20]. It should be noted that there exists no mathematical prescription to construct a Ljapunov function, although the theory of differential equations supplies some preconditions that a Ljapunov function has to meet. (Interested readers are referred to the literature for a detailed description [20].) The Ljapunov function for the system of N neurons (Eq. (6)) is given by

$$E = -\frac{1}{2}\sum_{i=1}^{N} v_i \sum_{\substack{j=1 \\ j\neq i}}^{N} w_{ij}v_j + \sum_{i=1}^{N} 1/R_i \int_0^{V_i} g_i^{-1}(v)\,dv + \sum_{i=1}^{N} v_i u_i \ . \tag{7}$$

If W is symmetric, Eq. (7) guarantees the asymptotic convergence of Eq. (2). It should be noted that the state space in which the network is operating can be represented by an N-dimensional unit hypercube. The valid solutions of the neural network with values of 0 or 1 at each neuron are equivalent to the corners of that hypercube.

Basically, two main application of Hopfield nets can be observed: as content-addressable or associative memory, and as an optimiser in scheduling problems.

Content-addressable memory or associative memory
In this case, the patterns to be stored are presented to the neural net according to a learning rule. In the original version this rule is simply a scalar product sum of patterns viewed as vectors. The patterns can be thought of as stored in the weight matrix. In terms of an energy function, each pattern will be represented by a local minimum in the energy state space. In the recall phase, the net has the capability to recognize the patterns which can be superimposed with some noise. The usage of Hopfield nets will cope with some numerical problems if the patterns to be stored are correlated. This phenomenon leads to some modifications of the learning rule, for example, as proposed in [21].

Optimizer
In this case the scheduling is mapped into an energy function. The terms of this energy function correspond to the constraints of the scheduling problem to be solved. Based on this artificial energy function, the weight matrix, and thus the structure of the Hopfield net, can be obtained. The scheduling problem's solution is located in the global minimum of the energy function. This application of Hopfield nets is often proposed to solve problems in communication systems, as far as the problem can be formulated as a scheduling problem. Hopfield nets as a scheduler cope with serious numerical problems, e.g. determination of the Lagrange parameters contained in the constructed energy function. This is discussed later in more detail.

2.3. Backpropagation Nets

In the following, we use the term 'backpropagation net' as a shorthand notation of 'feed-forward neural net with error back-propagation'. The basic structure of a backpropagation net is depicted in Fig. 2, where a three-layer net is chosen as an example. The neural net consists of a number of neurons connected by weight vectors W_i. As shown in Fig. 2, it contains an input layer, one (or more) hidden layer(s), and an output layer, connected in a fully meshed, feed-forward manner. There are two operation modes: learning and recall.

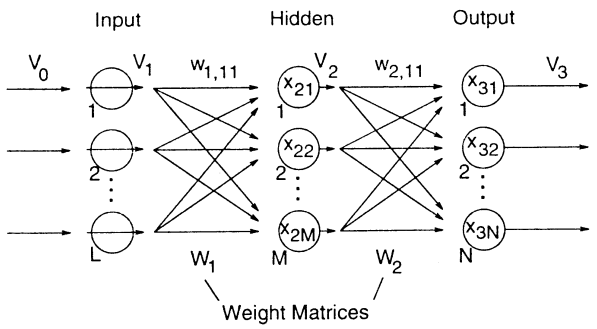

Fig. 2. Basic structure of a backpropagation net.

1. During the learning phase, a number of, for example, K pairs of input/output vectors $\{P_k, Q_k\}$, $k = 1, .., K$, are presented to the net. It computes its own output vector according to the equations mentioned above, and compares the computed output vector with the presented input vector. The comparison results in an error vector, which will be used to change the weight matrices according to a learning rule. A steepest descent gradient method is chosen as the learning procedure in the majority of papers describing applications of backpropagation nets. However, this method is not the most efficient when compared to other known numerical methods such as those discussed in [22]. The learning phase will end if all input/output pairs to be learned have been presented, and the total error is lower than a predefined threshold. After completion of the learning phase, the information about the input/output pairs, which represents a mapping, can be seen as being stored in the weight matrices.
2. In the recall mode, the backpropagation net can be used as an error-correcting classificator, where the response Q_k is expected from the net when an input vector of type $P_k + R$ is presented to the net. Thereby, R represents a noise component added to the input vector.

2.4. Self-organising Feature Maps

In contrast to the two neural net structures described in the two previous subsections, the class of *self-organising feature maps* employs multi-dimensional vectors as neurons, which are again the basic processing elements of the net.

Basically, a self-organising neural net mapping approximates a continuous topological mapping $\phi : V \subset \mathbf{R}^n \rightarrow A \subset \mathbf{R}^m$, which is implicitly defined. The term V represents an n-dimensional subspace of $\mathbf{R}^n$, and the term A an m-dimensional subspace of $\mathbf{R}^m$, the so-called target space. The structure of the net is shown in Fig. 3.

The basic principle of the self-organising algorithm stems from *competitive learning*, which has been investigated by Kohonen and Grossberg [23]. The idea of the algorithm will be briefly described in the following. A sequence of statistical samples of a vectorial observable $\mathbf{v} = v(t) \in \mathbf{R}^n$ (t is the time coordinate) is presented to the net. In a previous step the weights interconnecting the N input neurons with the M neurons in the two-dimensional mapping array have already been set to small uniformly distributed random numbers. Most important in this context is the definition of a suitable *environment*, i.e. neighbourhood function, which is shrinking in the course of time. At the beginning the neighbourhood function, $N(t)$ covers half of the net array and reduces its size accordingly during progression of the self-organising process.

The self-organising process can be characterised by the following steps:

1. Presentation of a new input vector $\mathbf{v(t)}$.
2. Computation of the distance d_j between all input neurons and all mapping array neurons j according to the following formula:

$$d_j = \sum_{i=1}^{N} (v_i(t) - w_{ij}(t))^2 \tag{8}$$

   with $v_i(t)$ as the i-th component of the N-dimensional input vector and $w_{ij}(t)$ as the connection weight between input neuron i and mapping array neuron j at time t corresponding to the Euclidean metric.
3. Selection of the mapping array neuron j^* with minimal distance d_j^*.
4. Update of all weights, restricted to the actual topological neighbourhood $N_j^*(t)$ according to

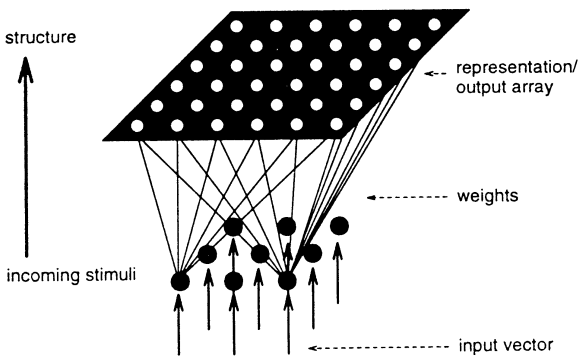

Fig. 3. Basic structure of a self-organising feature map.

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)(v_i(t) - w_{ij}(t)) \quad (9)$$

for $j \in N_j^*(t)$ and $1 \leq i \leq N$. Here $\eta(t)$ represents a monotonically decreasing function of the environment.

5. Iteration of the above steps until a predetermined error criterion is reached.

Due to the shrinking of the neighbourhood area during the self-organising process (which can be mathematically treated as a *Markov process* [24]), and since the adaptation of weights only takes place in the neighbourhood of the best-matching neuron, this process can be interpreted as a *Voronoi tesselation* of the input space (see [23] for further details.) A further result is that the point density function of the vector centroids approximates the probability density function (pdf) of the input vectors, and so provides a topological ordering both temporally and spatially.

Extensions of the algorithm concerning variances in input dimensions [25], approximation of discontinuities in the underlying probability density function [26] and adaptation to non-arbitrary topologies by dynamically defined neighbourhoods along *minimal spanning trees* (MST) [25] can be found in numerous recent studies.

3. Applications in Telecommunication Systems and Networks

In this section we discuss in more detail some applications and proposals for the use of neural nets in telecommunication environments. In particular, we take the following examples:

1. Feedback-oriented neural nets, Hopfield nets:
   - routing in communication networks
   - scheduling in high-speed interconnection networks
   - channel allocation in mobile communication systems.
2. Feedforward neural nets, backpropagation nets:
   - connection admission control in broadband networks
   - routing in communication networks.
3. Self-organising neural nets:
   - satellite scheduling.

We also devote attention to numerical aspects, which play an important role in discussions about the applicability of neural nets in a particular problem.

3.1. Routing in Communication Networks

3.1.1. Problem statement. We consider a communication network consisting of a number of nodes connected via communication links. The main aim in designing routing schemes is to find the most appropriate path from an originating node A to a terminating node B while taking into account a number of constraints. Let us call this path the optimal path. From a user viewpoint, the optimal path should guarantee a certain level or grade of service (GoS) parameters, which can be a delay or blocking probability threshold, and the optimal path should be cost-effective. From the network viewpoint, the optimal path should not touch network areas actually under overload, and thus prevent a magnification of the overload state, should use transmission links efficiently, and should minimise switching overhead involved. All of these factors are often expressed in the form of a cost function, e.g. a link as depicted in Fig. 4. The cost function may contain factors concerning, for example, transmission or transfer delays, or the use of transmission and switching functions. An example is given later in this subsection.

There exist various routing strategies, depending on implementation aspects as well as the need to include existing networks. From the network management and control points of view, three types of routing strategy can be distinguished: centralised, distributed and isolated routing. Depending on the ability of a routing strategy to react to dynamic traffic changes in the network or network topology, a classification into fixed, alternate and adaptive routing schemes can be made. Existing routing algorithms are often static in nature, using routing tables [27] as decision help. The estimated traffic is used to fix one or two paths through the network for each origination-destination pair of nodes. These paths are stored in the routing table, and are kept constant until the next change of network topology.

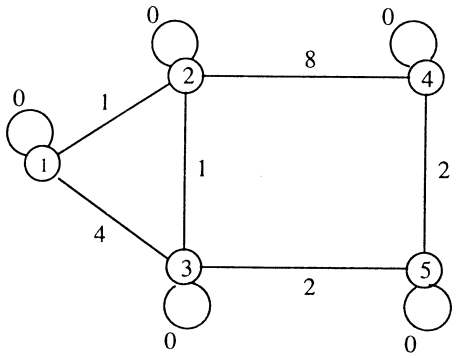

Fig. 4. A network example.

Neural nets to be used as routing schedulers in communication networks can be found in a number of studies [28–32], where the Hopfield type of neural nets is employed. In the next subsection we present this class of neural net router in more detail; later we discuss work using other net types, e.g. neural net application for adaptive routing in survivable communication networks [33] and for dynamic routing [34], both implementing the back-propagation neural net type.

3.1.2. Hopfield net in routing. As described above, the routing problem can be formulated as a scheduling problem. Thus, the Hopfield type of neural nets used as a scheduler appears as a possible solution. In [28] a neural net was first applied to the routing problem in communication networks.

In Fig. 4 an example of a network which is subject of routing is depicted. The network consists of N nodes. From node i to node j a cost factor w_{ij} is shown. Assuming that the cost between two nodes is independent of the transfer direction, a symmetric $N \times N$ cost matrix W with entries w_{ij} is obtained. Positions in the matrix with a non-existing link are set to large values symbolising the cost of infinity. The cost function is indicated in [28] as the expected delay across a link between two nodes, and depends on the link capacity c_{ij} and the actual traffic condition f_{ij}. The link cost w_{ij} can then be determined by defining w_{ij} as a function of c, e.g. $w_{ij} = f_0 + (f_{ij}/c_{ij})^2$, where f_0 denotes the transmission time on the link.

In turn, the cost-effective path through the network can be described by a $N \times N$ permutation matrix, where the rows represent the nodes of the network and the columns the position of the node in the path. By this method only those paths with a maximum number of N nodes can be represented (cf. Table 1). When we allow a node to appear many times on the path, the resulting path representation matrix is not a permutation matrix, but the obtained path length can be shorter. In this case, the cost factor from a node to itself is set to zero (cf. [29]).

Note that as a result, there are only zero-entries on the main diagonal. Then the following conditions can be derived from the description of the permutation matrix:

1. There are exactly N entries with value 1:
$$\sum_{i=1}^{N} \sum_{j=1}^{N} x_{ij} = N \,. \tag{10}$$
2. Each column contains exact one entry with value 1:
$$\sum_{i=1}^{N} x_i = 1 \,. \tag{11}$$

A neural net of the Hopfield type with N^2 neurons organised in a matrix-form is now employed. Each neuron in this matrix represents the position of a node in the path evaluated. Furthermore, in consideration of the conditions above, including a third term depending on the cost matrix W, the Hopfield energy function can be reformulated:

$$E = \frac{A}{2} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \sum_{k=2}^{N-1} w_{ij} v_{ik} (v_{jk+1} + v_{jk-1}) \qquad \text{Weighting of costs}$$
$$+ \frac{B}{2} \sum_{k=2}^{N-1} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} v_{ij} v_{jk} \qquad \text{Column restriction}$$
$$+ \frac{C}{2} \left(\sum_{i=1}^{N} \sum_{j=1}^{N} v_{ij} - N \right)^2 \tag{12}$$
$$\text{Exactly } N \text{ entries equal to 1 .}$$

The quadratic terms of Eq. (12) define the connection (weight) matrix T, and the linear terms of the external bias u. Replacing T and u in Eq. (5) leads to

$$\frac{dx_{ij}}{dt} = -\frac{x_{ij}}{\tau} - A \sum_{m=1}^{N} w_{im} (v_{m,j+1} + v_{m,j-1}) - B \sum_{\substack{m=1 \\ m \neq i}}^{N} v_{mj} - C \left(\sum_{i=1}^{N} \sum_{m=1}^{N} v_{im} - N \right)^2 \tag{13}$$

Table 1. Path represented by a permutation matrix.

	1	2	3	4	5
1	1	0	0	0	0
2	0	0	1	0	0
3	0	1	0	0	0
4	0	0	0	1	0
5	0	0	0	0	1

Table 2. The resulting cost-matrix.

	1	2	3	4	5
1	0	1	4	∞	∞
2	1	0	1	8	∞
3	4	1	0	∞	2
4	∞	8	∞	0	2
5	∞	∞	2	2	0

Given a source-destination pair A–B, A will take the first and B the last column. At initialisation of the numerical procedure, the values of the first and last column in the neuron matrix are preset accordingly, to 1 for the source and the destination nodes. The other columns are initialised with random values, taking into account that the sum over all matrix elements is N. Within each iteration of the Hopfield net, a system of N differential equations (Eq. (13)) has to be evaluated; this step is then repeated until convergence. It should be noted here that the effort for solving the optimisation problem rises with N^2.

A number of questions arise with the use of a Hopfield net to solve routing problems in particular, and scheduling problems in general. First, there exists the problem of finding a common parameter set for the Hopfield net, which can be used for a large variety of communication network topologies under different traffic conditions. Second, the stability of the solutions found by the Hopfield net must be confirmed in a mathematical sense rather than just through simulations. Furthermore, we observed that the use of appropriate numerical techniques depends on the properties of the differential equations to be solved. This is discussed later in more detail.

It was shown by Rauch and Winarske [28] and Zhang and Thomopoulos [29] that good results can be achieved using the Hopfield net, but only under the condition that the optimisation parameters A, B and C used as Lagrange parameters in Eq. (12) are determined after numerous simulations. There is no proof that these parameters work as well with nets of a higher dimensionality or a distinct topology. In [29], all simulations were carried out for a 5- and a 9-node-network with a constant cost-matrix and predetermined Lagrange parameters. This indicates that there were problems concerning the general use of these parameters. It was noticed that the algorithm is sensitive to these parameters, since a bad operating point may result in divergence or oscillation. A general applicability of the Hopfield net for routing problems thus depends on the answer to this crucial question. These problems are also expected to be applicable to simulations such as those presented in a subsequent paper [35] showing results for a 16-node network using the same parameters as for the 9-node network. In [31] and [32] it was shown that the tendency of the net solutions to oscillate depends on the *stiff* character of the underlying differential equations. In particular, the step length of the Euler method used cannot be adapted correctly to this class of *stiff* differential equations.

In general, Hopfield nets often cope with stiff differential equations, but are dependent on the Lagrange parameters and the number of neurons. As a consequence, strange effects may occur, also noticed in [29] and [35]. The exact result of an analytic solution and the numerical results diverge by a large degree. Mathematically, it is extremely difficult to forecast the appearance of stiff behaviour, therefore it is recommended that more appropriate methods are used (cf. [36]).

In [31] it was pointed out that, based on simulation results, no common parameter sets for nets of the same topology, nor for nets of a distinct topology, exist. The stiff behaviour of the differential equations was shown to be independent of the step length, therefore the classical Hopfield net, even with Lagrange parameter adaptation, cannot generally be applied in a practical sense. Unfortunately, there exist valid parameter sets to solve a given routing problem, but it becomes intractable to test a sufficiently large number of sets, until the right one is found. Even the modification of the Hopfield net by *simulating annealing* led to acceptable results only within a small range of network dimensions. Suppose that the parameters are optimised for a given network size, the neural net applied to a network of different size will probably not be able to find valid solutions when the same parameters are used. Referring to an intrinsic analysis of the eigenvalues of the connection (weight) matrix of the Hopfield TSP net (as discussed in Aiyer *et al.* [37]), an implementation of this method to solve routing problems was reported in [31] and [32]. Regarding the connection matrix of the Hopfield net, Aiyer et al. show that there exist relationships between the eigenvectors of the connection matrix and the space of solutions. More precisely, the corresponding eigenvectors to the eigenvalues of the connection matrix span a subspace of possible solutions. From a geometric view of this subspace, analytic expressions can be derived to determine the Lagrange parameters' independence of the dimension of the nets.

The connection matrix T of Eq. (12) can be found, for example, by means of partial differentiation, as follows:

$$T_{ij,mn} = -Aw_{im}(\delta_{n,j+1} + \delta_{n,j-1}) - B\delta_{j,n}(1 - \delta_{i,m}) - C\,, \qquad (14)$$

with $\delta_{ij} = 1$ if $i = j$ and 0 otherwise, and w_{im} as the cost between nodes i and m. The terms with parameter B and C are responsible for ensuring that the net achieves valid solutions. The term with the parameter A restricts the solutions to paths

with a minimal length, therefore the eigenvalues are determined only from the B and C terms. Thus, the eigenvalues of T are

$$\lambda_1 = -B(N-1) - CN^2$$

Confines the solutions to the subspace with exact N neurons having values 1.

$$\lambda_2 = B$$

Moves the solution towards the edges of the N-dimensional hypercube.

$$\lambda_3 = -B(N-2)$$

Confines the solution to the valid subspace.
Using these eigenvalues, the parameters B and C can be determined and the corrected connection matrix T is

$$T_{ij,kl} = -Aw_{ik}(\delta_{l,j+1} - \delta_{l,j-1}) - B_1\delta_{ik}\delta_{jl} - B\delta_{jl}(1 - \delta_{ik}) - C + \frac{BN + B_1}{N^2}. \quad (15)$$

Here B_1 denotes a new parameter, which determines the relative strength of λ_1 and λ_3.

Let us now discuss some mathematical properties concerning the eigenvalues. The recognition of the importance of the eigenvalues leads to a further aspect. From a mathematical viewpoint, the *stability* of the set of differential equations cannot be guaranteed with respect to the stability theorem by Ljapunov, as illustrated in [20]. For the arbitrarily chosen parameter B the symmetric connection matrix T is *not positive definite*, which is the necessary condition for stability. The positive definite property implies that all eigenvalues of T have values greater than 0. The eigenvalues λ_1, λ_2, λ_3 of T never have the *same* sign, choosing B negative or positive. Therefore, the sufficient condition cannot be satisfied for all different kinds of stability, as indicated in [20].

A modified differential equation set has been used to test the eigenvalue model with networks of size 10, 15 and 20 nodes. Results for the network with 20 nodes (Fig. 5) are shown in Table 3 [31]. It can be seen that two of the source-destination pairs (rows 4, 5) show wrong solutions. Looking at the wrong solutions one can observe an important behaviour of the Hopfield net. The net tries to find the best solution in the sense of a *global* optimization. Therefore, nodes in the communication network with the *largest degree* in a graph-theoretical sense are favoured, because those links add the least cost to the energy function in each iteration, thus minimising the energy of the net in a stronger way. As the influence of those nodes increases, the larger the networks become.

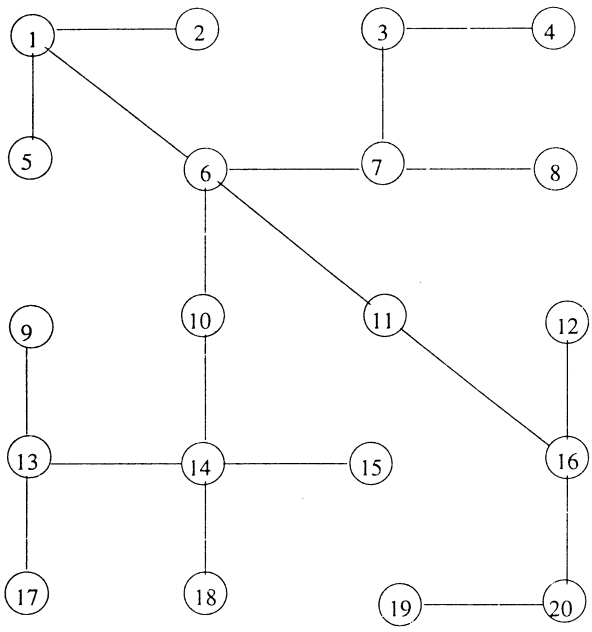

Fig. 5. Network example with 20 nodes.

Table 3. Results for a network with 20 nodes. A = 1,7 B = 8,0 $B1$ = 7,7 C = 0,34.

	Path	Iterations	Result	Valuation
1.	1–20	1256	1–6–11–16–20	optimal
2.	1–17	1543	1–6–10–14–13–17	optimal
3.	1–2	1246	1–6–11–6–1–2	optimal
4.	4–8	1058	4–6–7–8	wrong
5.	4–18	1314	4–6–10–14–18	wrong
6.	6–18	1374	6–10–14–18	optimal
7.	9–15	1650	9–13–14–10–6–10–14–15	optimal
8.	17–18	1008	17–13–14–10–6–10–14–18	optimal

A further interesting modification of the Hopfield net was recently introduced [30], where the energy function has been constructed differently to [29]. The authors focus on the optimum bifurcated quasi-static routing problem, with the goal of minimizing the network-wide average time delay. The approach outlined there reflects the fact that in practice the link cost in a communication network is usually time varying, depending on the amount of traffic flow the links carry. Consequently, this approach is more appropriate to the optimal routing problem [38].

The neural net is similarly constructed to that used in [29], where the following modification took place. First, the fact is taken into account that a link between some nodes, say x and i, may not exist. Thus, an additional term is included in the energy function, containing information as to

whether the specified link exists or not. The new term ρ_{ij} has the value 1 if the link between node i and j exists, otherwise it is 0. The neural net consists of $n \cdot (n-1)$ neurons, one neuron for a communication node in the possible path, except those on the diagonal. The output of a neuron v_{ij} is assumed to be 1 if the arc from node i is in the shortest path, and 0 otherwise. Like the energy function in [29], this function has to be minimized and the lowest energy state is equivalent to the shortest path. Distinct from [29], Ali and Kamoun [30] construct an energy function consisting of five terms, which are mostly linear after linearization of Eq. (16):

$$E = \frac{A}{2} \sum_{\substack{i=1 \\ (i,j)\neq(d,s)}}^{N} \sum_{j=1}^{N} w_{ij} v_{ij} + \frac{B}{2} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ i\neq j \\ (i,j)\neq(s,d)}}^{N} \rho_{ij} v_{ij}$$

$$+ \frac{C}{2} \sum_{i=1}^{N} \left\{ \sum_{\substack{j=1 \\ j\neq i}}^{N} v_{ij} - \sum_{\substack{j=1 \\ j\neq i}}^{N} v_{ji} \right\}^2$$

$$+ \frac{D}{2} \sum_{j=1}^{N} \sum_{\substack{j=1 \\ i\neq j}}^{N} v_{ij}(1-v_{ij}) + \frac{E}{2}(1-v_{ds}) \,. \qquad (16)$$

Here the indices d and s denote destination and source locations. The A-term minimises the *total cost of a path*, taking into account the cost of existing links. The B-term excludes non-existing links. The C-term ensures that if a node has been entered, it will also be exited by a path, thus it is guaranteed that there will be at most only one '1'-neuron at each row and at each column. The D-term forces the neurons to converge to one of the corners of the hypercube defined by the states of the neurons. The E-term is zero if the output of the neuron at location (d, s) has the value 1.

Considering the dynamics of a neuron at location (i, j) according to the Hopfield dynamics in Eq. (5) and substituting the energy function above into Eq. (5), the connection matrix can be obtained after a comparison of corresponding coefficients. This leads to the following connection strength $T_{ij,kl}$ between the neuron at location (i, j) and the neuron at location (k, l).

$$T_{ij,kl} = D\delta_{ik}\delta_{jl} - C\delta_{jl} + C\delta_{li} + C\delta_{jk} \,. \qquad (17)$$

The external input u_{ij} is given by

$$u_{ij} = -\frac{A}{2} w_{ij}(1 - \delta_{i,d} \cdot \delta_{js})$$

$$- \frac{B}{2} \rho_{ij}(1 - \delta_{id} \cdot \delta_{js}) - \frac{D}{2} + \frac{E}{2} \delta_{id}\delta_{js} \,. \qquad (18)$$

The advantage of the proposed scheme for connection matrix and externally supplied input is its flexibility, due to the fact that the link costs w_{ij} and the network topology information is mapped in to the biases rather than into the neural interconnections. Therefore, changes in topology can be expressed in terms of external inputs. This Hopfield net, however, underlies the same restrictions concerning the Lagrange parameters, as explained above.

Another approach to the routing problem by neural nets of Hopfield type can be found in [39], which is focused on routing in multihop radio networks. A decentralized structure for packet radio networks (PRNs) with local broadcast properties is considered. The use of frequency hopping techniques or spread spectrum as in *code division multiple access* (CDMA) leads to a scheduling-routing problem, which is indeed NP-complete. The topology of a PRN can be described by a graph $G = (N, L)$ with network nodes N and connection links L. Under the assumptions of a conflict-free channel access, the assignment of noise-free, periodical time-slots to unique stations, and further, the restriction to one receiver or sender, respectively, for each station and a distance between stations, which excludes interference of the same channel used, the routing problem can be formulated as described below.

Given a set of N_{sd} source-destination (SD) pairs and a set of paths connecting each SD pair, select a single path between each SD pair so that network congestion is minimised. The formulation of the routing problem as an optimisation problem allows us to use a Hopfield net. Each path of an SD pair is represented by a neuron, with a subscript (ij) indicating the j-th path of the i-th SD pair. The values of the neurons v_i are in the range between 0 and 1. Most important is the definition of the energy function, which can be expressed by the following terms:

$$E = -\frac{1}{2} \sum_{i=1}^{N_{sd}} \sum_{k=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{l=1}^{N_p(k)} T_{ij,kl}\, v_{ij}\, v_{kl}$$

$$- \sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} v_{ij}\, u_{ij} \,, \qquad (19)$$

where N_{sd} denotes the number of SD pairs, $N_p(i)$ the number of paths of SD pair i, v_{ij} the output value of neuron ij, $T_{ij,kl}$ the connection strength between neuron ij and kl, and u_{ij} the external input.

It has been shown [40] that for the case of continuous traffic and for a certain class of network topologies, the selection of paths that minimise the maximum node degree (defined by the sum of all incoming and outgoing flows of the node) in the

network permits the establishment of schedules with a minimal length. As this performance measure cannot be transformed into a Liapunov energy function, Wieselthier *et al.* [39] proposed the following congestion measure:

$$E_b = \frac{1}{2}\sum_{i=1}^{N_{sd}}\sum_{\substack{k=1\\k\neq i}}^{N_{sd}}\sum_{l=1}^{N_p(k)} |P_{ij} \bigcap P_{kl}| v_{ij} v_{kl} \qquad (20)$$

with P_{ij} the j-th path of SD pair i and $|P_{ij} \cap P_{kl}|$ the number of nodes located both in paths P_{ij} and P_{kl}. Including the following constraints that (i) no more than one path per SD pair is to be activated or selected, (ii) a total of exactly N_{sd} paths in the net is to be activated, and (iii) exactly one path per SD-pair is to be activated, the Lagrange multipliers method can be used to yield the final form

$$E_{\text{total}} = bE_b + \sum_{i=1}^{3} \lambda_i E_i - I \sum_{i=1}^{N_{sd}}\sum_{j=1}^{N_p(i)} v_{ij} \qquad (21)$$

with the coefficients b, λ_i, $I > 0$ and E_i for each constraint from above. The third constraint is redundant, but helps to achieve convergence.

The connection weight $T_{ij,kl}$ is the sum of all coefficients in E_{total} that multiply the product $v_{ij}v_{kl}$ in the equation for E_{total}, and is then given by the following expression:

$$T_{ij,kl} = -b|P_{ij} \bigcap P_{kl}|(1-\delta_{ik}) - \lambda_1\delta_{ik}(1-\delta_{jl}) - \lambda_2 - \lambda_3\delta_{ik} . \qquad (22)$$

Similarly, the coefficients of the linear terms correspond to bias values, and so u_{ij} is the sum of all coefficients that multiply v_{ij}.

In the first simulations with constant coefficients λ and with 24 nodes in the network, the quality of the solutions was not satisfactory, and determination of the adequate coefficients appeared to be rather heuristic. Therefore, the authors modified the energy equation by introducing variable coefficients λ_i. Within each iteration, all λ_i's were increased according to

$$\lambda_i(n+1) = \lambda_i(n) + (\Delta t)_\lambda E_i(n) \qquad (23)$$

with Δt as the Lagrange multiplier step width.

The simulations performed with the modified Lagrange method yielded reasonable results for both 24- and 100-node networks, giving 100% valid convergences, even better than those achieved by an additional simulation with MFA (mean field annealing) and constant coefficients. The simulations were performed from 50 different initial states of the network, and also used a modification with *multiple Lagrange multipliers* which, nevertheless, did not achieve significant improvements. Additionally, the authors tested other metrics, and reported solutions for non-uniform traffic and alternate routing, i.e. splitting the traffic over two or more routes. They concluded the interesting report by introducing a new model, which is an alternate formulation of the congestion-minimisation problem in which neurons are defined for each link along every path, rather than one for each total path, leading to a link-neuron model [39].

3.1.3. Routing with feedforward nets. Although backpropagation nets are mainly used for pattern recognition problems, there are some studies [33, 34] dealing with the application of this class of neural nets to routing problems.

A distributed approach for routing is considered in [35], where several small controllers share the routing function. This implies that there is no need to control the network-wide system state; each controller only needs information on its local neighbourhood. The design also supplies redundancy to the controller in order to cope with local malfunctions. A feedforward net is used in conjunction with the Hebbian learning principle.

Using an ordinary feedforward neural net for each node in the communication network, consisting of one input node for each possible destination site in the input layer and an output node for each neighbour of the site where the neural net is located, training of this net occurs each time a packet is received at a site. The controller asserts a 1 to the corresponding input node, the packet originated from, and 0 for all other neurons. The corresponding output node is assigned the time, the packet required, to reach this node from its origin. This so-called backward learning [27] can be interpreted as Hebbian learning, using mutually orthogonal input patterns for all different destinations. The net solution considers by that kind of learning only the total time in transit to the destination. For reasons of comparability, the simulations performed included four routing paradigms: random routing, hot potato routing [27], static routing, and neural-based Hebbian routing. The special coding of the input patterns make it seem reasonable that the neural net solution approximates the performance of static routing after a certain tuning phase.

The second article concerns the relationship between decision making in team theory and neural nets for dynamic routing in communication networks [34,41]. The authors say that there are two main reasons why using neural nets can become advantageous. The first is that due to the necessary delays throughout the whole network, it may be impossible

for a unique decision centre to gather and forecast all information. That is important for characterising the network status and, furthermore, to send reliable routing decisions to the nodes. They intend to construct an informationally decentralised decision structure, in which the routing nodes operate as 'the cooperating decision makers of a team'. Here it is meant that each node takes information out of its local neighbourhood, i.e. the contents of queues at the node and, additionally, some messages from neighbouring nodes. The second issue, addressed in this paper, is the place where the routing strategies are computed. It is assumed that the availability of a distributed algorithm, enabling the nodes to adapt to local routing strategies, constitutes an attractive property.

The authors formulate the dynamic routing problem as a team optimisation problem, more especially as an approximation of the original team problem stated in terms of functional optimisation to a parametric one. This is accomplished by assigning each routing node a set of multi-layer feed-forward neural nets capable of providing the network with routing decisions. The synaptic weights are then adjusted by the well-known back propagation algorithm.

The authors regard the communication network traffic flow, and consider the content of the local queues at node i at time t with the continuous state variable $x_i(t)$. A new state $x_i(t + 1)$ consists of the amount of traffic held in the queues of node i plus the incoming traffic from nodes 'upstream', along with a term $r_i(t)$ which has not been defined in the publication (it could represent the traffic originating from node i). The aim is the minimisation of the weighted traffic delay, i.e. the cost

$$W = \sum_{t=0}^{T-2} \sum_{i \in \hat{N}} \alpha_i x_i(t + 1) + \sum_{i \in \hat{N}} \alpha_{i,T} x_i(T) \qquad (24)$$

with α_i, $\alpha_{i,T}$ as positive constants. The cost W is assumed to be the sum over all states of all network nodes in discrete time steps. This includes that in the time interval $1 \ldots T$, cost appears as the number of packets which are queued and propagated through the whole network. The description of the cost function and the network dynamics leave some questions open. The meaning of the term $r_i(t)$ is unknown, for instance. Additionally, it should be mentioned that if blocking occurs, the necessary propagation time for a packet to travel from an origination node to a destination node can exceed the considered time interval.

The neural net solution shall now shortly be described. The total number of neural nets is $N * T$. Each neural net consists of L layers, and in the generic layer s, n_s neural units are active. In the following, a multivector $\underline{w}$ is constructed whose components are given by all the weight and bias coefficients of the neural net of node i at time t.

The goal is now to determine the vector $\underline{w}^*$ which minimises the expected cost W. The following mathematical description is exactly that, known from the usual back-propagation algebra. The modification here lies in the construction of an adequate multivector, which consists of all weights of all neural nets in the communication system. The gradient descent procedure is then performed in the usual way.

$$\underline{w}^{k+1} = \underline{w}^k - \eta_k \nabla_{\underline{w}} W\{\underline{w}^k, \underline{x}(0)^k, \underline{r}^k\}, \quad k = 0,1,\ldots . \qquad (25)$$

The updating algorithm is subdivided into two parts: a forward pass, where at iteration step k the routing nodes distribute the contents of their queues by means of their local routing strategies; and a backward pass, where the distributed computation is executed throughout the whole communication network for all states $\underline{x}^k$, with the neural nets acting as nodes of an network-wide 'overall' neural net connection structure. Each neural net contributes to the global minimization of W with its local gradient components. The authors report successful simulations of their implemented neural net structure under simplified conditions. Without doubt, it is worth mentioning that the network exhibits a high degree of adaptivity, based on the mathematical foundation of the algorithm. On the other hand, questions concerning real-time use of this approach can only be answered if hardware implementations not only of the neural nets but also of incorporated improvements of minimization techniques exist.

3.2. Broadband Network Admission Control

In high-speed broadband networks such as ATM-oriented systems (Asynchronous Transfer Mode) systems [42], the function of connection admission control (CAC) plays an important role in the resource and network management context. It should help to maximise the system throughput while maintaining the desired quality of service (QoS). The design of CAC is more complex in high-speed systems due to the complexity of bit-rate streams offered by a diversity in traffic sources.

The use of neural nets for admission control was first described in Hiramatsu [43,44], where a back-propagation net has been used. Morris [50] discussed more general aspects concerning the performance of neural net applications. In [51] a stochastic

approximation in conjunction with a back-propagation net has been proposed for network congestion control mechanisms. In [52] different neural net structures for CAC are presented, and a basic architecture is investigated to show the control performance of neural nets in comparisons with conventional CAC mechanisms.

In this subsection we will take the example given in [52] to discuss the use of neural nets in broadband network admission control. According to the CCITT (cf. [42]), the connection admission control function (CAC) is defined as follows: 'CAC is the set of actions taken by the network at the call set-up phase (or during the call re-negotiation phase) in order to establish whether a (virtual channel or virtual path) connection can be accepted or rejected'. We will first illustrate the major problems for CAC arising in modern communication systems. Multi-service networks cope with traffic of different connection types, according to different services offered. Upon acceptance of a connection of type i, this connection will be active during a connection holding period. In this period a connection generates a bit-rate process characterised by means of a few parameters, e.g. the mean bit rate m_i and the peak bit rate h_i. In ATM systems the bit-rate process is packetised in fixed-size units called cells. By accepting a connection the network has a contract with the user: the user agrees to keep the negotiated traffic characteristics during the connection duration, and the network gives a promised quality of service. The CAC has to accept a new connection in such a way that all connections before and after the admission decision of the new connection are treated according to the negotiated quality of service (QoS). In the context discussed here, the quality of service will be the cell blocking probability B_{CELL}.

We now assume a number of M different connection types to be served by the network. The system state seen by the network is $X = \{n_1, n_2, \ldots, n_M\}$, where n_i denotes the number of active connections of type i being in the system. From a mathematical viewpoint, the main CAC function can be seen as a functional mapping of the system state X to a decision vector Z such as $Z = \{z_1, z_2, \ldots, z_M\}$, where $z_i = 1$ stands for the acceptance decision of a connection establishment request of type i, and $z_i = 0$ for the rejection case. The CAC is thus reduced to the implementation of a mapping

$$f_{\text{CAC}} : X \rightarrow Z = f_{\text{CAC}}(X) \tag{26}$$

according to the predefined quality of service of the network. The functional mapping f_{CAC} can further be simplified using the state description $X^* = \{n_1, n_2, \ldots, n_i + 1, \ldots, n_M\}$, i.e. the system state just after accepting the connection request of type i. The decision vector is accordingly reduced to

$$Z^* = \{z_i\} = \begin{cases} 0 & \text{connection } i \text{ should be accepted} \\ 1 & \text{connection } i \text{ should be rejected} \end{cases} \tag{27}$$

and the CAC mapping to

$$f^*_{\text{CAC}} : X^* \rightarrow Z^* = f^*_{\text{CAC}}(X^*) \, . \tag{28}$$

This functional mapping separates the M-dimensional state space into two regions: the *accept region* and the *reject region*. Thus the CAC problem can be formulated as a pattern recognition problem: upon recognition of the load pattern X, a yes/no decision has to be made to accept/reject the connection request. This property leads logically the use of a neural net for connection control purposes in ATM systems.

In most of papers on neural nets for CAC, the feed-forward class of neural nets with the back-propagation learning algorithm is proposed. The general structure is illustrated in Fig. 7. This basic structure was first proposed in [43], and further developed in [44] to the neural net structure depicted

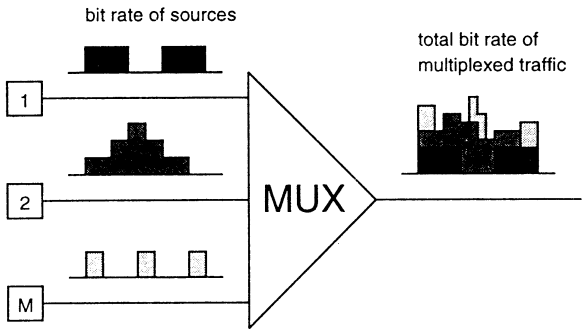

Fig. 6. Source traffic and multiplexed traffic.

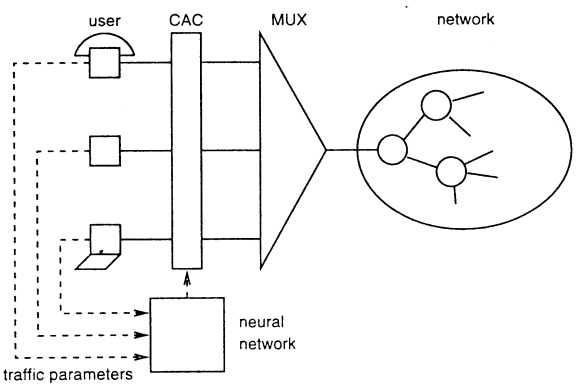

Fig. 7. Neural net for admission control.

in Fig. 8. The neural net is designed to perform the functional mapping given in Eq. (26), i.e. the CAC based on network state. With the M different classes of connections, each with different known characteristics, the pairs of input/output patterns to be learned by the neural net are computed as indicated in Fig. 8. Starting with a state vector $X = \{n_1,n_2, .., n_M\}$ as the input part of a pattern, the multiplexed bit rate function is determined. Having this bit-rate function as traffic stream, the cell blocking probability can, for example, be estimated giving the actual quality of service. Upon a comparison of this measure with the target QoS, the acceptance decision Z is made. This can be interpreted as the decision to be made to accept/reject a connection request of type i if the actual system state is $\{n_1,n_2, .., n_i, .., n_M\}$. The working mode of the neural net during the recall phase is as shown in Fig. 8, where the net will answer with an accept/reject decision Z^* for a connection request of type i when the input vector $X = \{n_1,n_2, ..., n_i + 1, ..., n_M\}$ is presented.

After the learning phase, the neural net performs the CAC by separating the M-dimensional input state space into two regions corresponding to a $(M - 1)$-dimensional decision surface. The decision surface, which separates the 'accept' region from the 'reject' region in the state space, is stored in the weight vectors of the neural net. The performance of the neural net as admission controller (cf. [52]) is illustrated below using the example of an ATM multiplexer and corresponding connection types specified as follows: the output of the multiplexer has a capacity of 600 Mbps, and the buffer space is 0.5 Mb. We take the following types of connection into account:

1. Type 1: on/off, mean m = 10 Mbps, peak h = 40 Mbps, c_x = 1.73.
2. Type 2: binomial, mean m = 5 Mbps, peak h = 40 Mbps, c_x = 0.42.
3. Type 3: binomial, mean m = 5 Mbps, peak h = 80 Mbps, c_x = 0.19, where c_x denotes the coefficient of variation of the bit-rate process.

On connection traffic level, the arrival process of connection requests is assumed to be Poisson. The connection duration is assumed to be negative-exponentially distributed with a mean of 20 s. To compare the performance of the neural net (NN), we take a few CAC methods proposed in the literature: the peak reservation (PR) method, the equivalent bandwidth (EB) method and the weighted variance (WV) method. Figure 9 shows the decision surface of the considered connection admission control methods, which separates the accept and reject regions. The accept region lies on the left hand side of the decision surface. The EB and WV methods have almost the same decision line, which again indicates the similarity of their performance. The NN decision surface is very different. It can be seen that the NN algorithm accepts many more sources with a small mean bit rate (type 2) and fewer sources with a high mean bit rate (type 1) than the EB and WV methods. Furthermore, the NN algorithm is more sensitive with respect to the variability of the bit rate process. From the communication network point of view, this results in the same multiplexer utilisation, whereas from the user's viewpoint, the differences for user groups using connection type 1 or 2 are significant.

For the purposes of comparison, it is also important to know how the CAC mechanism reacts upon temporary overload situations, i.e. the overload control performance of the CAC methods.

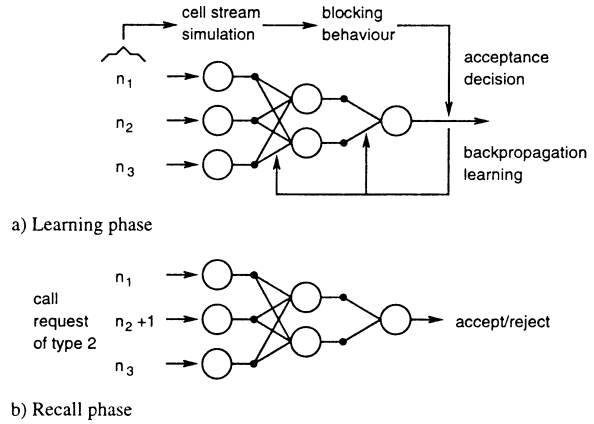

Fig. 8. Working modes of a neural net based admission controller.

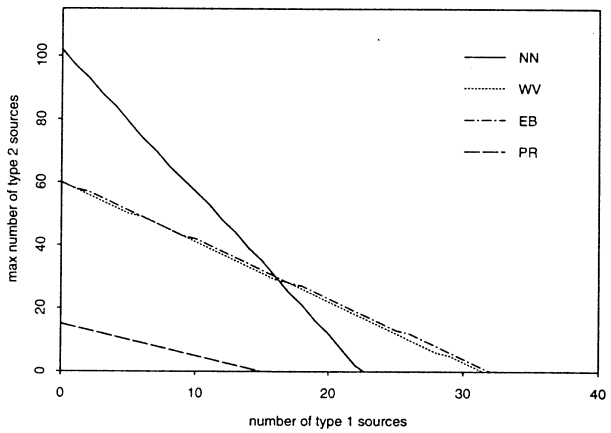

Fig. 9. Decision surface for CAC.

As shown in Fig. 10, a rectangular load pulse is used as the overload pattern, and the time-dependent CAC reaction in terms of cell and connection blocking probabilities is observed. Clearly, a better CAC mechanism should survive the overload phase with a smaller connection blocking probability while keeping the cell loss rate on the same level as under normal load conditions.

A comparison of the non-stationary connection blocking probabilities of connection type 3 is shown in Fig. 10 for the four CAC methods. In this case, it can be seen that the overload performance of the neural net solution is the most efficient. In most load scenarios under consideration, the CAC performance of the neural net structure investigated is comparable with, and in some cases better than, the CAC methods mentioned above, even using a very small and simple neural net. To improve the performance of CAC by neural nets, other neural net structures or other input representations can be developed. One promising candidate is a combined solution of an adaptive neural net, the net having learning patterns which contain more information about the past of the observable load situation.

3.3. Scheduling in Interconnection Networks

The implementation of future high-speed telecommunication networks requires powerful high-capacity switch structures or interconnection networks. Due to the recommendation for ATM to be used in B-ISDN (broadband integrated services digital network), data or information will be partitioned into packets or *cells* (in ATM) and transmitted over the network. This leads to the development of a number of switch structures which handle cells, slots or minipackets as information units. The most common operation mode of appropriate switches is the synchronous mode, i.e. the time axis is slotted and the cells are transmitted from the input to the output lines within one time slot. Important packet switching types under discussion in modern cell-based switching systems are mainly crossbar and banyan switches. Both types require a switch controller to select the cells for transmission within the next time slot, taking into account certain conditions and constraints such as minimising the delay or maximising the throughput of the switch.

The idea for the application of a Hopfield net to scheduling in crossbar switches was first presented by Marrakchi and Troudet in 1989 [53]. They implemented the Hopfield net with a two term energy function in hardware, and showed that the VLSI implementation is able to solve the scheduling task within the given time boundary. Ali and Nguyen [54] took up this proposal and added a third term to the energy function to improve the convergence of the Hopfield net. Brown [55] presented an application of 'winner takes all' circuits to multistage crossbar switches. In Brown and Liu [56], banyan switches are investigated, where winner takes all circuits were applied to avoid internal blocking.

In general, the switch task can be formulated as a scheduling problem. The aim is to maximise throughput while taking into account fairness criteria. Furthermore, in switching systems operating in high-speed environments, the time interval left to run the scheduling task is very short. It must be done during one cell duration; for a 140 Mbps network and a standardised cell size of 53 bytes, the cell duration is about 3 μs. With such a short interval in which to perform the scheduling task, suboptimal solutions can be of great interest.

The studies for crossbar switches and banyan switches are first presented. Consider a crossbar switch with N input and N output lines. The switch is assumed to queue the cells at the input; each input maintains one queue for each output (see Fig. 11). Furthermore, the switch operates in the full-duplex mode, i.e. within one time slot only one

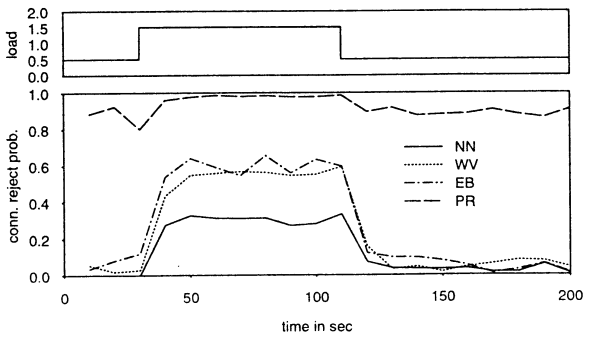

Fig. 10. Comparison of overload performances.

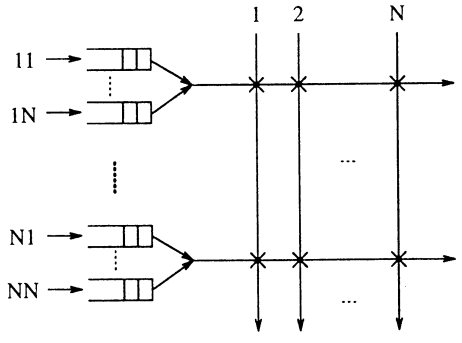

Fig. 11. Crossbar interconnection network.

cell can be transmitted on a particular link in each direction.

A time slot is now observed. All pending transmission requests for this slot can simply be mapped onto a binary matrix $\mathbf{R} = (r_{ij})_{N\times N}$, which will be referred to as the request matrix, whereby $r_{ij} = 1$ indicates that there is at least one cell in queue j at input i waiting for transmission. During this slot, cells will be transmittted according to a schedule matrix $\mathbf{S}$, which has to be determined by the switch scheduler. As mentioned above, this has to be done within a slot duration, which is quite short in high-speed systems. To maximise the throughput of the switch within one slot, the schedule matrix $\mathbf{S}$ must be chosen in such a way that the overlap with the request matrix is maximised. Due to the full duplex transmission mode at most one entry $s_{ij} = 1$ is allowed within each row and each column.

For the switch scheduler, a hardware implementation of a Hopfield neural net is presented in [53]. This net maintains one neuron for each crosspoint of the crossbar switch, and the neurons are only connected by rows and columns via inhibitory connections. The following energy function (without bias term) is assumed:

$$E = -\frac{A}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} v_{ij}\sum_{\substack{k=1\\k\neq j}}^{N} v_{ik} - \frac{B}{2}\sum_{j=1}^{N}\sum_{i=1}^{N} v_{ij}\sum_{\substack{l=1\\l\neq i}}^{N} v_{lj}\,. \tag{29}$$

The first (second) term reaches its global minimum if at most one neuron is turned on within each row (column). The connection strength from one neuron to another within the same row (column) is denoted by A (B). As the authors pointed out, the results obtained for several 8×8 request matrices are optimal for more than 98% of the cases. In all other cases, the Hopfield net achieved a stable final state, but the solutions correspond to non-optimal schedule matrices. The authors of the original publication warranted the omission of a bias term in the energy function by the results obtained.

Note that the energy function also reaches the global minimum in the case of all neuron outputs being zero. Therefore, in [54] the following bias term is added to the energy function given in Eq. (29):

$$+\frac{C}{2}\left(\sum_{i=1}^{N}\sum_{j=1}^{N} v_{ij} - N\right)^2\,. \tag{30}$$

This additional term forces the schedule matrices to be permutation matrices, corresponding to final stable states with N neurons activated. Since the number of cells that can be transmitted without blocking may be lower than N, the schedule matrices do not always have to be permutation matrices. A function to compute this number is presented in [57], but it does not work accurately. Moreover, the scheduling task can be considered as a matching problem appearing in graph theory, and there are efficient methods to solve this class of problems (see, for example, [58]).

The connection strengths are determined by means of a partial differentiation of the energy function. Another method not taken into consideration in this paper is presented in [59]. The authors of [54] simulated the calculation of the switching matrices for 200 input request matrices. During the simulation the throughput was observed, but the portion of the optimal solutions is not reported in detail. Ali *et al.* explored the same neural net controller for switch sizes of 4, 8 and 16, as published in [60]. The throughput of the switch and the mean delay of a packet have been observed. It was shown that the throughput yielded by the neural net controller was closer to an analytically estimated upper bound than the corresponding lower one.

As presented in [61], the switching fabric using the neural net controller was simulated under realistic input traffic assumptions. This study compares the performance of the Hopfield net switching with such fabrics using the *round-robin* strategy and the *Hungarian algorithm*, well-known from graph theory. As major performance measures of the interconnection network, the following metrics are considered:

- the *access delay* D, defined as the time passing from the packet's arrival until the first appertaining cell is transmitted, and
- the *transmission time* T, the time interval from arrival until the packet is completely transmitted.

The simulation results for a simple 8×8 switch can be summarised as follows. The neural net controller yielded optimal throughput for more than 97% of the input request matrices. The remaining switching matrices were not optimal, although the stable states of the Hopfield net always represented global minima of the energy function. This fact is a consequence of the choice of the starting neuron activities. The network always enters the equilibrium state being the local minimum closest to the starting point. This minimum does not always have to be an optimal solution. For that reason, the mean delay and mean transmission times of the neural net controller are usually longer than the corresponding times achieved by the matching algorithm which yields maximal throughput at each time slot. The neural net performance is practically comparable to

the round-robin scheme, and is superior for higher variations of the incoming traffic.

In [61], scheduling fairness issues are also addressed. The highest throughput does not always correspond to a fair schedule. While optimising only the switch throughput, an individual cell could possibly be forced to wait a very long time until it is transmitted. The resulting delay and transmission time of the affiliated packet can be very high. From this viewpoint the round-robin schedule seems to offer an *a priori* fairness. It does not use the whole switch capacity within each time slot, but each {input, output}-pair is selected for transmission at least once every $2N$ slots.

Simulating the Hopfield network requires solving the coupled set of ordinary differential equations given in Sect. 2.2. The most commonly used methods for this task are explicit one-step methods. Since the eigenvalues of the weight matrices are usually very distinct, the set of ode's is referred to as *stiff*, as already stated in Sect. 3.1.2. In spite of the fact that explicit methods are not appropriate for integrating stiff differential equations, the results obtained are very reasonable.

The advantage of crossbar switches is that they are internally nonblocking. Blocking may only occur at the inputs or outputs. The disadvantage is the second order increase in the number of crosspoints, which is very expensive. Therefore, larger switches are built up from smaller crossbars. For example, suppose that the switch consists of three stages: inlet, centre and outlet. Thereby, the inlet and outlet stages are built up from $rn \times n$-crossbars and the centre stage maintains $nr \times r$-crossbar switches. Thus, the number of inputs and outputs is given by $N = r \cdot n$. The switches of one stage are connected to those in the next stage with only a single path. Figure 12 shows such a multi-stage switch for $r = 4$ and $n = 2$. The actual state of such a multistage

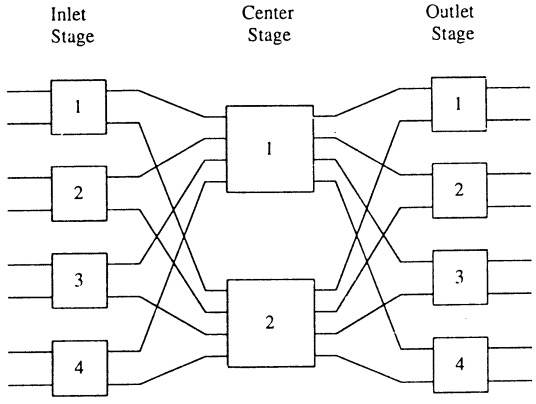

Fig. 12. Three-stage switch.

switch can be described formally by introducing so-called *Paul*-matrices, as presented in [55]. Each row (column) of such an $r \times r$-matrix P represents a particular inlet (outlet) stage. If a request from inlet stage i is routed to the outlet stage j through the centre stage k, then the matrix entry p_{ij} is added the label k. To avoid blocking, the label k is allowed to appear only once within each row and each column, such that each call from an inlet switch is routed through a different centre stage switch to its destination switch. Because a particular centre stage switch can route at most r different calls simultaneously, each entry of the matrix P contains at most r different labels.

Now, the three-dimensional grid structure of the used Hopfield net is obvious. For each element of the matrix, n neurons are maintained with discrete outputs -1 and 1. If there is a neuron with $v_{ijk} = 1$, then a call from inlet i is routed to the outlet j by crossing the centre stage k. The equilibrium states of the neural network have to meet the following constraints: (i) each label appears only once in each row and each column, and (ii) the number of neurons must not be greater than the number of labels for each matrix entry p_{ij}.

Since the multi-stage switch considered is explored for circuit switching communications, the neural network operates as follows. Suppose there is a set of nonblocking calls at an arbitrarily chosen instant. When a newly arriving call cannot be routed through the switch without blocking, the neural network tries to compute a rearrangement of all calls such that the blocking situation is dissolved.

To ensure the convergence of the network to valid solutions, some additional control neurons were added to the original model. It is doubtful whether this extension enforces the neural net to work accurately, because the determination of the connection strengths is fully omitted. This fact is astonishing, since in applying Hopfield networks to optimisation problems, the crucial step is to design a suitable energy function, and then to determine the connection strengths accurately. In the following we present a neural network application to crossbar switches built up from one-sided crosspoint chips, as proposed in [62]. The considered switch providing full-duplex paths maintains N input ports and M internal busses, and is constructed of an $r \times c$-matrix of $n \times m$-chips. Notice that $N = r \cdot n$ and $M = c \cdot m$. Such a switch is depicted in Fig. 13 for $r = c = 2$, $n = 4$ and $m = 2$. A connection between a source and a destination port will be established if any unused internal bus is found to be intersecting with the source and destination port and the crosspoints at the intersection are closed. If the two

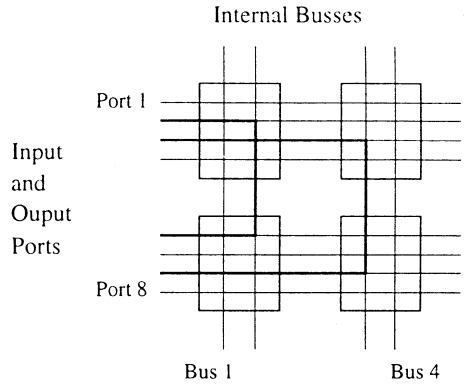

Fig. 13. One-sided crosspoint switch.

crosspoints establishing a connection are on the same chip, no internal bus driver needs to be activated. If the switch operates in a batch mode, the neural net controller selects as many connections as possible, considering the constraints explained later. Using the incremental mode, connections are attempted to be set up at the arrival instants.

The suggested neural net controller has the following architecture. A single neuron is provided for each of the crosspoints, whereby the arrangement of the neurons corresponds to the matrix of crosspoints. The neurons are enumerated from (1,1) to (N,M). A control neuron is added for each row and each column. Setting up a connection between ports i and j implies that there is a column k with exactly two neurons (i,k) and (j,k) having an output of 1. Within the rows i and j, these are the only ones that are on. Suppose that port i is idle, then the output of the row control neuron $(i,0)$ is set to 1, keeping all other neurons off within this row; otherwise, the output is set to 0. The column control neurons should enforce that within each column of the neuron matrix either none or exactly two neurons occur with an output of 1. Further, the neural network is augmented by one neuron for each of the chips, to control the number of active bus drivers on the chip.

The neuron updating formula describing the time-dependent behaviour of the neurons is presented in the study mentioned[1]. Unfortunately, the parameters determining the connection strengths are once again omitted. This neural net has been tested for 100 different request matrices using the following partially linear transfer function: $f(x) = 0$ for $x < 0$, $f(x) = x$ if $0 \leq x \leq 1$, and $f(x) = 1$ if $x > 1$. It reportedly fails to converge for several request matrices, especially for larger switches. Moreover, from the statistical point of view, the number of tests seems to be very small for a sufficient performance evaluation. Another application of neural networks to a banyan network for packet switching is suggested in [56]. This banyan network has N input and N output ports, and consists of $\log_2 N$ stages, each assembled of internally nonblocking 2×2 switching elements. Figure 14 shows such a switch for $N = 8$. The inputs are connected to the first stage of switching elements through a perfect shuffle, since this is well known to increase the performance of a banyan switch due to the avoidance of cell collisions. Banyan switches are self-routing from source to destination, i.e. at stage k a particular cell is sent on the upper outlet, if the k-th bit of the destination address is zero, and *vice versa*. Using this method, every cell arrives at its correct destination. A blocking situation occurs if more than one cell attempts to make use of a particular link.

The links of the perfect shuffle are denoted as the link stage 0, the links between stages k and $k + 1$ are denoted as link stage k, and the output links of the switch as link stage $n + 1$. A_j^k is defined as the set of all {source, destination}-pairs that attempt to use link j of link stage k. Therefore, a set of cells is nonblocking if, within all stages, the sets A_j^k contain at most one $\{s,d\}$-pair. Since the stable states of a winner takes all network contains exactly one active neuron, such a network is provided for each of the sets A_j^k to select a single $\{s,d\}$-pair from it for transmission. Note that output and input blocking can only be avoided by adding queues. The banyan switch considered is supposed to provide input queueing with N queues per input port. Given an input request matrix (such as that known from above), to maximise the throughput of the switch first requires the solving of the corresponding matching problem, as reported above and originally presented in [53]. The determination of the connection's strengths, as well as their

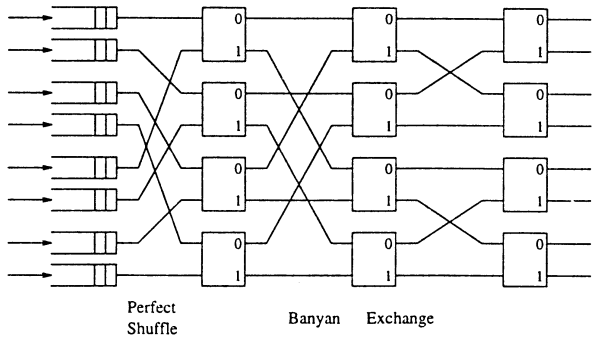

Fig. 14. A banyan switch structure.

[1] For details, see the original literature about Hopfield neural nets concerning this topic.

estimation, is, once again, not reported in this paper. Moreover, there is no information as to whether the network always produces valid solutions or not. Nevertheless, results obtained for several switch sizes up to 32 are presented. The average queue size was observed, and was compared to that achieved for a Batcher–banyan nonblocking switch[2]. Furthermore, distribution of the number of cells in a queue was also recorded to determine the loss probability of cells for a given buffer size.

Finally, we refer to a further application to banyan networks [64]. The algorithm presented there primary initialises a feasible set of conflict-free paths to avoid internal blocking. Then it iteratively tries to augment the number of paths by exchanging neuron outputs. An energy function is defined for the considered problem, but during computation of the best path assignment no neuron updating is applied, such as that known from Eq. (4). Moreover, no connecting weights are estimated, and the neurons are just simple binary variables, so it has to be emphasized that the algorithm presented is not a neural network. Nevertheless, the authors report that the results achieved are better than results achieved by applying another heuristical method.

3.4. Channel Allocation in Mobile Communication Systems

In a number of recent studies [45–47] the use of neural nets in mobile communications, e.g. the adaptive assignment of demanded radio channels to mobile subscribers in a cellular mobile radio system, has been addressed.

3.4.1. Problem statement. A cellular mobile radio system is considered, based on a number of cells, each covering a certain geographical region. In Fig. 15 an idealised cellular structure with hexagons

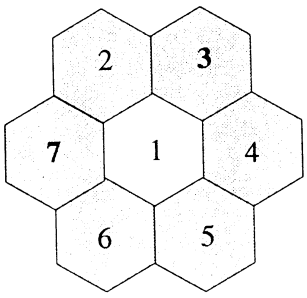

Fig. 15. A cellular mobile radio system with interference area.

[2] For the definition of a Batcher-banyan switch see, for example, [1].

is depicted, though in reality it may have a totally irregular form. In each cell a number of channels is assigned according to the traffic intensity, and other factors. The concept of channel reuse is employed, i.e. if a channel is used in a cell, say cell 1 in Fig. 15, it cannot be used simultaneously in an area, e.g. in the adjacent cells 2 to 7, but it can be used outside the shaded area. Physically, the border of the cells represents the border of the region, where the signal-to-noise ratio (SNR) or the SINAD (signal, noise and distortion) ratio doesn't fall below a threshold underneath which no acceptable radio reception is possible. The traffic intensity in a cellular system is further modulated according to a *handover* function which is caused by customers moving from one cell to another. If a mobile unit leaves the covered region of $BS(i)$ to $BS(j)$, an adjacent cell, the call or connection being served has to be handed over under the condition that the neighbouring cell has a channel frequency left, which of course must be different to those frequencies used in the originating cell. The same problem arises if a call arrives from outer cells, perhaps with a certain number of hops between the originating cell and destination cell. The problem of assigning channels to cells is basically a scheduling problem. A limited number of channels has to be assigned to a finite number of cells following constraints such as avoiding the use of the same channel in a reuse area and two adjacent channels in a cell. The schedule can be of a static or dynamic nature. Due to the actual traffic intensity, which is again time-dependent, a static assignment is not efficient. Conventional algorithms try to take this changing channel demand into account by using splitting techniques of cells at certain times, or by borrowing channels either from neighbouring cells or from an overall available pool of free channels. On the other hand, there is the need to exploit the available spectrum optimally, and so the concept of free channels is critical. The time-dependency of a scheduling algorithm for channel assignment would be the most attractive property, if the probability density function of the traffic in cellular systems is known. Further, due to the mobility of the users, the algorithms should be fast enough to react to real-time traffic changes.

3.4.2. Static channel assignment with Hopfield nets. In [46] and [45] solutions of the channel assignment problem using neural nets of Hopfield type are proposed. In Kunz [45], two main constraints have been considered, which are to be mapped on an appropriate Liapunov energy function:

1. Service of the expected traffic must be accomplished. This concerns the availability of $r(j)$ radio channels at $BS(j)$.
2. The service quality must be sufficiently high. This affects the interference restrictions, the co-channel interference of a channel, which is used by BS and the adjacent-channel interference. This means that the frequencies for different channels used at the same BS should have a certain predefined channel separation.

A Hopfield net with $n \cdot m$ neurons for n base stations and m channels is used. The neurons are arranged in a matrix form. The output of the neuron (i,j) has the value 1 if the channel i is used in $BS(j)$. In the same row, negative couplings (inhibitory connections) of the neurons exist, indicating the avoidance of co-channel interference. In the same column, there are also inhibitory connections, standing for the prevention of adjacent-channel interference. Together with other constraints, the energy function is given as follows (from [46]):

$$E = \frac{1}{2} A \sum_{j=1}^{n} \sum_{j'=1}^{n} \sum_{i=1}^{m} \sum_{|c|<c_{jj'}} (1-\delta_{jj'}\delta_{0c}) v_{ij} v_{i+c,j'} + \frac{1}{2} B \sum_{j=1}^{n} \left(\sum_{i=1}^{m} v_{ij} - r(j) \right)^2 + C \sum_{l=1}^{m} v_l + D \sum_{j=1}^{n} \sum_{i=1}^{m} v_{ij}(1 - v_i) . \quad (31)$$

Here $r(j)$ denotes the number of radio channels at base station j. The first term of the energy function favours states which affect the interference constraints. The minimal allowable channel separation between stations j and j' is denoted by $c_{jj'}$. The second term is responsible for the channel demand for each base station. The third term concerns states with minimal use of assigned channels, and the last term couples the (l,j) neurons with the neuron representing channel l, and reaches a minimum if $l = i$. The connection matrix and the external input obtained from this energy function is thus

$$T_{ij,i'j'} = -A \sum_{|c|<c_{jj'}} (1-\delta_{jj'}\delta_{0c})\delta_{i,i'+c} - B\delta_{jj'}$$

$$T_{ij,l} = D\delta_{il} , \quad T_{ll} = 0$$

$$u_{ij} = Btr(j) - D , \quad u_l = -C . \quad (32)$$

In the study mentioned, simulations results were carried out for real data from the Helsinki region. Furthermore, a fuzzy-logic-based program called GRAND has been involved. The data concerning topographical and morphographical structure were used to define traffic density; after the placement of 25 base stations inhomogeneously over the region, the radio frequency field strength distribution was calculated. Thus the interference matrix (c_{ij}) and the function $r(j)$ were given as input for the neural net, which consisted of 25*121 neurons. This number depends on the maximal degree of the expanded interference graph, which was 121. The results of the simulations given in [45] (and in the former study [46] seem to promise a good performance (73 channels are required to solve the channel assignment problem, and the neural network achieved 78 channels). However, the parameters are left to be interpreted, since it is not completely clear which parameters of the Hopfield net were used, and how they are adapted for other parameter sets.

3.4.3. Dynamic channel assignment using feed-forward neural nets. The use of a feed-forward net in cellular mobile systems for the *dynamic channel assignment* problem (DCA), as explained above, is presented in [47]. As discussed, DCA should make optimal use of the available spectrum, and implements time-dependent adaptation to the traffic situation. The use of conventional graph-theoretic methods is extremely time-consuming, therefore the application of a neural nets promises to be an alternative. An idealisation of a DCA algorithm, called the maximum packing (MP) strategy, was recently proposed and analysed by Everitt *et al.* [48]. This strategy assumes that the minimum number of channels required in the system at a certain time is equal to the actual number of channels in use. The simplest practical implementation of the MP strategy to a cellular system is to use an ordered channel search with no rearrangement.

The neural net used by the authors is of the feed-forward type with back propagation. It was trained off-line, using solutions obtained from an ordered channel search technique. The net is subdivided into two parts, one net for the inner cell and one for the six outer neighbouring cells. The inner cell net contains 42 input neurons, six hidden neurons and three output neurons. The outer cell net differs only by the number of 24 input neurons. The training data consists of the usage of the available channels of a requested cell and that of its neighbours. For outer cells, the training data comprises its own channel usage and that of the three closest neighbours for each cell. In Fig. 16, the basic structure of the system is shown.

The traffic load in a cell was measured by the average number of new calls which arrived in that cell over a certain interval. The performance of the DCA-algorithms using ordered search and neural

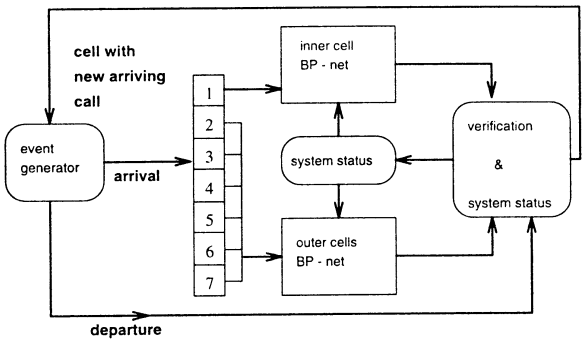

Fig. 16. Structural diagram of the neural network approach.

nets was assessed by measuring the call blocking probability, which showed a good approximation of the ordered channel search by the neural net.

3.5. Applications of Self-organising Feature Maps

Applications of self-organising feature maps to solve problems in telecommunication systems are quite restricted in number. As a Kohonen net provides a topological approximation of the distribution of the input vectors, this type of neural net would be appropriate for the representation of traffic distribution in space and time. Tasks such as traffic management are therefore potential candidates to be processed or controlled by a self-organising feature map. The purpose of traffic management is to best meet the traffic communication requirements of the users under the constraint of fixed network capacity. Consequently, the communication network has to be monitored and analysed constantly to reconfigure the network performance facilities for prompt reaction to traffic changes.

In [49] and [65] an application was presented describing a modified version of the Kohonen algorithm for the management of a digital satellite communications network using the TDMA access technique. TDMA (Time Division Multiple Access) is a multi-access technique often used in satellite communications. Several earth stations are allowed to share the capacity of a satellite by allocating time slots to the earth stations. The station capacity is thus defined by the number of channels which are assigned to it. The TDMA satellite communications network has a meshed topology, allowing station-to-station direct communication. The sum over all capacities of all links connecting the stations with each other is the total fixed network capacity. The load and demand to a network thus define the network state, which is quantified by the number of channels allocated to each link. The capacity of each link should vary according the changing traffic demands. The assignment of channels to links in a network topology is referred to as a *map*.

The problem to be dealt with using a neural net can be posed as the channel assignment task in TDMA satellite communication networks involving a pattern recognition and generation problem. The fluctuation of the network state constitutes a time-varying vectorial pattern, which has in turn to be assigned to a predetermined reference pattern. This pattern represents an existing mapping, which was achieved by human experience or statistical analysis. The actual network state pattern can be obtained using the Common Signalling Channel, e.g. the similarity of the actual pattern to a reference pattern can be proved by means of some distance metric (e.g. the Euclidean metric). Depending upon how close the pattern is matched to each of the reference patterns, one of the following three steps can be taken:

1. Using the original map if the distance between input pattern and reference map is very small.
2. Generating an intermediate map which deviates gradually from the original pattern and corresponds to another reference pattern.
3. Generating and using a new reference pattern, replacing one of the existing reference patterns if the distance is very large.

These three stages depend on the comparison of incoming network state vectors with reference vectors, and so implement the LVQ (learning vector quantization) described in [23]. The generation of new reference maps reflects the variation of traffic flow in time, and therefore a decision method is needed to decide whether a new map should be generated to replace the current reference map. In [65], the following formula is used:

$$d_j(t) = \sum_{i=1}^{L} |d_{ij}(t)| \quad \text{with} \quad j = 1,2, \ldots, N \quad \text{and} \quad i = 1,2, \ldots, L \,, \tag{33}$$

where $d_{ij}(t) = v_i(t) - C_{ij}(t)$, L is the total number of links in a network, and $C_{ij}(t)$ is the total number of channels assigned to link i of the j-th reference map at time t. After normalisation it is possible to determine which reference map best meets the network requirements. This is clearly the one with the smallest normalised distance. The modification of the reference map to achieve an intermediate map is governed by the following equation:

$$C'_{ij}(t+1) = C'_{ij}(t) + \beta(n(j))d'_{ij}(t) \,, \tag{34}$$

with $\beta(n(j))$ as a gain factor depending on the number $n(j)$ of occurrences of reference map j. The convergence factor $E(j) = \max_i|\beta(n(j))d'_{ij}|$, $i = 1,2, \ldots L$, becomes smaller if the j-th map is selected more often, indicating that the modified map j meets the network requirements better than the other maps. A new map is then created based on the following conditions:

$$E \geqslant r \Rightarrow C_{ij}(t+1) = C_{ij}(t)$$

$$E < r \Rightarrow C_{ij}(t+1) = \frac{T}{B} C'_{ij}(t)$$

with $B = \sum_{i=1}^{L} C'_{ij}(t)$ as the total number of channels assigned to the modified map j, r as a threshold set, and T as the capacity of the entire network. If a decreasing function $\beta(n(j))$ is chosen, E can be forced to be smaller than r, and so the condition for the generation of a new map is fulfilled in dependence of the number of selections of a reference map.

In [65] the algorithm introduced in [49] has been extended in such a way that a second-level management was appended, including a state-dependent routing scheme, which tries to route each call in those directions where the blocking probability of future calls is low.

The results in the studies cited showed that self-organising algorithms are appropriate in cases where the traffic in several links is highly variating in time slot transitions. The state-dependent routing algorithms essentially reduce the blocking rate, if the traffic intensity is not too high. The best method was the combination of the two algorithms, where the traffic throughput closely approached the arrival traffic rate.

4. Concluding Remarks

The artificial neural net development has faced a renaissance in the last decade, with a large number of application areas. From the viewpoint of telecommunication networks and systems, an increasing number of studies can be observed in the recent literature dealing with proposed applications of neural nets in telecommunication environments, such as connection admission control in broadband networks, the control of high-speed interconnection networks, channel allocation in cellular mobile systems, adaptive routing, etc.

These applications mainly use three neural net types: feed-forward nets with back-propagation learning, Hopfield feedback nets, and self-organising neural nets. In this paper, we first give an overview of neural net classes and their main properties, and then give a review of applications in telecommunication systems, where attention is devoted to numerical aspects such as the convergence property and learning speed of the proposed neural nets.

It should be noted that, with a few exceptions, the applications discussed here are all in proposal or preliminary phases. To become practicable, issues concerning structural design and dimensioning of the neural nets under consideration must be better understood.

In this early phase of neural net development, the list of applications mentioned in this paper is by no means complete. The theory of neural nets is in fact still in an early phase. With better understanding of processes going on in such massively parallel structures, together with the development of new net architectures and numerical convergence properties which will allow us to dimension such neural nets properly, more and more efficient applications are still to come.

References

1. Alston MD, Chau PM. A decoder for blocked-coded forward error-correcting systems. In: Proceedings IEEE-IJCNN, Washington (DC), 1990; 2: 302–305
2. Chineh TD, Goodman R. A neural network classifier based on coding theory. In: Proceedings of the 1st Neural Information Processing Systems (NIPS), Denver, CO, 1987: 174–183
3. Izquierdo AC, Sueiro JC, Mendez AH. Self-organizing feature maps and their application to digital coding of information. In: Proceedings of the International Workshop on Artificial Neural Networks (IWANN'91), Granada, Spain, 1991: 401–408
4. Jeffries C. Code recognition and set selection with neural networks. Boston: Birkhäuser, 1991
5. Ortuno I, Delgado JA. Neural networks as error correcting systems in digital communications. In: Proceedings of the International Workshop on Artificial Neural Networks (IWANN'91), Granada, Spain, 1991: 409–414
6. Vecchi MP, Salehi JA. Neuromorphic networks based on space optical orthogonal codes. In: Proceedings of the 1st Neural Information Processing Systems (NIPS), Denver (CO), 1987: 814–823
7. Bradburn DS. Reducing transmission error effects using a self-organization network. In: Proceedings IEEE-IJCNN, Washington (DC), 1989; 2: 531–537
8. Bradburn D. Self-organization of non-numeric data sets. In: Proceedings IEEE–IJCNN, Seattle, WA, 1991; 1: 37–41
9. Manikopoulos C, Antoniou G, Metzelopoulou S. ANS classification of finite state machine for high compression video conference coding. Poster presentation at the IEEE-INNC, Paris, France, 1990; 1: 55
10. Matsumoto T, Koga M, Noguchi K, Aizawa S. Proposal for neural-network applications to fibre-

optic transmission. In: Proceedings IEEE-IJCNN, San Diego (CA), 1990; 1: 75–80
11. Gibson GJ, Siu S, Cowan CF. Multi-layer perceptron structures applied to adaptive equalizers for data communications. In: Proceedings of ICASSP, 1989; 1183–1186
12. Koers PHJ, Vogel JA, Zeelen R, van der Putten FG, Berkhout AJ. Active noise reduction using a neural network processing system. In: Proceedings IEEE-INNC, Paris, France, 1990; 1: 145–148
13. Kohonen T, Raivio K, Simula O, Ventä O, Henriksson J. An adaptive discrete-signal detector based on self-organizing maps. In: Proceedings IEEE-IJCNN, Washington (DC), 1990; 2: 249–252
14. Kohonen T, Raivio K, Simula O, Ventä O, Henriksson J. Combining linear equalization and self-organizing adaptation in dynamic discrete-signal detection. In: Proceedings IEEE-IJCNN, San Diego (CA), 1990; 1: 223–228
15. Matthews MB, Moschytz GS. Neural network nonlinear adaptive filtering using the extended Kalman filter algorithms. In: Proceedings IEEE-INNC, Paris, France, 1990; 1: 115–118
16. Nobakht RA, Van den Bout DE, Townsend JK, Ardalan SH. Optimization of transmitter and receive filters for digital communication systems using mean field annealing. IEEE J Selected Areas in Commun 1990; 8(8): 1472–1479
17. Stubbendieck GT, Oldham WJB. Recognition of patterns in electronic communication signals using neural networks. In: Knowledge-based systems and neural networks, eds: Sharda R. et al., Amsterdam: Elsevier, 1991
18. Hopfield JJ. Neurons with graded response have collective computational properties like those two-state neurons. In: Proceedings of the National Academy of Sciences USA 81, May 1984: 3088–3092
19. Hopfield JJ, Tank DW. Neural computations of decisions in optimization problems. Biol Cybern 1985; 52: 141–152
20. Parks PC, Hahn V. Stability theory. (In German.) Berlin: Springer-Verlag, 1981
21. Müller B, Reinhardt J. Neural networks – an introduction. Berlin: Springer-Verlag, 1990
22. Watrous RL. Learning algorithms for connectionist networks: Applied gradient methods of non-linear optimization. In: Proceedings of IEEE International Conference on Neural Networks, 1987; II: S; 619–627
23. Kohonen T. Self-organization and associative memory (2nd ed.). Berlin: Springer-Verlag, 1988
24. Ritter H, Martinetz Th, Schulten K. Neural nets: an introduction to neural computer science of selforganising networks. Reading (MA): Addison-Wesley, 1990
25. Kohonen T, Kangas J, Laaksonen J, Simula O, Ventä O. Variants of self-organizing maps. In: Proceedings IEEE-IJCNN, Washington (DC), 1989; 2: 517–522
26. De Sieno, D. Adding a conscience to competitive learning. In: Proceedings IEEE-ICNN, 1988; I: 117–124
27. Tanenbaum AS. Computer networks. Englewood Cliffs (NJ): Prentice-Hall, 1988
28. Rauch HE, Winarske T. Neural networks for routing communication traffic. IEEE Control Syst Mag 1988: 26–30
29. Zhang L, Thomopoulos SCA. Neural network implementation of the shortest path algorithm for traffic routing in communication networks. Private communication
30. Ali MM, Kamoun F. A neural network shortest path algorithm for optimum routing in packet-switched communications networks. In: Proceedings GLOBECOM, 1991; 0120–0124
31. Fritsch T, Mandel W. Communication network routing using neural nets. Research Report, No. 30, Institute of Computer Science, University of Würzburg, 1991
32. Fritsch T, Mandel W. Communication network routing using neural nets – numerical aspects and alternative approaches. In: Proceedings IEEE-IJCNN, Singapore, 1991; 1: 752–757
33. Jensen JE, Eshera MA, Barash SC. Neural network controller for adaptive routing in survivable communication networks. In: Proceedings IEEE–IJCNN, San Diego (CA), 1990; 2: 29–36
34. Frisiani G, Parisini T, Siccardi L, Zoppoli R. Team theory and back-propagation for dynamic routing in communication networks. In: Proceedings IEEE-IJCNN, Seattle (WA), 1991; 1: 325–334
35. Thomopoulos SCA, Zhang L, DerWann C. Neural network implementation of the shortest path algorithm for traffic routing in communication networks. In: Proceedings IEEE-IJCNN, Singapore, 1991; 3: 2693–2702
36. Grigorieff RD. Numerics of ordinary differential equations. (In German). Teubner, 1977
37. Aiyer SVB, Niranjan M, Fallside F. A theoretical investigation into the performance of the Hopfield model. IEEE Trans Neural Networks (1990); 1(2): 204–215
38. Bertsekas D, Gallager R. Data networks. Englewood Cliffs: Prentice-Hall, 1987
39. Wieselthier JE, Barnhart CM, Ephremides A. The application of Hopfield Neural Network Techniques to Problems of Routing and Scheduling in Packet Radio Networks. NRL Memorandum Report 6730, Naval Research Laboratory, Washington, 1990
40. Hajek B, Sasaki G. Link scheduling in polynomial time. IEEE Trans Information Theory 1988; 34: 910–917
41. Aicardi M, Davoli F, Minciardi R, Toccalino M, Traversa R, Zoppoli R. A neural network approach for adaptive decentralized routing in communication networks. In: Proceedings 29th IEEE Conference on Decision and Control. Hawaii, 1990
42. COST 224 Committee (ed. Roberts J.), COST 224 Final report: Performance evaluation and design of multiservice networks. Paris, France, October 1991
43. Hiramatsu A. ATM communications network control by neural networks. In: Proceedings IEEE-IJCNN, Washington (DC), 1989; 1: 259–266
44. Hiramatsu A. ATM communications network control by neural networks. IEEE Trans Neural Networks 1990; 1: 122–130
45. Kunz D. Practical channel assignment. In: Proceedings of the IEEE, 1990: 652–655
46. Duque AM, Kunz D. Parallel algorithms for channel assignment in cellular mobile radio systems: the neural network approach. In: Parallel processing in neural systems and computers, eds: Eckmiller R, Hartmann G, Hauske G, 1990
47. Chan PTH, Palaniswami M, Everitt D. Dynamical channel assignment for cellular mobile radio system

using feedforward neural networks. In: Proceedings IEEE-IJCNN, Singapore, 1991; 2: 1242–1247
48. Everitt DE, MacFayden NM. Analysis of multicellular mobile radiotelephone systems with loss. Br Telecom Technol J 1983; 1(2)
49. Ansari N, Chen Y. Dynamic digital satellite communication network management by self-organization. In: Proceedings IEEE-IJCNN, Washington (DC), 1990; 2: 567–570
50. Morris RJT. Prospects for neural networks in broadband network resource management. In: Proceedings 13th ITC: Teletraffic and Datatraffic in a period of change, Copenhagen, Denmark, June 1991; 335–340
51. Milito RA, Guyon I, Solla SA. Neural network implementation of admission control. Advances in Neural Information Processing Systems (NIPS), 1991: 4
52. Tran-Gia P, Gropp O. Structure and performance of neural nets in broadband system admission control. Research Report, No. 37, Institute of Computer Science, University of Würzburg, December 1991
53. Marrakchi A, Troudet T. A neural net arbitrator for large crossbar packet-switches. IEEE Trans Circuits and Syst 1989; 36(7): 1039–1041
54. Ali MM, Nguyen HT. A neural network implementation of an input access scheme in a high-speed packet switch. In: Proceedings GLOBECOM, 1989; 1192–1197
55. Brown TX. Neural networks for switching. IEEE Commun Mag 1989: 72–81
56. Brown TX, Kuo-Hui L. Neural network design of a banyan network. IEEE J Selected Areas Commun 1990; 8: 1428–1438
57. Ali MM, Nguyen HT. A neural network controller for a high-speed packet switch. In: Proceedings International Telecommunications Symposium, 1990; 493–497
58. Jungnickel D. Graphen, Netzwerke und Algorithmen. (In German). Bibliographisches Institut, Mannheim, 1990
59. Tagliarini RG, Christ JF, Page EW. Optimization using neural networks. IEEE Trans Comput 1991; 40(12): 533–541
60. Ali MM, Youssefi M. The performance analysis of an input access scheme in a high-speed packet switch. In: Proceedings INFOCOM, 1991; 0454–0461
61. Mittler M, Tran-Gia P. Performance of a neural net scheduler used in packet switching interconnection networks. In: Proceedings IEEE International Conference on Neural Networks, San Francisco, CA, 1993, 2: 695–700
62. Ghosh J, Hukkoo A, Varma A. Neural networks for fast arbitration and switching noise reduction in large crossbars. In: Proceedings of the International Neural Network Conference (INNC) of the IEEE, Paris, France 1990; 1: 270–273
63. Ahmadi H, Denzel WE. A survey of modern high-performance switching techniques. IEEE J Selected Areas in Commun 1989; 7(7): 1091–1103
64. Sun KT, Fu HC. A neural network algorithm for solving the traffic control problem in multistage interconnection networks. In: Proceedings IEEE-IJCNN, Singapore, 1991; 2: 1136–1141
65. Ansari N, Liu D. The performance evaluation of a new neural network based traffic management scheme for a satellite communication network. In: Proceedings GLOBECOM, 1991; 0110–0114