

HOPFIELD NEURAL NETWORKS

5 The present invention relates to an improved Hopfield neural network. Specifically, although not exclusively, the improved neural network may be used to control packet switching in a packet switching network.

10 In a packet switching network, information to be transmitted is digitised and then formed into small addressable packets which are then transmitted over a synchronous network. It is expected that the broad band switching systems of the future will make use of packet switching [1], and although it is generally agreed that such networks will rely on Asynchronous Transfer Mode (ATM), many challenges still exist in realising the
15 necessary high speed packet switching technologies [2,3]. In particular, one of the essential features of such a system is the availability of fast packet switches to route the individual packets reliably and rapidly to
20 their addressed destinations.

 It has been proposed by a number of authors [4] to use neural networks for achieving ATM network control. More particularly, it has been proposed to apply the techniques of neural networks to switching and routing
25 schemes; a general review of this field has been given by Brown [5].

 In a recent study, Ali and Nguyen explored the use of a Hopfield network for controlling a high speed packet-switch [6]. Use of a Hopfield neural network for control
30 of a switch was first proposed by Marrakchi and Toudet [7]. Although Ali and Nguyen demonstrated that the Hopfield dynamical network could be used to obtain high solution accuracy, long simulation times are still a problem. This puts severe restrictions on the size of

the switching system which can be studied. Although the approach used by Ali and Nguyen was an impressive advance, it was to a large extent ad hoc and resulted in proposals for a network which was not only sub-optimal so far as speed of operation was concerned, but which also could not be guaranteed to converge to a valid solution.

Various attempts have been made to improve convergence speed, one technique being to alter the threshold function of the neurons as calculations proceed, to direct operation of the network to the desired result. Such an approach has been used by Honeywell [11]. An alternative approach, used for example by Foo et al [15], is to apply constant biases to certain of the neurons to enforce operation precedence relationships. Although both of these approaches may be of assistance in certain circumstances, they do not address the central problem of preventing the calculations from becoming extremely lengthy and unwieldy, and hence slow, for large networks having many active neurons.

A common aim of researchers in the neural network field is to design a robust network which does not easily get trapped in local minima. Various approaches have been used to improve network robustness, including the use of a transfer function which changes with time in a defined way. Ueda et al [16] discloses the use of a time-varying transfer function which gradually becomes sharper as the calculation proceeds. Neelakanta et al [17] uses a rather similar approach, but based upon the analogy of a gradually decreasing annealing temperature. Cheung [18] and Gsosh [19] disclose other approaches for changing the operation of individual neurons in a defined way as the calculations proceeds.

In order to optimise the type of neural network now

referred to by his name, Hopfield suggested first of all determining an energy function for the problem to be solved, and then comparing that energy function with what is now called the Hopfield Energy Function to determine the weights and biases for the network. Such a method is described in, for example, Hopfield [13] and Chu [14]. The present applicants have found, surprisingly, that using an extension of this technique considerably more information can be obtained to assist in optimising the network.

Neural networks have been applied in a variety of circumstances, and these include routing systems and crossbar switches - see for example Fujitsu [20] and Troudet et al [12].

It is an object of the present invention to provide an improved neural network, based upon the Hopfield model, and particularly for use in operating high speed packet switches (although many other applications may be envisaged).

It is a further object to improve on the work of Ali and Nguyen [6] and to provide a neural network which converges more rapidly to a guaranteed, or virtually guaranteed, valid solution.

It is a further object of the present invention to provide an improved general purpose neural network, based upon the Hopfield model, which will have applications in many technical fields.

According to a first aspect of the present invention there is provided a method of operating a Hopfield network to solve a problem, the solution to which is partially known, the method comprising forcing at least some of the neurons for which the correct values are known to take on the respective correct values, and operating the network to complete the solution to the

This Sheet Has Been Replaced



problem.

5 The invention also extends to apparatus for carrying out the method, and it accordingly extends to a Hopfield network adapted to solve a problem the solution to which is partially known, the network including means adapted to force at least some of the neurons for which the correct values are known to take on the respective correct values, and means for operating the network to complete the solution to the problem.

10 By forcing at least some of the neurons to their known correct values, it has been found that convergence of the calculations is considerably speeded up and the network generally becomes more stable.

15 Typically, where the network is used to solve switching or queuing problems, the individual elements of the solution matrix will converge to either an upper attractor (which may be 1) or a lower attractor (which may be 0). Where solutions are already known for the individual elements of the solution matrix (for example
20 because of some constraints on the problem to be solved, or from other external sources) the relevant entries may be forced to the corresponding attractor.

 According to a second aspect of the present invention there is provided a method of operating a
25 Hopfield network incorporating neurons having a transfer function with a graded response, the method comprising repeatedly updating the neuron outputs according to an updating rule, characterised in that the transfer function has at least one parameter which changes
30 randomly or pseudorandomly between iterations.

 The invention also extends to apparatus for carrying out the method, and accordingly also extends to a Hopfield network incorporating neurons having a transfer function with a graded response, the network including

This Sheet Has Been Replaced



means for updating the neuron outputs according to an updating rule, characterised by means for varying a parameter of the transfer function randomly or pseudorandomly between iterations.

5 Changing the updating rule from iteration to iteration introduces noise into the system, and enables the algorithm to avoid at least some of the non-global minima in which it might otherwise become trapped. This may be achieved by randomly or pseudorandomly changing
10 the constant (β) of a sigmoid function.

According to a third aspect of the present invention there is provided a method of operating a Hopfield network using an updating rule

$$x_{ij}(t) = x_{ij}(t-1) + \Delta t \left(-ax_{ij} - A \sum_{k \neq j}^n y_{ik} - B \sum_{k \neq i}^n y_{kj} + \frac{C}{2} \right)$$

(23)

15 or an equation substantially mathematically equivalent thereto, where:

x_{ij} is an input to the neuron referenced by ij ,

y_{ij} is an output to the neuron referenced by ij ,

20 A , B and C are optimisation values,

and, further, where

x_{ij} and y_{ij} are related by a monotonic activation (transfer) function

$$y_{ij} = f(x_{ij})$$

(1)

25 and C is chosen to be very much smaller than the sum of A and B .

The invention also extends to an apparatus for carrying out the method, and according extends to a Hopfield network adapted to operate using an updating



rule

$$x_{ij}(t) = x_{ij}(t-1) + \Delta t \left(-ax_{ij} - A \sum_{k=1}^n y_{ik} - B \sum_{k=1}^n y_{kj} + \frac{C}{2} \right) \quad (23)$$

or an equation substantially mathematically equivalent thereto, where:

- 5 x_{ij} is an input to the neuron referenced by ij ,
 y_{ij} is an output to the neuron referenced by ij ,
 A , B and C are optimisation values,

and, further, where

- x_{ij} and y_{ij} are related by a monotonic activation
 10 (transfer) function

$$y_{ij} = f(x_{ij})$$

(1)

and C is chosen to be very much smaller than the sum of A and B .

- 15 It is not impossible that the network may still converge to a correct solution if the above inequality is not satisfied, but it has been found that the performance will not be as good. A value of Δt as high as 0.1 or 0.2 or even greater may be used provided that A , B and C are appropriately chosen.

- 20 Preferably, C lies within the range 40 to 150, and A and B (which will often be identical) are at least ten times greater than C (e.g. 20 times or 30 times greater).

The preferred optimisation parameters are $A=B=1250$ and $C=100$. This allows use of $\Delta t=0.2$ or higher.

- 25 Previous work has concentrated upon other ranges for A , B and C . For example, Ali and Nguyen [6] suggest $A=B=100$ and $C=40$, which we consider to be sub-optimal. We have found it is impracticable to determine the values of A , B and C solely by numerical simulation, since for



any realistic application the number of possibilities to be tried is impossibly high. To get around this problem we have developed a new methodical system for determining more closely what the parameters should be, and we have found, surprisingly, that Ali and Nguyen's values were not ideal. By setting the parameters within our preferred new ranges and selecting an appropriate value of β , we have also managed to improve the stability of the neural network, and this enables us to use a very much larger step size ($\Delta t = 0.2$ as opposed to $\Delta t = 10^{-4}$ in Ali and Nguyen [6]).

The preferred application of the present invention is in a telecommunications switch, preferably a packet switch for use in a packet switching system. The practical realisation of the switch will depend very much upon the application, and the limitation of the various technologies involved. VLSI, optics, software and so on could all be exploited.

The neural network could be embodied either in hardware or in software. It could also be split between the two (hybridware).

In a practical embodiment, a packet switch will desirably be associated with a separate queue manager, which sits in front of the switch, and provides prioritisation information to the neural network. If an incoming packet is experiencing an unacceptable delay (or alternatively if an incoming packet is designated as having a high priority), the queue manager may modify the input matrix to the neural network to take account of the desired sequencing. The function of the queue manager could be achieved using conventional, neural, genetic algorithm, fuzzy algorithm or hybrid techniques.

In implementations in which the present invention is used for problem solving other than in telecommunications



switches, it may still be desirable to have an input manager, sitting in front of the neural network, and modifying the input matrix according to known constraints on the problem to be solved. In particular, the input manager will provide the neural network with prioritisation information where the network is undertaking some sort of sequencing task.

The invention may be carried into practice in a number of ways and one specific embodiment will now be described, by way of example, illustrating the use of the present invention in the field of high speed packet switching. The description of the preferred embodiment will refer to the accompanying drawings, in which:

Figure 1 is a schematic diagram of a high speed packet switch;

Figure 2 is a graph showing the Sigmoid function for values of β of 0.08 and 0.16;

Figure 3 shows the simulation results for an 8 x 8 input matrix with optimisation parameters set at $A=B=1250$, $C=700$ and $\beta=0.08$;

Figure 4 shows the values of $f(x)$ for the simulations of Figure 3;

Figure 5 shows the simulation results for an 8 x 8 input matrix with optimisation parameters set at $A=B=1250$ and $C=100$;

Figure 6 shows the values of $f(x)$ for the simulations of Figure 5;

Figure 7 shows the simulation results for an 8 x 8 input matrix with optimisation parameters set at $A=B=1250$ and $C=100$. The figure shows the variation of y when the non-requested neurons are connected;

Figure 8 is a graph corresponding to that of Figure 7 but in which the non-requested neurons are not connected and are consequently excluded from the

calculations; and

Figure 9 shows the relationship of the neural network and the queue manager.

5 The specific embodiment relates particularly to a
neural network for operating a high speed packet switch,
of the type illustrated in Figure 1. This figure is
adapted from a similar drawing in Ali and Nguyen [6].
The purpose of a switch is to ensure that the addressed
10 packets within the system are rapidly routed to their
correct destination, along their respective requested
pathways. A typical switch has a plurality of inputs
and a plurality of outputs, the inputs receiving the
packets to be routed and the outputs being connected to
the various available pathways. Thus, an incoming packet
15 on input 1 may request to be routed via any one of the
switch's outputs. Similarly, packets arriving on any of
the other inputs may request to be routed to any output.

As will be seen from Figure 1, the switch of the
present embodiment is an $n \times n$ switch; in other words
20 there are n inputs and n outputs. Each of the n inputs
has N separate input queues, one for each of the outputs.
Accordingly, an incoming package on input 1 which
requests to be routed to output 1 will be queued in the
first queue of input 1. Other packets on that input
25 requesting to be routed to output 2 would be stored on
the second queue of input 1, and so on. It will be
evident that there are a total of n^2 input queues.

The switch is operated synchronously, and its task
is to transfer the packets from the input queues as
30 rapidly as possible, to the requested outputs. Where
the requests for transfers are arriving more rapidly than
the capacity of the switch, the switch has to choose the
packets from the various input queues in such a way as to
maximise the throughput. In the present embodiment, it

is a neural network which solves this optimisation problem.

Following the method of Ali and Nguyen [6], a Hopfield neural network is used. The general Hopfield energy function is compared with the calculated energy function that will maximise the throughput to give the desired differential equation which is the solution to the switching problem. The differential equation includes undefined constants (optimisation parameters) which Ali and Nguyen determined by simulation. As will be described later, the present invention provides a substantially improved method for determining what the optimisation parameters should be for stable and rapid convergence.

- We will start by considering the basic Hopfield model, and then go on to consider how this can be used to deal with the particular switching problem under consideration.

20 Hopfield Neural Network - Basic Model

The neural network used in this present embodiment is the Hopfield model [8,9], this consists of a large number of processing elements (the neural cells) which are interconnected via neural weights. At any moment in time, each neuron can be described by two continuous variables, the neural activity level x_{ij} and the neural output y_{ij} . These variables are related by the non-linear processing function f , as follows:

$$y_{ij} = f(x_{ij})$$

30

(1)

where f is taken to be some non-linear monotonically

increasing function. This function is called the activation (or transfer) function. The exact form of f is not particularly important, and any appropriate non-linear monotonically increasing function could be used. In the preferred embodiment, however, f is taken to be the sigmoid function

$$f(x) = \frac{1}{1 + \exp(-\beta x)} \quad (2)$$

where β is the gain factor, which controls the steepness of the sigmoid function, as illustrated in Figure 2.

The Hopfield equation describing the dynamics of an individual neuron is given by

$$\frac{dx_{ij}}{dt} = -ax_{ij} + \sum_{k,l=1}^n T_{ij,kl} y_{kl} + I_{ij} \quad (3)$$

where $T_{ij,kl}$ is the weight matrix which described the connection strength between the neurons indexed by (ij) and (kl) . I_{ij} describes the external bias (sometimes referred to as the "external bias current") which is supplied to each neuron.

Hopfield has shown that for the case of symmetric connections $T_{ij,kl} = T_{kl,ij}$ and a monotonically increasing processing function f , the dynamical system (3) possesses a Lyapunov (energy) function which continually decreases with time. The existence of such a function guarantees that the system converges towards equilibrium states. These equilibrium states are often referred to as "point attractors".

The Hopfield-Energy function is of the following form:

(4)

$$E = - \sum_{i,j,k,l=1}^n T_{ij,kl} y_{ij} y_{kl} - \sum_{i,j=1}^n I_{ij} y_{ij} + \sum_{i,j=1}^n \lambda_{ij} \int_0^{f(x_{ij})} f^{-1}(x'_{ij}) dx'_{ij}$$

where the λ_{ij} are positive constants, and x' is the dummy variable over which the integration is made. This equation is set out in Hopfield [8].

It is easily established that with $\lambda_{ij}=a$ for all ij ,
5 the equation of motion (3) can be written as

$$\dot{x}_{ij} = - \frac{\partial E}{\partial y_{ij}} \quad (5)$$

where the dot denotes differentiation with respect to time. From this relation we derive the partial differential equation

$$\dot{E} = - \sum_{i,j=1}^n \left(\frac{\partial E}{\partial y_{ij}} \dot{y}_{ij} + \frac{\partial E}{\partial x_{ij}} \dot{x}_{ij} \right) = -2 \sum_{i,j=1}^n \frac{\partial f}{\partial x_{ij}} \left(\frac{\partial E}{\partial y_{ij}} \right)^2 \leq 0 \quad (6)$$

10

The inequality follows from the fact that the processing function (2) is monotonically increasing. If we drop the integral term in (4) (as is usual in calculations of this sort) the time derivative of the
15 energy function becomes

$$\begin{aligned} \dot{E} = & - \sum_{i,j=1}^n \left(\sum_{k,l=1}^n T_{ij,kl} y_{kl} + I_{ij} \right) \left(\frac{df(x_{ij})}{dx_{ij}} \right) \left(\sum_{p,q=1}^n T_{ij,pq} y_{pq} + I_{ij} \right) \\ & + \sum_{i,j=1}^n a x_{ij} \left(\frac{df(x_{ij})}{dx_{ij}} \right) \left(\sum_{k,l=1}^n T_{ij,kl} y_{kl} + I_{ij} \right) \end{aligned} \quad (7)$$

If we are to obtain convergence, the equation (4) must continually decrease, and hence the right hand side of equation 7 must be less than or equal to 0. In general,
20 this would not be the case, because of the second term on

the right hand side. But in the large gain limit (in other words as the equation (2) tends towards a step function, and β is large) the derivative df_{ij}/dx_{ij} becomes a delta function and therefore $(df_{ij}/dx_{ij})x_{ij}$ tends to zero. This establishes a result discussed by Hopfield [8]. Accordingly, provided that the value of β in equation (2) is appropriately chosen, we can be certain that the system converges and that at equilibrium we have not introduced any inaccuracies by dropping the integral term from equation (4).

There are other approaches to making a Lyapunov function, for example making the decay a zero in the dynamical equation. While this approach has been discussed by Aiyer et al- [10], it is not the preferred approach in the present embodiment and accordingly will not be discussed further.

Formulation of the Switching Problem for Neural Network Solution:

Referring back to Figure 1, it will be recalled that we are concerned with solving an optimisation problem for a square switch of size $n \times n$. We will use the method of Brown, and Ali and Nguyen [5,6] to define the status of the switch at any time as an $n \times n$ binary matrix which will be designated y . Let r_{ij} be the number of packets at input i requesting a connection to the output j . The status of each input queue is given as an initial condition to a neuron; so, for an $n \times n$ switching fabric a total of n^2 neurons are required. The initial conditions of the neurons (in other words the status of

the input queues) are then represented by the matrix

$$y = \begin{pmatrix} y_{11} & \dots & y_{1n} \\ \dots & \dots & \dots \\ y_{n1} & \dots & y_{nn} \end{pmatrix}$$

(9)

Where

$$y_{ij} = \begin{cases} 0 & \text{if } r_{ij}=0 \\ 1 & \text{if } r_{ij} \geq 1 \end{cases}$$

(10)

5 In other words, r_{ij} is unity if a particular input queue is busy, and is zero if it is idle. In this formulation, the rows of the matrix y represent the input lines, and the columns represent the output lines. Every index pair (ij) defines a connection channel.

10 During each time slot, only one packet can be permitted per channel: in other words, during each time slot at most one packet may be sent to each of the outputs, and at most one packet may be chosen from each of the inputs. The task of the neural network is to
15 take the input matrix y , and operate upon it, repeatedly, to produce an output or configuration matrix which actually sets up the channel connections, that is defines the packets that are to be chosen to achieve maximum throughput within the switching constraints.

20 Given the switching constraints, it is clear that the output (configuration) matrix can have at most one non-vanishing element in each row and one non-vanishing element in each column. More than a single element in each row, or a single element in each column, would mean
25 that the switching constraints have been violated in that the switch was either trying to pass two packets at once from a single input or to pass two packets at once to a single output.

30 If the input matrix contains more than one non-vanishing element in any row, or more than one non-vanishing element in any column, then there are more requests for connections than the switch can handle

during that time slot. Since only one input can be connected to one output at any time, the switching mechanism will have to choose only one request and force the rest to wait.

5 To take an example, if there are requests that each of the inputs for packets to be transmitted to all of the outputs, the input matrix y will be

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

10 For that input equation, the output (configuration) matrices that maximise the packet flow through the switch are as follows:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

15 Each of these will be called "valid" solutions because they both satisfy the switching constraints and they also maximise the switch throughput. Other configuration equations such as

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

20 are not valid. The first of these satisfies the switching constraints but does not maximise the throughput because no packet is chosen from the first input even though there is at least one packet waiting. The second matrix violates the switching constraints in

that it attempts to select two packets at once from the first input, and to send two packets at once to the first output.

5 Sometimes, of course, there will not be packets waiting at each of the inputs, and the input matrix y will not be full. In that case, the number of valid solutions will be fewer. To take an example, if the input matrix is as follows:

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

10 The valid output matrices are

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

15 It should be noted that if a row or column of the input matrix contains only zero entries, then in the nomenclature we are using, the resultant output or configuration matrix is valid only if that also contains only zeros in the corresponding rows and/or columns. Otherwise, connections would be made by the switch even through there was no request for such a connection. The following matrix:

20

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

would not be a valid configuration matrix, even though it maximises the throughput of the switch and does not violate the switching constraints, because it sets up a connection between the first input line and the third

output line which was not requested by any incoming packet. The way in which this is achieved, which is one of the novel features of the present embodiment, will be discussed in more detail below.

5

The Energy Function:

From an analysis of the configuration matrices, one can construct an energy (Lyapunov) function for the switching problem. This function can then be compared with the standard Hopfield energy function to find the resulting weight connection matrix and the external biases. From the description in the preceding section it is easily established [6] that an energy function for the switching problem is given by

15

$$E = \frac{A}{2} \sum_{i,j,l=1}^n y_{ij} y_{il} + \frac{B}{2} \sum_{j,i,k=1}^n y_{ij} y_{kj} + \frac{C}{2} \sum_{i,j=1}^n (n - y_{ij})$$

(11)

This energy function takes on zero values only for configuration matrices that are solutions for a full request matrix/input matrix y (that is, for an input matrix which has at least one non-vanishing element in each row and one non-vanishing element in each column. The last term on the right hand side of equation (11) takes on positive values if the input matrix contains one or more zero rows or columns. A comparison with equation (4) (neglecting the integral term) gives the following expression for the weights and biases:

25

$$\begin{aligned} T_{ij,kl} &= -A\delta_{ik}(1-\delta_{jl}) - B\delta_{jl}(1-\delta_{ik}) \\ I_{ij} &= \frac{C}{2} \end{aligned}$$

(12)

Where δ_{ij} is the kronecker delta. Substituting this back into the dynamical equation (3) gives

$$\frac{dx_{ij}}{dt} = -ax_{ij} - A \sum_{k \neq j}^n y_{ik} - B \sum_{k=1}^n y_{kj} + \frac{C}{2}$$

(13)

which is the desired differential equation for the switching problem. Under this dynamical equation the individual neurons will develop towards configurations which minimise the energy function (11). The parameters A, B and C are known as "optimisation parameters", and in previous work [6] these have been determined purely by trial and error. If the optimisation parameters are not chosen carefully, the equation (13) will either not converge at all, or it will converge only slowly. A further possibility is that the equation might converge to a solution which is not "valid" in the sense described above, for example, because it does not maximise the throughput of the switch.

In practice, equation (13) is used in its iterated form as follows:

$$x_{ij}(t) = x_{ij}(t-1) + \Delta t \left(-ax_{ij} - A \sum_{k \neq j}^n y_{ik} - B \sum_{k=1}^n y_{kj} + \frac{C}{2} \right)$$

(13a)

where Δt is the step length.

Establishing the Optimisation Parameters:

An important feature of the present invention relates to the determination of the optimisation parameters. As was indicated above, optimisation has previously been carried out by trial and error, but it is now believed that significant further progress can be made in this field by the use of a more methodical approach.

To gain some initial information about the optimisation parameters A, B and C, we will consider the dynamical equation (13) under conditions of equilibrium, that is when

$$\frac{dx_{ij}}{dt} = 0 \quad (14)$$

then from equation (13), and substituting in equation (2), we find

$$x_{0,ij} = -A \sum_{k \neq j}^n f(x_{0,ik}) - B \sum_{k \neq i}^n f(x_{0,kj}) + \frac{C}{2} \quad (15)$$

where $x_{0,ij}$ is the value of x_{ij} at equilibrium.

To put some further restrictions on the parameters, we should recall that in the final solution to the switching problem each neuron (that is, each entry in the configuration matrix y) will either be zero or one. Given that a valid solution has been found, this means that the matrix will have at most one active unit per row and one active unit per column. We can use this additional information to set further bounds on the optimisation parameters in the region close to equilibrium. If we assume, first of all, that (i,j) is a zero position, then it is straightforward to establish that the equilibrium condition reads

$$x_1 = -A - B + \frac{C}{2} \quad (16)$$

where x_1 denotes the first equilibrium solution. Because we are at equilibrium, we know that the associated y value must be close to zero, and from equation (2) we know that y only tends to zero as x tends to minus

infinity. Accordingly, we can rewrite equation (16) as the following inequality

$$-A-B+\frac{C}{2} \ll 0$$

(17)

In general, there will be n^2-n positions in the network satisfying this condition. This solution may be referred to as the "negative attractor" solution, as it is the equilibrium solution obtained as x tends to minus infinity.

On the other hand, if we assume that the position (i,j) represents a location which is tending towards the value 1, in equilibrium, then the equilibrium condition becomes

$$x_2 = \frac{C}{2}$$

(18)

where x_2 represents the second equilibrium solution. Using equation (2), it is easily seen that y tends to 1 as x tends to infinity, and we can accordingly rewrite equation (18) as the inequality

$$\frac{C}{2} \gg 0$$

(19)

In general, this condition will have to be satisfied at n positions in the network.

The final equilibrium condition mean that n neurons in the network have converged to one of the two attractors and n^2-n neurons have converged to the other attractor. From equations (19) and (17) we find that the parameters have to satisfy the following inequalities

(20)

$$0 < C < 2(A+B)$$

If a symmetric configuration matrix is required, this can be achieved by setting $A=B$, in which case the condition (20) reads

$$0 < C < 4A$$

(21)

5 We have found, in fact, by extensive simulation, that at the limit the magnitudes of x_1 and x_2 are equal. Putting $x_1 = -C/2$ into equation (16) gives revised versions of equations (20) and (21), namely

$$0 < C < (A+B)$$

(20a)

$$0 < C < 2A$$

(21a)

10

In the simulations of the Hopfield model for the switching problem we find that any values of A and C which satisfy equation (21a) give the correct result provided that β and Δt are correctly chosen. This has
15 been confirmed for a large number of input matrices and different matrix sizes.

A practical point is that A is always taken to be very much larger than C . This is due to the fact that a large proportion of the neurons have to approach the
20 negative attractor while only a small number of them will approach the positive attractor. Taking A much greater than C speeds convergence, and allows us to use a large value of Δt .

We can assist convergence further by selecting an
25 appropriate value for β . In practice, we find that β should be greater than about 0.08, because the

approximation in the basic Hopfield model which allows us to neglect the integral term in equation (4) assumes that β is reasonably large.

5 If a small value of Δt were to be acceptable in equation (13a), then the term βx in equation (2) can be very large. To make the limiting values of y reasonably close to 0 and 1 we choose, approximately, βx greater than about 4. Hence, from equation (18):

$$\beta \geq \frac{8}{C}$$

10

(21b)

Since β has to be greater than about 0.08, as discussed above, we can write a general equation for β as

$$\beta = 0.08 + 2 \frac{K}{C}$$

(21c)

15 where K is an arbitrary constant, greater than or equal to zero, and C lies between zero and $2A$ (see equation (21a)).

20 If we require Δt to be larger, we have to be careful not to set β too large, or the model does not converge. Empirically, we find that for Δt greater than about 0.01 we obtain convergence provided that K in equation (21c) does not exceed about 10. A large value of Δt is of course what we are interested in, as this provides more rapid convergence than small Δt .

25

Discussion:

Having established these limitations of the values on the optimisation parameters, we have been able to investigate areas that Ali and Nguyen [6] did not

consider. Ali and Nguyen used the values $A=B=100$ and $C=40$, which they considered quite satisfactory. The present work has established that the values used by them were in fact very substantially sub-optimal. To verify this, we have checked one of the cases ($n=4$) which these authors considered. For the case of $n=4$, there are 65536 possible request matrices, of which only 200 were checked at random by Ali and Nguyen. They found their parameter settings to be acceptable for the 200 particular cases which were checked. Using their parameter settings for A , B and C we have checked all 65536 of the possible cases: it was found that 65392 outputs were correct but 144 outputs were incorrect (in other words, they provided less than optimal throughput). Similar conclusions would apply to larger matrices such as $n=5$ and $n=6$. Ali and Nguyen did not specify a value of β , so we have used our value of 0.08. When we repeated the simulations with β set at 1.00 we found that their values of A , B and C do provide correct convergence for all cases, but convergence is extremely slow. Because of the very large number of possible input matrices for $n=5$ and $n=6$, it is quite out of the question to try all of them with our values, but for the 100000 that we have tried (which were randomly chosen) we have found that using parameter settings $A=B=1250$ and $C=100$ convergence was achieved in typically fewer than 30 time intervals, and in each case the solution has been valid. We cannot absolutely guarantee that with our methodical setting of the parameters the system will never converge to a non-global minimum, but it has never done so in the examples we have investigated so far. Provided that A and C satisfy equation (21a), that Δt is correctly chosen, and that β satisfies equation (21c) we expect that the network will handle all types of input matrix. For rapid convergence,

we take A very much greater than C.

From simulations that we have carried out, we believe that the value of C should typically be in the range of 40 to 150 and that A (which equals B) should have a value of at least one order of magnitude greater than that of C. So, for example, if C is taken to be 100 then A and B should both be greater than 1000. We have found that in practice the exact values of A, B and C are not critical to convergence provided that equation (21a) is satisfied, but A has to be much greater than C for convergence to be rapid. As an illustration of the evolution of the system with $n=8$ (that is, an 8×8 input matrix), reference will now be made to Figures 3 to 6.

Figures 3 and 4 show the results for simulation of an 8×8 input matrix with optimisation parameters set at $A=B=1250$ and $C=700$.

Figure 3 shows how dx/dt varies with time, for each of the n^2 neurons, and Figure 4 similarly shows how the value of y varies with time. It will be seen that in Figure 4 some neurons are stable, but there are a number which are not and that there is no convergence to any final solution. Here, since A is not very much greater than C, one would expect convergence only for small Δt . The value of Δt used ($\Delta t=0.2$) is too large.

In contrast, Figures 5 and 6 are corresponding graphs but for simulations based upon the values $A=B=1250$ and $C=100$. It will be seen that the system attains stability after only about 40 time intervals, and the system rapidly converges to a valid solution, with some of the neurons converging to one and others to zero. Here, Δt has been set to 0.2, as before, but now because A is very much greater than C the system is not unstable at larger values of Δt . Taking A very much greater than C allows us to use large Δt and hence obtain more

rapid convergence than was possible in the prior art.

We have found by experiment that when no equilibrium condition is achieved this is often due to the fact that sometime during the evolution of the system one of the two separate attractors will disappear. By choosing A very much greater than C, we ensure the existence of two attractors throughout the time evolution of the dynamical equation. In this way, we have found that oscillations and unstable behaviour of the network is avoided.

Avoidance of Local Minima:

In any system in which the equations of motion are determined by a continually reducing energy function, there is a risk that the system may become trapped at a local minimum of the energy function. In the present embodiment, that would be equivalent to convergence upon a solution which satisfied the switching constraints but was merely a local maximum for the switch throughput, and not the global maximum. Where there is no foolproof method of avoiding convergence onto local minima, we propose that in practice some "noise" should be introduced into the system to avoid spurious convergences as far as possible. We achieve this, as the neural network is calculating its solution, by randomly varying the value of β in equation (2). In the first iteration, the value of β is set at 0.08, and in subsequent iterations the value of β is randomly chosen to lie somewhere within the range 0.08 to 0.16. β , of course, is the gain factor which controls the steepness of the sigmoid function, as illustrated in Figure 1. Where β is taken at a value of greater than 0.08 (see equation 21(c)), the maximum amount of noise can be taken to be equivalent to the value of β .

Imposed Attractors:

A network to be used for switching purposes will normally be sparsely connected. This is easily seen as the first term on the right hand side of equation (12) is non-zero only if $i=k$ and j does not equal k . This gives $n^2(n-1)$ connections. The second term has equally many non-vanishing contributions. The connection matrix (12) therefore defines $2n^2(n-1)$ connections. This has to be compared with the maximum possible number of connections that a network of n^2 neurons can have: that is $n^4 - n^2$.

As has previously been mentioned, if the input matrix contains a null row or a null column, it is desirable that the output matrix should also contain a corresponding null row or column, since otherwise the neural network will have introduced connections where none have been requested. The energy function (11) and the connection matrix (12) do not guarantee this to be the case. This follows from the fact that the energy function (11) does not take on minima if one of the rows or columns have only vanishing entries. To avoid this occurrence, we propose that the null rows and columns should be decoupled from the Hopfield dynamics. We have found in practice that this both improves the convergence of the other neurons, and also permits considerable increases in speed to be achieved.

The null rows and columns may be decoupled from the Hopfield dynamics in a number of ways. One simple mechanism would simply be not to include the null rows and columns at all in any of the calculations. An alternative, and preferred, method is to force all of the neurons in a null row or a null column onto the negative attractor. What we do in practice is to decouple the input to the null rows and columns from all the other neurons. However, their constant outputs are still

coupled. Hence the forced neurons remain fixed at all times at the negative attractor, but their outputs are still fed into the rest of the calculations, and will affect time evolution of the unforced neurons. The external biases of all the forced neurons are set to zero.

The effect of decoupling the null rows and columns will be illustrated by reference to Figures 7 and 8. These show the results for simulations with an 8x8 input matrix, with optimisation parameters set at $A=B=1250$ and $C=100$. The input matrices in each case had non-vanishing elements in the first and last rows, while all other elements were zero. It is obvious that this type of input will require only two connections.

Allowing all the neurons to change freely according to the dynamical equation produced the output shown in Figure 7. This shows that $n=8$ neurons are connected, in other words that the network has artificially connected calls which are not required by the initial input matrix. However, if the zero request neurons are disconnected from the dynamics of the system by an imposed attractor, as previously described, then the network will converge to a valid solution as shown in Figure 8.

If we call the imposed attractor x_3 , to distinguish it from the positive and negative attractors x_1 and x_2 of equations (16) and (18), we can use an equivalent to equation (16) to provide us with some idea of what the imposed attractor should be to ensure that the respective neuron always converges on zero. Since $A=B$ and A is very much greater than C , we have from equation (16) the approximate equation

$$x_3 = -2A$$

(22)

In fact, we find that in practice the exact size of the imposed attractor does not particularly matter, provided that it is large enough and negative enough.

5 As an extension, it would be possible to use imposed attractors not only for the null rows and the null columns, but also for each individual neuron in respect of which no connection request has been made. If there is no connection request, an ideal solution will never cause a connection to be set up, and it is possible
10 effectively therefore to decouple all of the zeros in the initial input request matrix, and to run the neural network only upon the entries which are initially non-zero. This further improves the speed of convergence.

Queue Manager:

In a practical embodiment of the present invention it may be necessary for particular arrangements to be
5 made to deal with incoming packets that are experiencing unacceptable delays, are required to have a particularly high priority, or other specific sequencing requirements. In the preferred embodiment, therefore, there would be a queue manager which would sit in front of the Neural
10 network and provide prioritisation labels or arrangements for the incoming traffic. This is shown schematically in Figure 9:

If an incoming packet is experiencing an unacceptable delay (or alternatively if a critical packet
15 has been designated as having a high priority), the queue manager will modify the input matrix (9) to take account of the desired sequencing. When the queue manager receives a packet from the input line, it will examine the destination address to determine the appropriate
20 destination output line. It will also determine the priority and/or delay of that particular packet. Based

upon the information it has read, it will then appropriately update the input matrix (9), so changing the initial conditions that the neural network has to operate upon. It may also impose attractors, either
 5 positive or negative, or in other ways decouple certain neurons from the network according to the requested priorities of the individual packets.

The function of the queue manager could be achieved using conventional, neural, genetic algorithm, fuzzy
 10 algorithm or hybrid techniques.

One particular function that might be provided by the queue manager would be to adjust the input matrix (9) to take account of the fact that there may be more than one packet which is waiting for the particular
 15 connection. If a large number of packets start to build up, all waiting for a particular connection, the queue manager should have some mechanism for effectively increasing the priority of those packets to ensure that the queue does not become unacceptably long. An
 20 alternative method of achieving the same result might be to make use of an input request matrix in which each element is not merely 0 or 1, but is an integer representing the number of packets that are awaiting that particular connection. The higher the number waiting for
 25 a particular connection, the greater would be the initial value of y , according to equations (1) and (2), and accordingly the greater likelihood there would be of that particular neuron converging to the value 1.

30

Calculation Procedure:

Finally, we will turn to the computational procedure to be used in operating the neural network. The procedure will effectively be the same, whether the

network is embodied in hardware or in software.

The procedure is as follows:

1. Receive the input request matrix, which contains 0's and 1's in the simplest case, indicating which connections are requested and which are not.
2. To centre the request matrix on 0, convert the 0's to $-2A$'s (the negative attractor).
3. [Optional] Use the queue manager to make any appropriate amendments to the resultant matrix, for example to allow for higher priority on certain requests. Higher priority would be achieved by replacing a particular 1 entry with a higher number. At the same time, the queue manager will set up any imposed negative attractors that may be required by changing the corresponding entries to large negative numbers, (eg $-2A$), and at the same time will arrange for those neurons to be decoupled from the others (as explained above) before calculations start.
4. The modified input matrix is then replaced by a neuron matrix using equation (2) (with the non-requested or forced elements remaining at the negative attractor, $-2A$). The elements of this neuron matrix are the y_{ij} . The parameter β is set randomly within the range 0.08 to 0.16. In the first iteration of step 4, it is set at $\beta=0.08$.
5. Iterate the differential equation (13a) (which contains the parameters A, B and C) to calculate new values of x_{ij} . The step length Δt may be, for example, 0.2.
6. Go to step 4, unless the system has converged, that is unless the y_{ij} have not changed by a given amount since the last iteration. If all the neurons have converged, stop.

Practical Realisation:

5 The network of the present invention could be embodied either in hardware or in software, or in a combination of the two. Potential hardware technologies would include VLSI, optics and so on. The environment in which the network is to be embedded will of course play a key role in the determination of the medium. For example, if speed is of the essence, a hardware implementation would appear preferable, but of course
10 that has to be offset against the fact that hardware implementations for large switches would be exceedingly complicated. A specification for a hardware (electrical) realisation of a Hopfield neural network has already been published by Brown [5]. - The various parameters in the
15 Hopfield model can be related to values of the various electrical components in the circuitry. -

 It is expected that the present invention will have application in very many different fields, and in particular to any problem in which a Hopfield energy can
20 be calculated, and there is a requirement for an input matrix having at most one non-vanishing element per row and at most one non-vanishing element per column, in other words where the problem is equivalent to the Travelling Salesman Problem. Potential application areas
25 include network and service management (including switching of lines, channels, cards, circuits, networks etc.); congestion control; distributed computer systems (including load balancing in microprocessor systems, cards, circuits etc.) and decentralized computer systems;
30 work management systems; financial transaction systems (in banking, stockmarkets, shops etc.); traffic scheduling (airlines, trains, underground, shipping etc.); alternated production lines; reservation, storage and cargo systems (including airline, stock etc.);

general scheduling and resource allocation problems;
 general channel assignment problems; fluid control
 systems (including oil, gas, chemical etc.); general
 control systems (including nuclear, cars, airlines,
 5 transport systems etc.); system management systems
 (including satellite, mobile, audio etc.); robotic
 systems dealing with task assignment; task assignment in
 general; real time data acquisition and analysis
 systems; and queuing delay procedures.

10

The continuous case

The present work can be extended to the continuous
 case, where the inputs to the network can take on any
 value within a given range, rather than being restricted
 15 to 0 and 1. One particular way of doing this, as
 previously described, would be to allow the inputs to be
 any positive integer, for example an integer
 corresponding to the number of packets awaiting switching
 in the respective queue. Alternatively, the inputs could
 20 be truly continuous (not just stepped).

The inputs may be multiplied by a spread factor f ,
 prior to the network calculations being started, to vary
 the range that the input values span. Because the
 network calculations are non-linear, altering the input
 25 range may have profound effects on the operation and
 speed of convergence of the net.

In the truly continuous case the addition of noise
 is to be avoided. Imposed attractors may be used but,
 30 in practice, it has been found that they do not add very
 much speed to the speed of convergence of the net in many
 cases.

The continuous case can be considered for performing
 first order task assignment in a multi-service or

heterogenous environment by maximising the sum of indicators of suitability of tasks for the chosen resources. The network can also be used for higher order task assignment taking into account, amongst others, intertask communication costs. Application areas include, amongst others, network and service management, distributed computer systems, systems for work management, financial transactions, traffic scheduling, reservation, storage, cargo handling, database control, automated production, and general scheduling, control or resource allocation problems.

REFERENCES

- [1] A Pattavini, 'Broadband Switching Systems: First Generation', European Transactions on Telecommunications, Vol 2, No 1, p75, 1991.
- [2] M Listanti and A Roveri, 'Integrated Services Digital networks: Broadband Networks', European Transactions on Telecommunications, Vol 2, No 1, p59, 1991.
- [3] CCITT, Draft Recommendation I.150: B-ISDN ATM Aspects, Geneva, January 1990.
- [4] A Hiramatsu, "ATM Communications Network Control by Neural Networks", IEEE Transactions on Neural networks, Vol 1, No 1, p122, March 1990.
- [5] T X Brown, 'Neural Networks for Switching from E C Posner, Ed', 'Special Issue on Neural Networks in Communications', IEEE Communications Magazine, p72, November 1989; see also A Maren, C Hartson and R Rap, 'Handbook of Neural Computing Applications', Academic Press, London, 1990.
- [6] M M Ali and H T Nguyen, 'A Neural Network Controller for a High-Speed Packet Switch' Proc. Int. Telecommunications Symposium 1990, pp. 493-497.
- [7] A Marrakchi and T Troudet, 'A Neural Network Arbitrator for Large Crossbar Packet Switches', in IEEE Transactions on Circuits and Systems, Vol 36, No 7, p1039, 1989.

[8] J J Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons. Proc. Natl. Sci. USA Vol.81, pp 3088-3092, Biophysics, May 1984.

[9] A good discussions of Hopfield networks and further references is given in J Hertz, A Krogh and R G Palmer, 'Introduction to the Theory of Neural Computation', Addison-Wesley, Redwood City, 1991.

[10] S.V. Aiyer, M. Niranjan and F. Fallside, A Theoretical Investigation into the performance of the Hopfield Model. IEE Transactions on Neural Networks, Vol 1, No-2, June 1990.

[11] Honeywell Inc, EP-A-0 340 742.

[12] Troudet et al, IEEE Transactions on Circuits and Systems, vol. 38, no. 1, January 1991, New York, US, pages 42 - 56.

[13] J J Hopfield, US-A-4 660 166

[14] P P Chu, IJCNN-91: International Joint Conference on Neural Networks, vol. 1, 8 July 1991, Seattle, USA, pages 141 - 146.

[15] Y S Foo, IEEE International Conference on Neural Networks, vol. 2, 24 July 1988, San Diego, USA, pages 275 - 282.

[16] Ueda et al, IJCNN International Joint Conference on Neural Networks, vol. 4, 7 June 1992, Baltimore, USA, pages 624 - 629.

[17] Neelakanta et al, Biological Cybernetics, vol. 65, no. 3, September 1991, Heidelberg, Germany, pages 331 - 338.

5 [18] Cheung et al, IECON89, 15th Annual Conference of IEEE Industrial Electronics Society, vol. 1, 6 November 1989, Philadelphia, USA, pages 759 - 763.

10 [19] Ghosh et al, 1993 IEEE International Conference on Neural Networks, vol. 1, 28 March 1993, San Francisco, USA, pages 359 - 364.

[20] Fujitsu Limited, EP-A-475 233.

15

CLAIMS

- 5 1. A method of operating a Hopfield network to solve a problem, the solution-to which is partially known, the method being characterised by forcing at least some of the neurons for which the correct values are known to take on the respective correct values, and
10 operating the network to complete the solution to the problem.
2. A method as claimed in Claim 1 in which the inputs to the forced neurons from the unforced neurons
15 are decoupled during operation of the network.
3. A method as claimed in Claim 1 or Claim 2 in which the outputs from the forced neurons remain coupled to the unforced neurons during operation of the network.
20
4. A method as claimed in any one of the preceding claims in which the problem is mathematically equivalent to the Travelling Salesman Problem.
- 25 5. A method as claimed in Claim 4 in which, when the values of the neurons are represented by a matrix, the rows and columns which are known to be null in the solution to the problem are forced to remain null.
- 30 6. A method as claimed in Claim 4 in which, when the values of the neurons are represented by a matrix, the matrix entries which are known to be null in the solution to the problem are forced to remain null.



7. A method as claimed in any one of the preceding claims in which the said correct values are all identical, and form one attractor of the network.

8. A method as claimed in Claim 7 in which the network has exactly two attractors, an upper attractor and a lower attractor, each of the unforced neurons being arranged by an operating algorithm of the network to converge on either the upper or the lower attractor.

9. A method as claimed in Claim 8 in which the forced neurons are forced to remain at the lower attractor value.

10. A method as claimed in Claim 8 in which the forced neurons are forced to remain at the upper attractor value.

11. A method as claimed in any one of the preceding claims in which the starting inputs to the neurons may take any value within a continuous range of values.

12. A method as claimed in Claim 11 in which the starting input values are multiplied by a spread factor f before the network is operated.

13. A method of operating a Hopfield network incorporating neurons having a transfer function with a graded response, the method comprising repeatedly updating the neuron outputs according to an updating rule, characterised in that the transfer function has at least one parameter which changes randomly or pseudorandomly between iterations.





14. A method as claimed in Claim 13 in which the parameter is a non-monotonic function of the iteration number.

15. A method as claimed in Claim 12 or Claim 14 in which the transfer function is defined by $y=f(x)$, or a function substantially mathematically equivalent thereto, where x is the neuron input, y is the neuron output, and

$$f(x) = \frac{1}{1+\exp(-\beta x)}$$

(2)

β being the said parameter.

16. A method as claimed in Claim 15 when dependent on Claim 16 in which β is chosen randomly within the range between about 0.08 and 0.16.

17. A method of operating a Hopfield network using an updating rule

$$x_{ij}(t) = x_{ij}(t-1) + \Delta t \left(-ax_{ij} - A \sum_{k=1}^n y_{ik} - B \sum_{k=1}^n y_{kj} + \frac{C}{2} \right)$$

(23)

or an equation substantially mathematically equivalent thereto, where:

x_{ij} is an input to the neuron referenced by ij ,

y_{ij} is an output to the neuron referenced by ij ,





A, B and C are optimisation values,
and, further, where

x_{ij} and y_{ij} are related by a monotonic activation
function

$$y_{ij} = f(x_{ij})$$

5

(1)

characterised in that C is chosen to be very much smaller
than the sum of A and B.

10

18. A method as claimed in Claim 17 in which C lies
between about 40 and 150.

19. A method as claimed in Claim 17 or Claim 18 in
which A is at least ten times greater than C.

15

20. A method as claimed in any one of Claims 17 to
19 in which B is at least ten times greater than C.

20

21. A method as claimed in any one of Claims 17 to
20 in which the network is operated with A set equal to
B.

25

22. A method as claimed in any one of Claims 17 to
21 in which A, B and C are substantially as follows:
A=B=1250; C=100.

23. A method as claimed in any one of Claims 17 to
22 in which the activation function $f(x)$ is defined by

$$f(x) = \frac{1}{1 + \exp(-\beta x)}$$

(2)

30

24. A method as claimed in Claim 22 in which the



* C L N S *



value of βC is approximately 4 or greater.

25. A method as claimed in Claim 23 or Claim 24 in which β satisfies the equation

$$\beta = 0.08 + 2 \frac{K}{C}$$

5

where K is an arbitrary constant, greater than zero.

26. A method as claimed in Claim 25 in which K is less than about 10.

10

27. A method as claimed in any one of Claims 17 to 24 further including setting an initial condition for those neurons for which the correct solution is known, to hold the said neurons at their respective correct values.

15

28. A method as claimed in Claim 24 in which the initial condition is

$$x_i = -A - B + \frac{C}{2}$$

(16)

20

29. A method as claimed in Claim 27 or Claim 28 in which the said initial condition is

$$x_i = \frac{C}{2}$$

(18)

25

30. A method as claimed in any one of the preceding claims including the step of amending the





initial conditions prior to operating the Hopfield network, in dependence upon constraints on the acceptable solutions.

5 31. A method as claimed in Claim 28 in which the said step comprises multiplying the initial values of the neurons by a spread factor f , not equal to one.

10 32. A method as claimed in Claim 28 in which the amending step is carried out by a further neural network.

15 33. A method of operating a telecommunications switch using a method of operating a Hopfield network as claimed in any one of the preceding claims.

 34. A method of operating a telecommunications switch as claimed in Claim 33 in which the switch is a packet switch.

20 35. A neural network operated by a method as claimed in any one of the preceding claims.

 36. A telecommunications switch incorporating a neural network as claimed in Claim 35.

25 37. A telecommunications switch as claimed in Claim 36 including a queue manager adapted to amend the initial conditions, prior to operating the neural network, in dependence upon the priorities of calls waiting to be
30 switched.

 38. A telecommunications switch as claimed in Claim 36 or Claim 37 in which the switch is a packet switch.





39. A telecommunications network incorporating a switch as claimed in any one of Claims 36 to 38.

5 40. A Hopfield network adapted to solve a problem the solution to which is partially known, the network being characterised by means adapted to force at least some of the neurons for which the correct values are known to take on the respective correct values, and means for operating the network to complete the solution to the
10 problem.

41. A Hopfield network incorporating neurons having a transfer function with a graded response, the network including means for updating the neuron outputs according to an updating rule, characterised by means for varying
15 a parameter of the transfer function randomly or pseudorandomly between iterations.

20 42. A Hopfield network adapted to operate using an updating rule

$$x_{ij}(\tau) = x_{ij}(\tau-1) + \Delta \tau \left(-ax_{ij} - A \sum_{k=1}^n y_{ik} - B \sum_{k=1}^n y_{kj} + \frac{C}{2} \right) \quad (23)$$

or an equation substantially mathematically equivalent thereto, where:

25 x_{ij} is an input to the neuron referenced by ij ,
 y_{ij} is an output to the neuron referenced by ij ,
 A, B and C are optimisation values,
 and, further, where

30 x_{ij} and y_{ij} are related by a monotonic activation function

(1)





$$y_{ij} = f(x_{ij})$$

characterised in that C is chosen to be very much smaller than the sum of A and B.

43. A method of operating a Hopfield network substantially as specifically herein described with or without reference to any one or compatible combination of Figures 1, 2, 5, 6, 7, 8, 9.

44. A packet switch substantially as specifically herein described with or without reference to any one or compatible combination of Figures 1, 2, 5, 6, 7, 8, 9.

45. A method of optimising a Hopfield neural network comprising:

- (a) determining an energy function for the problem to be solved;
- (b) comparing the said energy function with the Hopfield energy function

$$E = - \sum_{i,j,k,l=1}^n T_{ij,kl} y_{ij} y_{kl} - \sum_{i,j=1}^n I_{ij} y_{ij} + \sum_{i,j=1}^n \lambda_{ij} \int_0^{f(x_{ij})} F^{-1}(x'_{ij}) dx'_{ij}$$

(4)

to determine the weights and biases for the network; characterised by:

- (c) substituting the weights and biases into the Hopfield dynamical equation

$$\frac{dx_{ij}}{dt} = -ax_{ij} + \sum_{k,l=1}^n T_{ij,kl} y_{kl} + I_{ij}$$

(3)

to give a differential equation describing the



expected time-development of the network;

- (d) deriving information about any undetermined optimization constants in the said differential equation by considering the equilibrium condition:

$$\frac{dx_i}{dt} = 0$$

5

(14)

- (e) deriving further information about the optimization constants by separately considering (i) equilibrium for those neurons which have converged to an upper attractor; and (ii) equilibrium for the neurons which have converged to a lower attractor.

10

46. A method as claimed in Claim 17 in which the starting inputs to the neurons may take any value within a continuous range of values.

This Sheet Has Been Replaced



* C L H S *