Routing in computer networks using artificial neural networks

S. Pierre\textsuperscript{a,*}, H. Said\textsuperscript{b}, W.G. Probst\textsuperscript{c}

\textsuperscript{a}Mobile Computing and Networking Research Laboratory (LARIM), Department of Electrical and Computer Engineering, École Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville, Montréal, Québec, Canada H3C 3A7
\textsuperscript{b}Canada Customs and Revenue Agency, 875, Heron Road 3160, Ottawa, Ont., Canada K1A 0L5
\textsuperscript{c}Department of Computer Science, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal, Québec, Canada H3C 3P8

Received 6 December 1998; revised 4 February 2000; accepted 15 June 2000

Abstract

This paper proposes a heuristic approach based on Hopfield model of neural networks to solve the problem of routing which constitutes one of the key aspects of the topological design of computer networks. Adaptive to changes in link costs and network topology, the proposed approach relies on the utilization of an energy function which simulates the objective function used in network optimization while respecting the constraints imposed by the network designers. This function must converge toward a solution which, if not the best is at least as close as possible to the optimum. The simulation results reveal that the end-to-end delay computed according to this neural network approach is usually better than those determined by the conventional routing heuristics, in the sense that our routing algorithm realizes a better trade-off between end-to-end delay and running time, and consequently gives a better performance than many other well-known optimal algorithms.

© 2000 Elsevier Science Ltd. All rights reserved.

Keywords: Computer networks; Flow assignment; Heuristic method; Neural method; Routing algorithm; Topological design

1. Introduction

As illustrated in Fig. 1, a typical packet switching network consists of a set of nodes connected by communication links [5,7,11,18,35]. The term node refers to terminals, printers, servers or hosts, and switches. A reliable communication network should permit two hosts to communicate with each other, as well as to tolerate certain breakdowns of nodes and links, with a possible reduction in performance [15,27,31,42].

In a packet switching network, communication between two hosts generally takes place in the following manner: the transmitting host delivers to a node a block of data, called a packet, which is addressed to the destination host. The latter forwards the packet to an intermediate switching node, which in turn routes the packet to another, and so on, in like fashion until the packet finally reaches the linking node of the destination host. The time taken by a typical packet to travel from the source node to the linking node of the destination host is called the end-to-end delay of the network. The nodes through which the packet is transmitted from source to destination constitute a path or a route; the mechanism used to select one route from various alternatives to link each source–destination pair is called a routing strategy.

The objective of a routing strategy is essentially to minimize the mean delay of the packets in a network, subject to some reliability or capacity constraints [1,2,20,21,38]. There exist some exact mathematical programming methods which are based on certain properties of the mean delay function to solve the problem of optimal routing [6,10,26]. The implementation of these methods necessitates complex and lengthy calculations. As a result, heuristic routing procedures have been used in order to determine, within reasonable computation time, the routes along which the packets must travel without causing network congestion [2,21,23–25].

Routing procedures may be classified into three categories: static routing, adaptive routing, and optimal routing. Static routing consists of defining the paths to be followed by the various packets based on the general characteristics of the network, such as the topology and anticipated mean traffic on the communication links. The traffic matrix gives the average number of packets per second exchanged between each node pair of the network. The routing procedures determine the link flow \( f \) which refers to the effective number of bits per second (bps) carried by a link; the link capacity \( C \) denotes the maximum number of bits per second carried by this link [6,9,29,30,32,38,40].

\* Corresponding author. Tel.: +1-514-340-4711, ext. 4685; fax: +1-514-340-3240.
E-mail address: samuel.pierre@polymtl.ca (S. Pierre).
Adaptive routing is based on the dynamic characteristics of the network in order to manage the traffic requirements expressed by the traffic matrix. Among these characteristics, consideration must be given to the utilization of the links \( f/C \), their residual capacity \( C - f \), the cost of paths, and so on. Optimization techniques of non-linear functions can be used for determining the optimal routes within a given network topology. A routing strategy can redirect, at each iteration, a part of the flow from the more heavily loaded paths to those less congested. Such a strategy generally leads to the routing of packets derived from the same message over several different paths.

Despite their simplicity of implementation and their speed of execution, the heuristics of static routing do not generally take into account the level of traffic in the network, and thus do not guarantee a good solution to the problem. On the other hand, the adaptive routing procedures take into account the level of traffic, but the storage requirements of the current state of the network at each node may lead to the eventual saturation of the buffer at these nodes. As for optimal routing algorithms [3,19], they are practically never used, due to their complexity and relatively high running time. This raises the question of considering the possibility to design a routing algorithm which, while being simple to implement and fast to execute, also takes into account the level of traffic on each link for efficient operation and for a solution approaching the optimum. The Hopfield model of neural networks appears as a suitable answer to this question [12,13].

Hopfield and Tank [14] have initiated the utilization of neural networks to solve some optimization problems. Other researchers have applied neural networks to the traveling salesman problem [17]. Since then, much research has been carried out on applications of neural networks to other types of optimization problems [4,8,16,23–25,28,33,34,37,39,41].

This paper proposes a heuristic approach based on Hopfield neural networks to solve the routing problem in a topological design context. It is organized as follows. Section 2 summarizes symbols and notation used in the remainder of the paper. Section 3 exposes our approach to solve the routing problem, while Section 4 gives some implementation details. Section 5 reports comparison results with other conventional routing algorithms.
2. The proposed routing method

This section presents an adaptation of Hopfield network in order to solve the problem of selecting routes in packet switching computer networks. This heuristic method determines the shortest path between any node pair in an attempt to minimize the end-to-end delay in a given network topology.

2.1. Outline of the concept of Hopfield networks

The Hopfield networks constitute a class of artificial neural networks which consist of n neurons completely connected, each neuron having two possible states: $V_i = -1$ and $V_i = 1$ (Hopfield used 0 and 1, but this is equivalent) [12–14,36]. The connection of neuron $i$ to neuron $j$ is denoted by $w_{ij}$, and the total entry of a neuron $i$ is equal to $\sum_j w_{ij} V_j$. The network has a dynamic functioning assumed to be sequenced by a clock.

Hopfield [12] introduces the concept of energy of a neural network at a given time $t$. If the states of the network are $V_i \in \{-1, +1\}$, then the energy of the network is defined as:

$$E = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} V_i V_j$$  \hspace{1cm} (1)

The dynamics of Hopfield networks may be described by the following relation [24]:

$$\frac{dU_i}{dt} = \sum_{j=1}^{n} w_{ij} V_j - \frac{U_i}{\tau} + I_i$$ \hspace{1cm} (2)

For a symmetric connection matrix, and for a sufficiently high gain of the amplifiers ($\lambda_i \to \infty$), the dynamics of the neurons follow a decreasing gradient descent of the quadratic energy function $E$ [13]:

$$E = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} V_i V_j - \sum_{i=1}^{n} I_i V_j$$ \hspace{1cm} (3)

In terms of energy function, the following relation describes the dynamics of the $i$th neuron [24]:

$$\frac{dU_i}{dt} = -\frac{U_i}{\tau} - \frac{\partial E}{\partial V_i}$$ \hspace{1cm} (4)

In order to find the shortest path between any node pair of a computer network, we propose in this paper an adaptation of this Hopfield model.

2.2. Network representation

Routing in a packet switching network essentially consists of determining the best path or route between each node pair (source/destination) through the network in order to minimize the network delay, that is the average time taken by a packet traveling from one end to another end of the network. The communication networks considered can be modeled by a graph $G = (N, A)$, where $N$ denotes the set of nodes and $A$ the set of arcs or links. Each arc is associated with a non-negative number $C_{ij}$ that can represent the cost, distance or time of transmission between node $i$ and node $j$. The length of the route between the nodes $s$ and $d$ is defined as follows:

$$L_{sd} = C_{si} + C_{ij} + C_{jk} + \cdots + C_{rd}$$

The problem is to find through the network, the route with the minimal cost (shortest length) between the two nodes $s$ and $d$.

The use of neural networks for finding the best route between the two source/destination nodes was initiated by Rauch and Winarske [34]. They proposed a neural network model arranged in a matrix of dimensions $n \times m$, where $n$ refers to the number of nodes in the network and $m$ the number of nodes which form the searched route. The output of the neuron $(x,i)$ is defined as follows:

$$V_{xi} = \begin{cases} 1 & \text{if the arc } (x,i) \text{ is part of the path} \\ 0 & \text{if not} \end{cases}$$

The limitation of this model is that it requires prior knowledge of the number of nodes which make up the route. Rauch and Winarske have fixed $m$ as the minimum number of nodes which constitute a route. However, their algorithm does not always produce a good solution as the length of the path can vary from one network to another. To remedy this, $m$ has been extended to the total number of nodes of the network, that is the maximum length that a route may have. The problem with this change is that the synaptic connections of the neurons need updating and adjustment each time the searched route is changed.

Based on the Hopfield networks, our model consists of $n(n-1)$ neurons represented by a matrix $n \times n$ where all the neurons on the diagonal are eliminated. The coordinates of the neurons are $(x,i)$, where $x$ denotes the rows, and $i$ the columns. The neuron at $(x,i)$ is characterized by its output $V_{xi}$ and defined as follows:

$$V_{xi} = \begin{cases} 1 & \text{if the arc } (x,i) \text{ is part of the route} \\ 0 & \text{if not} \end{cases}$$

and the variable $\rho_{xi}$ defined as follows:

$$\rho_{xi} = \begin{cases} 0 & \text{if the arc } (x,i) \text{ exists} \\ 1 & \text{if not} \end{cases}$$

The cost of the arc $(x,i)$ is denoted by $C_{xi}$ which is a real positive variable. Null costs are assigned to the non-existent arcs.

To solve the problem, it is necessary to define the energy function used for the minimization process of the neural networks. For the purposes of numerical manipulation associated with calculations of the derivatives, but without loss of generality, we have adopted the function proposed by
2.3. Determination of routes

As the neurons are organized into a two-dimensional table Eqs. (2) and (4) can be rewritten, respectively, as follows:

\[
\frac{dU_{si}}{dt} = \sum_{y=1}^{n} \sum_{j=1}^{n} w_{sj,y} V_{yi} - \frac{U_{si}}{\tau} + I_{si}
\]

(6)

\[
\frac{dU_{si}}{dt} = -\frac{U_{si}}{\tau} - \frac{\partial E}{\partial V_{si}}
\]

(7)

By substituting Eq. (5) in Eq. (7) and calculating the derivative \(\partial E/\partial V_{si}\), we obtain:

\[
\frac{dU_{si}}{dt} = -\frac{U_{si}}{\tau} - \frac{\mu_1}{2} C_{si}(1 - \delta_{sd} \delta_{ds}) - \frac{\mu_3}{2} \rho_{si}(1 - \delta_{sd} \delta_{ds})
\]

\[- \mu_3 \sum_{y=1}^{n} \sum_{y \neq i} (V_{xy} - V_{sy}) + \mu_3 \sum_{y \neq i} (V_{iy} - V_{yi})
\]

\[- \frac{\mu_4}{2} (1 - 2V_{si}) + \frac{\mu_5}{2} \delta_{sd} \delta_{ds}
\]

(8)

where \(\delta\) is the Kronecker symbol, which is defined as follows: \(\delta_{ab} = 1\) if \(a = b\), and \(\delta_{ab} = 0\) if not. By comparing the coefficients of Eqs. (6) and (8), the \(w_{sij}\) of the
connection matrix take the following values:

\[ w_{x,y} = \mu_2 \delta_{x,y} - \mu_3 \delta_{y} - \mu_3 \delta_{x} + \mu_3 \delta_{y} + \mu_3 \delta_{x,y} \] (9)

\[ I_{d(t)} = -\frac{\mu_1}{2} C_{d(t)}(1 - \delta_{a,d} \delta_{b,t}) - \frac{\mu_2}{2} \rho_{a,d}(1 - \delta_{a,d} \delta_{b,t}) - \frac{\mu_3}{2} \] 

\[ \delta_{a,b} \delta_{b,t} \]

\[ = \left\{ \begin{array}{ll}
\frac{\mu_5}{2} - \frac{\mu_4}{2} & \text{if } (x, i) = (d, s) \\
-\frac{\mu_1}{2} C_{d,i} - \frac{\mu_2}{2} \rho_{d,i} - \frac{\mu_3}{2} & \text{if not } \Psi(x \neq i), (y \neq j)
\end{array} \right. \] (10)

The initial data of the neurons \( U_{d(t)} \) at time \( t \) are equal to zero, and the evolution of the state of the neural network is simulated by the solution of a system of \( n(n - 1) \) differential equations where the variables are the neuron outputs \( V_{d(t)} \). To solve this system, we have used the numerical method of Runge–Kutta of the fourth order. The solution consists of observing the outputs of the neurons \( V_{d(t)} \) for a specific duration \( \delta t \). The circuit’s time constant \( \tau \) for each neuron is initialized to 1. We have also noted that, in order to obtain good results, \( \delta t \) must be between \( 10^{-7} \) and \( 10^{-3} \). To avoid bias in favor of any particular path, it must be assumed that all inputs \( U_{d(t)} \) are equal to 0. However, to help the network converge rapidly, while preventing it from adopting an undesirable state (for example, convergence of two different paths), small perturbations must be made to the initial inputs of the \( U_{d(t)} \) network. At the start, it is assumed that \(-0.0002 \leq \delta U_{d(t)} \leq +0.0002 \). The calculations cease when the network reaches a stable state, that is, when the difference between the outputs is less than \( 10^{-5} \) \( V_{d(t)} \) \( \leq 0.00001 \) from one update to another. When the network is in a stable state, the final values of \( V_{d(t)} \) are rounded off, that is, they are set to 0 if \( V_{d(t)} < 0.5 \), and to 1 otherwise.

The \( \mu_i \) coefficients of the neural network serve as regulators and as precision factors to avoid blocking in a local minimum. Hopfield and Tank [14] used a formulation which combines mixed constraints of legality. This solution, although simple, is far from efficient since much time is lost avoiding illegal states.

Mehmet and Kamoun [24] have considered the problem in the form of a system of inequalities as follows:

\[ 2 \mu_3 - \mu_4 > 0 \]

\[ \mu_5 \gg \mu_1 (C_{d(i)})_{\text{max}} \]

\[ \mu_2 = \mu_5 \]

\[ \mu_1 \gg 2 \mu_3 (C_{d})_{\text{max}} \]

These are based on the fact the energy function is quadratic and positive definite. In fact, in solving combinatorial optimization problems using Hopfield model, one major efficiency issue is the lack of rigorous guidelines allowing selecting appropriate values of the energy function coefficients. For this reason, some general guidelines related to this selection are needed. The objective is to choose appropriate values of \( \mu_i \)'s such that the neural dynamics will converge to the shortest valid path. For these purposes, under some reasonable assumption and by taking into account some specific requirements for the \( \mu_i \) values, Mehmet and Kamoun [24] have chosen the weighting coefficients as follows:

\[ \mu_1 = 950; \mu_2 = 2500; \mu_3 = 1500; \mu_4 = 475; \mu_5 = 2500. \]

By using these values, the simulated neural algorithm proposed by these researchers is run 100 times using different randomly generated link costs, between 0 and 1, for a network of 5 nodes and 8 links. The number of iterations needed by this algorithm to converge to valid solutions varied from 3000 to 8000. In order to reduce this number and therefore speed up the convergence, we have adopted for the \( \mu_i \) coefficients the following form:

\[ \mu_i = \text{const} \cdot \text{Vit} \]

where const is an integer coefficient and Vit an integer variable which takes values from 1 to 5. This variable serves as a factor of speed and precision. Simulation has shown that, beyond the value of 5, the neural network begins to diverge, leading to poor results.

By interpreting in an other way the general guidelines proposed by Mehmet and Kamoun [24] for the selection of the weighting coefficients, we propose the following set of constraints:

\[ \mu_4 < \mu_1 < \mu_3 \]

\[ \mu_3 < \mu_5 < \mu_2 \]

By merging these constraints, it follows that \( \mu_4 < \mu_1 < \mu_3 < \mu_5 < \mu_2 \), that is congruent to the system of inequalities proposed by Mehmet and Kamoun. Furthermore, we have observed through simulation that good results is generally obtained for:

\[ \mu_2 \approx 5 \mu_4 \]

In summary, the method proposed in this paper to select the weighting coefficients is defined by the following set of hybrid constraints:

\[ \mu_4 < \mu_1 < \mu_3 < \mu_5 < \mu_2 \]

\[ \mu_2 \approx 5 \mu_4 \]

\[ \mu_3 \approx 5 \mu_4 \]

By taking into account these constraints, we have simulated the time evolution of the state of the neural network by numerically solving Eq. (8), for a great number of network topologies. This simulation leads to the following values for
the $\mu_i$ coefficients:

$\mu_1 = 76\,000 \cdot \text{Vit}$

$\mu_2 = 250\,000 \cdot \text{Vit}$

$\mu_3 = 150\,000 \cdot \text{Vit}$

$\mu_4 = 50\,000 \cdot \text{Vit}$

$\mu_5 = 230\,000 \cdot \text{Vit}$

where Vit takes, for a specific network topology, one value from 1 to 5. The mean number of iterations needed to converge to states corresponding to valid solutions which are often global optima is equal to 100, compared to 5000 in the work of Mehmet and Kamoun [24].

Here are the main steps in the search of a route:

**Step 1.** Obtain the network data: the number $n$ of nodes, the matrix $r$ of the links, the matrix of link costs;

**Step 2.** Initialize the matrix of the $V_{ui}$: the $V_{ui}$'s take random values between $-0.0002$ and $+0.0002$;

**Step 3.** Trigger the process of minimization in the neural network to solve the differential equations and to stabilize the network;

**Step 4.** Round off the values of the matrix of outputs and obtain the searched route.

The example in Fig. 3 concerns a network of six nodes, which has two corresponding matrices: the link matrix, and the cost matrix (expressed in km). The problem is to find all the routes between all node pairs in the network. Fig. 4 shows the paths between node s and node d in the matrix of outputs, the 1's denote only the connections belonging to the path. The connections found form a cycle; however, in the construction of the path, it is necessary to eliminate the link between the source and destination nodes.

### 2.4. Routing and delay computation

The ultimate goal of a routing procedure remains the flow and capacity assignment which allows us to compute the end-to-end delay of a given network topology [26]. The routing matrix provides, for each source and destination pair, the best route along which the packets must travel through the network. From this matrix, we can calculate the flow carried by each link, by distributing the traffic between each node pair on the routes of the routing matrix. Then, a capacity value can be assigned to each link by using capacity options available on the marketplace, and to ensure that the link flow does not exceed the link.

In the case of moderate-speed networks, the end-to-end delay...
In the case of high-speed networks, the delay may be of the connections, as shown in Fig. 3.

The traffic matrix $V_{ij}$, each element represents a neuron and the value $\hat{i}$ represents the output of the neuron $(i,j)$ after it is updated. Link costs are denoted by a matrix having the same dimensions as $V_{ij}$.

The capacity options are given in Table 2. In applying our routing method to the network in Fig. 5, for a uniform traffic of 10 packets/s and an average packet size $(1/\mu)$ of 1000 bits, the end-to-end delay obtained from Eq. (11) is 48.7 ms. An increase in traffic level from 10 packets/s to 150 packets/s between all node pairs results in a high-speed network. The end-to-end delay is then calculated using Eq. (12) and the capacity options given in Table 3: it is equal to 11.2 ms.

For both traffic levels, the attributes of the links are shown in Table 5. In this table, for a low traffic level, about 50% of the links (8 over 19) have relatively great values of flow, leading to some great link capacities of 230.4 kbps; for a high traffic level, only less than 22% of these links (4 over 19) have truly high values of flow, and consequently high-speed links of 3.152 Mbps corresponding to DS-1C links (see Table 3). Such a reduction in the number of high-capacity links possibly results in scale economy in terms of communication costs according to the traffic requirements.

### 3. Implementation details

In our implementation, the configuration of the given network is represented by a characteristic matrix $\text{Mat} = [\text{Mat}_{ij}]$, $\text{Mat}_{ij} \in \{0, 1\}$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n$, and $\forall i, j \neq n$. The matrix Mat is composed of a series of 1’s and 0’s which indicate whether a link exists or not. For example, Table 1 represents the configuration of the network shown in Fig. 3.

The neural network is represented by a matrix $V = [V_{ij}]$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n$, and $\forall i, j \neq n$. In this matrix, each element represents a neuron and the value $V_{ij}$ represents the output of the neuron $(i,j)$ after it is updated. Link costs are denoted by a matrix having the same dimensions as the matrix of neurons, where the entries represent the length of the connections, as shown in Fig. 3.

The traffic matrix $\Gamma = [\gamma_{ij}]$, with $i, j = 1, 2, \ldots, n$, and $i \neq j$, may be uniform (the same number of packets circulates between each node pair, or random (the number of packets between each node pair is expressed as an integer ranging from 1 to 10). Table 2 gives in kbps a sample of modular values of low and moderate link capacities available on the marketplace corresponding to low traffic level. High link capacity options in Mbps, corresponding to high traffic level, are given in Table 3. For each link of a network topology, the capacity is selected from these tables; it must be greater the link flow and be nonnegative.

Our method essentially consists of obtaining the network configuration and the associated data; then it determines all the routes between each pair of nodes, and generates the link flows and capacities allowing us to calculate the mean delay. An implementation has been realized in Turbo-Pascal version 7.0 in a DOS environment. The experiments and simulation have been carried out on an IBM compatible, Pentium 133 MHz microcomputer.

Let us consider the network topology of 10 nodes shown in Fig. 5. The Cartesian coordinates are shown in Table 4. The capacity options are given in Table 2. In applying our routing method to the network in Fig. 5, for a uniform traffic of 10 packets/s and an average packet size $(1/\mu)$ of 1000 bits, the end-to-end delay obtained from Eq. (11) is 48.7 ms. An increase in traffic level from 10 packets/s to 150 packets/s between all node pairs results in a high-speed network. The end-to-end delay is then calculated using Eq. (12) and the capacity options given in Table 3: it is equal to 11.2 ms.

For both traffic levels, the attributes of the links are shown in Table 5. In this table, for a low traffic level, about 50% of the links (8 over 19) have relatively great values of flow, leading to some great link capacities of 230.4 kbps; for a high traffic level, only less than 22% of these links (4 over 19) have truly high values of flow, and consequently high-speed links of 3.152 Mbps corresponding to DS-1C links (see Table 3). Such a reduction in the number of high-capacity links possibly results in scale economy in terms of communication costs according to the traffic requirements.

### 4. Comparisons with optimal routing algorithms

In order to evaluate the efficiency of our neural routing algorithm, it has been compared with well-known conventional and optimal routing procedures such as: Shortest distance, Minimum number of hops, Flow Deviation, and Bertsekas and Gallager. *Shortest distance* refers to the well-known procedure called *Shortest path*, which plays a central role in the design and analysis of networks. Indeed, many routing problems can be solved as shortest path problems once an appropriate metric is chosen and an appropriate “length” is assigned to each link in the network.

*Shortest distance* is a routing procedure which consists of finding the shortest path linking each node pair of a network, by taking into account the length $l_k$ of each link $k$ composing the path. In this context, the length of a link is expressed as the Euclidean distance between its two end nodes. There exist in the literature many algorithms to solve this problem. In this paper, Dijkstra’s algorithm has been chosen [19] for the sake of simplicity and without loss of generality. There may be constraints on the number of links (or hops) in the path. In this case, the more general shortest distance...
procedure can be slightly modified by setting \( l_k = 1 \) for all \( k = 1, 2, \ldots, m \), for becoming the procedure called *Minimum number of hops*.

To solve the problem of finding a set of routes which minimize the average end-to-end delay, given a network topology and a set of traffic requirements, Flow Deviation (FD) [6] algorithm can be also used. The FD algorithm is based on the observation that, if a small amount of flow is moved from a path with larger incremental delay to a path with smaller incremental delay, then the average end-to-end delay is decreased. It proceeds by assigning lengths to the links, based on their incremental delay, then finding the shortest path from each source to each destination. The amount of flow deviated to the new path is chosen such that the end-to-end delay is minimized. When the FD algorithm terminates, all flow is on minimum incremental delay paths.

An alternative method for finding minimum routes which minimizes the end-to-end delay has been proposed by Bertsekas and Gallager [19]. The BG algorithm differs from the FD algorithm in two ways: it moves flow for one source–destination pair at a time; it computes the link flow to move directly rather than performing a line search. Such changes improve significantly the efficiency of the routing algorithm in most cases.

Let us consider now a network topology of eight nodes and a uniform traffic of 2 packets/s. The configuration of the network is shown in Fig. 6, the node coordinates are indicated in Table 6, the cost matrix is given in Table 7, and the capacity options are presented in Table 8. We have measured the time \( t_{\text{run}} \) required to obtain each of the results.

The comparison of these values with optimal routing algorithms are summarized in Table 9. The end-to-end delay derived from our neural-network-based routing is less than those given by other heuristic routing procedures such as conventional shortest path (shortest distance) and minimum number of hops, with comparable running times. In spite of the gap that separates our delay results from those provided by optimal algorithms, running times, which are considerably shorter in our case, compensates this difference. The large number of iterations required by optimal algorithms before obtaining a solution causes this contrast in running times.

In order to put in evidence the trade-off between end-to-end delay \( T \) and running time \( t_{\text{run}} \), an extensive simulation experience has been carried out with 500 network topologies of variable size. For simulation purposes, the size of a network topology is characterized by both the number \( n \) of nodes and the number \( m \) of links of such topology. For sake of simplicity, the number of links of a topology is fixed to \( m = 3n \), where \( n = 10, 15, 20, 25, 30 \). For each value of \( n \), 100 network topologies of \( 3n \) full-duplex links and at least two routes between each node pair have been randomly generated. For each topology \( \theta \), the end-to-end delay \( T(\theta) \) and the running time \( t_{\text{run}}(\theta) \) have been computed, using successively the five routing methods mentioned in Table 9. For each value of \( n \), the average values \( T_{\text{aver}} \) and \( t_{\text{run}} \) have been then computed as follows:

\[
T_{\text{aver}} = \frac{1}{100} \sum_{\theta=1}^{100} T(\theta)
\]

\[
t_{\text{aver}} = \frac{1}{100} \sum_{\theta=1}^{100} t_{\text{run}}(\theta)
\]

Then, we have defined the trade-off between delay and running time by the following index:

\[
I_p = \log(100 T_{\text{aver}}^{-1} t_{\text{aver}})
\]

The log function and the factor 100 are introduced only for making easier the graphical representation of the trade-off delay versus running time. The results of this experience are shown in Fig. 7. Our neural routing algorithm realizes a better trade-off between end-to-end delay and running time.

<table>
<thead>
<tr>
<th>Type of link</th>
<th>DS-1</th>
<th>DS-1C</th>
<th>DS-2</th>
<th>2 DS-2</th>
<th>12 DS-1</th>
<th>4 DS-2</th>
<th>DS-3</th>
<th>2 DS-3</th>
<th>4 DS-3</th>
<th>DS-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C ) (Mbps)</td>
<td>1.544</td>
<td>3.152</td>
<td>6.312</td>
<td>12.624</td>
<td>18.528</td>
<td>25.248</td>
<td>44.736</td>
<td>89.472</td>
<td>178.94</td>
<td>274.18</td>
</tr>
</tbody>
</table>

Table 3
Options of high link capacities

<table>
<thead>
<tr>
<th>Node</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abscissa</td>
<td>10</td>
<td>5</td>
<td>10</td>
<td>50</td>
<td>45</td>
<td>55</td>
<td>70</td>
<td>70</td>
<td>95</td>
<td>95</td>
</tr>
<tr>
<td>Ordinate</td>
<td>90</td>
<td>40</td>
<td>15</td>
<td>50</td>
<td>80</td>
<td>95</td>
<td>75</td>
<td>20</td>
<td>35</td>
<td>70</td>
</tr>
</tbody>
</table>

Table 4
Cartesian coordinates of the nodes in Fig. 5
and consequently gives a better performance than the other conventional algorithms considered.

5. Conclusions

Routing constitutes one of the key aspects of the overall problem of topological design of communications networks. Its solution by means of neural networks presents advantages and inconveniences with respect to existing routing techniques. Optimization with neural networks is based on the utilization of an energy function, which simulates the objective function used in network optimization. Such a function must respect the constraints imposed by the network designers and must converge toward a solution which, if not the best, is at least as close as possible to the optimum. The optimization is done in a global manner, in contrast to conventional routing heuristics that are in general of a greedy type. In this sense, all the neurons simulating the nodes of the network communicate with one another.

In summary, the method proposed in this paper to select the weighting coefficients relies on a set of hybrid constraints which are congruent with the approach proposed by Mehmet and Kamoun [24]. By taking into account these constraints, we have simulated the time evolution of the state of the neural network for a great number of network topologies. With the values derived from this simulation for the weighting coefficients, the mean number of iterations needed to converge to states corresponding to valid solutions which are often global optima is equal to 100, compared to 5000 in the work of Mehmet and Kamoun [24].

Furthermore, the analysis of the results reveals that the end-to-end delay computed according to our method is, in most cases, better than those determined by conventional routing methods such as the Shortest Path and the Minimum Number of Hops. Compared with optimal algorithms such as

Table 5
Link attributes of the network in Fig. 5, for low and high traffic levels

<table>
<thead>
<tr>
<th>Link</th>
<th>Low Traffic Level</th>
<th>High Traffic Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Flow (Kbps)</td>
<td>Capacity (Kbps)</td>
</tr>
<tr>
<td>1–2</td>
<td>20</td>
<td>50.00</td>
</tr>
<tr>
<td>1–3</td>
<td>40</td>
<td>50.00</td>
</tr>
<tr>
<td>1–5</td>
<td>100</td>
<td>230.00</td>
</tr>
<tr>
<td>1–6</td>
<td>60</td>
<td>100.00</td>
</tr>
<tr>
<td>2–3</td>
<td>80</td>
<td>100.00</td>
</tr>
<tr>
<td>2–4</td>
<td>200</td>
<td>230.00</td>
</tr>
<tr>
<td>3–8</td>
<td>40</td>
<td>50.00</td>
</tr>
<tr>
<td>3–9</td>
<td>20</td>
<td>50.00</td>
</tr>
<tr>
<td>4–5</td>
<td>180</td>
<td>230.40</td>
</tr>
<tr>
<td>4–7</td>
<td>120</td>
<td>230.40</td>
</tr>
<tr>
<td>4–8</td>
<td>160</td>
<td>230.40</td>
</tr>
<tr>
<td>5–6</td>
<td>80</td>
<td>100.00</td>
</tr>
<tr>
<td>5–7</td>
<td>100</td>
<td>230.00</td>
</tr>
<tr>
<td>6–7</td>
<td>20</td>
<td>50.00</td>
</tr>
<tr>
<td>6–10</td>
<td>60</td>
<td>100.00</td>
</tr>
<tr>
<td>7–10</td>
<td>140</td>
<td>230.40</td>
</tr>
<tr>
<td>8–9</td>
<td>60</td>
<td>100.00</td>
</tr>
<tr>
<td>8–10</td>
<td>40</td>
<td>50.00</td>
</tr>
<tr>
<td>9–10</td>
<td>100</td>
<td>230.40</td>
</tr>
</tbody>
</table>

Fig. 6. Network of 8 nodes.

Table 6
Node coordinates of the network in Fig. 6

<table>
<thead>
<tr>
<th>Nodes</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abscissa</td>
<td>444</td>
<td>60</td>
<td>324</td>
<td>287</td>
<td>433</td>
<td>8</td>
<td>507</td>
<td>176</td>
</tr>
<tr>
<td>Ordinate</td>
<td>447</td>
<td>16</td>
<td>346</td>
<td>96</td>
<td>396</td>
<td>89</td>
<td>103</td>
<td>191</td>
</tr>
</tbody>
</table>

Table 7
Cost matrix associated with Fig. 6

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.05</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.08</td>
<td>0</td>
<td>0</td>
<td>0.13</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.21</td>
<td>0.04</td>
<td>0.34</td>
<td>0.21</td>
<td>0.22</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0.21</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.12</td>
</tr>
<tr>
<td>5</td>
<td>0.05</td>
<td>0.04</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.23</td>
<td>0.26</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0.08</td>
<td>0.34</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.08</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0.21</td>
<td>0.23</td>
<td>0</td>
<td>0</td>
<td>0.28</td>
<td>0.28</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0.13</td>
<td>0.22</td>
<td>0.12</td>
<td>0.26</td>
<td>0</td>
<td>0.28</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 8
Capacity options

<table>
<thead>
<tr>
<th>C (kbps)</th>
<th>9.6</th>
<th>19.2</th>
<th>28.8</th>
<th>38.4</th>
<th>48.0</th>
<th>57.6</th>
<th>67.2</th>
<th>76.8</th>
</tr>
</thead>
</table>

Table 9
Comparisons with optimal routing algorithms

<table>
<thead>
<tr>
<th>Routing method</th>
<th>End-to-end delay $T_{\text{mean}}$ (s)</th>
<th>Running time $t_{\text{run}}$ (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortest distance</td>
<td>0.795</td>
<td>0.110</td>
</tr>
<tr>
<td>Minimum no. of “Hops”</td>
<td>0.627</td>
<td>0.110</td>
</tr>
<tr>
<td>Neural network</td>
<td>0.3450</td>
<td>0.114</td>
</tr>
<tr>
<td>Flow-deviation</td>
<td>0.190</td>
<td>5.54</td>
</tr>
<tr>
<td>Bertsekas–Gallager</td>
<td>0.187</td>
<td>13.94</td>
</tr>
</tbody>
</table>
the Flow Deviation method and the algorithm of Bertsekas–
Gallager, our delay results are slightly less favorable than
the ones obtained with these algorithms; however, our
execution times are considerably shorter. For a low traffic
level, about 50% of the links have relatively great values of
flow, leading to some great link capacities; for a high traffic
level, only less than 22% of these links have truly high
values of flow, and consequently high-speed. Such a reduc-
tion in the number of high capacity links possibly results in
scale economy in terms of communication costs according
to the traffic requirements. Finally, our neural routing algo-
rithm realizes the better trade-off between end-to-end delay
and running time, and consequently gives a better perfor-
mance than the other conventional algorithms considered.

Acknowledgements

This work was supported in part by the Natural Science
and Engineering Research Council (NSERC) of Canada
under grant 140264-98.

References

packet switching networks: Gb/s challenge. IEEE Network

tion networks. Proceedings of the 22nd IEEE Conference on Compu-
ter Networks, LCN 97, 2–5 November, 1997, Minneapolis,


packed switching networks. IEEE Trans Neural Networks


models for communication network design. IEEE Trans Knowledge


and-forward communication network design. Networks 1974;3:97–
133.

[10] Gavish B. Topological design of computer networks — the overall


[12] Hopfield JJ. Neural network and physical systems with emergent
collective computation abilities. Proc Natl Acad Sci

[13] Hopfield JJ. Neurons with graded response have collective computa-
tional properties like those of two-state neurons. Proc Natl Acad Sci

[14] Hopfield JJ, Tank DW. Neural computations of decisions in optimization

nication network subject to reliability constraints. IEEE Trans Reliab

[16] Kagiavas AG, Papantoni-Kazakos T. Operations and learning in
neural networks for robust prediction. IEEE Trans Systems Man

[17] Kahnrg A. Traveling salesman heuristics and embedding dimension in

ecomonal configuration of elementary packet-switched networks.


mesh network topological optimization and routing. IEEE Trans

supra-high-speed metropolitan (or wide) area networks. Comp


[23] Lee S, Chang S. Neural network for routing of communication
networks with unreliable components. IEEE Trans Neural Networks

computation and routing in computer networks. IEEE Trans Neural

distance minimization. Proc IEEE Int Conf. Neural Networks

[26] Neumann I, system A. For priority routing and capacity assignment in


