FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 19, no. 2, August 2006, 317-329

Neural Network for Optimization of Routing in Communication Networks

Nenad Kojić, Irini Reljin, and Branimir Reljin

Abstract: The efficient neural network algorithm for optimization of routing in communication networks is suggested. As it was known from literature, different optimization and ill-defined problems may be resolved using appropriately designed neural networks, due to their high computational speed and the possibility of working with uncertain data. Under some assumptions, the routing in packet-switched communication networks may be considered as optimization problem, more precisely, as a shortest-path problem. The Hopfield-type neural network is a very efficient tool for solving such problems. The suggested routing algorithm is designed to find the optimal path, meaning, the shortest path (if possible), but taking into account the traffic conditions: the incoming traffic flow, routers occupancy, and link capacities, avoiding the packet loss due to the input buffer overflow. The applicability of the proposed model is demonstrated through computer simulations in different traffic conditions and for different full-connected networks with both symmetrical and non-symmetrical links.

Keywords: Routing, shortest path, Hopfield type neural network, optimization, packet switching, communication networks.

1 Introduction

In modern communication networks, particularly in packet switched networks, routing is an important process that has a significant impact on the network's performance. Ideal routing algorithm comprises finding the "optimal" path(s) between

Manuscript received Mart 10, 2006. An earlier version of this paper was presented at the 7th Seminar on Neural Network Applications in Electrical Engineering, NEUREL-2004, September 23-25, 2004, Belgrade, Serbia and Montenegro.

The authors are with the Faculty of Electrical Engineering, University of Belgrade, Bulevar kralja Aleksandra 73, 11000 Belgrade, Serbia, (e-mails: nkojic@eunet.yu, ireljin@ptt.yu, reljinb@etf.bg.ac.yu).

source and destination router, enabling high-speed data transmission and avoiding a packet loss. Since modern communication traffic is characterized by huge amount of source-destination pairs, high variability (burstiness), nonlinearity and unpredictability, the routing policy is a very hard task. Under some assumptions the optimal routing may be considered as the shortest path (SP) computations that have to be carried out in real time. This makes neural networks very good candidates for solving the problem, due to their high computational speed and the possibility of working with uncertain data. In their famous paper [1], Hopfield and Tank described a neural network suitable for solving different optimization problems - among others the well-known Traveling Salesman Problem (TSP): it has to visit given set of n cities, passing only once each city and returning to starting point, forming the shortest path loop. Algorithmically speaking, the TSP may be described by a square matrix of dimension $n \times n$ whose rows correspond to the particular city, X, while columns denote the sequence, i, of passing through cities. Such a matrix has only one non-zero element, of the value of unity, at each row and at each column. As an example, for $n = 5$ cities, labeled A, B, C, D and E, with shortest path loop $A-C-B-E-D-A$, this matrix has the following form:

	1	2	3	4	5
A	1				
B			1		
C		1			
D					1
E				1	

. (1)

For n cities case there are $n!$ possible loops of general form described by (1). Among all solutions there are $2n$ loops with the same lengths, since each loop has the degeneracy of n-th order, depending on the starting point in the loop, and the second-order degeneracy depending on the loop direction. Consequently, there is $n!/2n$ different loops among which it is necessary to find the shortest one. The TSP problem is computationally very hard if the number of cities increases. For instance, for $n = 5$ there is only 12 different loops, for $n = 10$ the number is 181,440, while for $n = 30$ it amounts 4.4×10^{30}, which highly extends the number of stars in Milky Way (about 10^{11} stars)!

For solving different optimization problems Hopfield and Tank proposed a neural network of the recurrent type [1]. The computational power of their approach was demonstrated just on the n cities TSP problem. Since then many researchers used similar model in solving a variety of combinatorial optimization problems. Among others, the routing problems in telecommunication and computer networks were under assumption. The graph nodes in neural network correspond to positions of source and destination routers (i, j) in the initial communication network.

Instead of distances, d_{ij}, between routers (in normalized space $[0,1]$) some other attributes, called the 'cost' can be introduced describing the transmission conditions; for instance, the link capacity, traffic flow, transit time, etc. [2]. Under these assumptions the routing can be considered as an optimization problem of the shortest path type.

In this paper a Hopfield-like neural network designed for solving the routing problem, is presented. Section 2 gives a brief review of the Hopfield network and its applicability to solve the TSP problem. In Section 3, a neural network for solving the routing problem in packet switched networks, initiated by previously suggested algorithm [2], is described. The applicability and efficiency of the new algorithm was demonstrated through computer simulations for different full-connected networks. Several characteristic examples are described in Section 4. Some concluding remarks are given in Section 5.

2 Hopfield Neural Network and the TSP Problem

The block scheme of the Hopfield neural computational circuit [1] is depicted in Fig. 1(a). The processing elements (neurons) are full-connected: output, v_i, of each ith cell is connected to inputs of all other neurons via synaptic weights, T_{ij}, producing the modification of cell inputs, u_i, and thus changing the network state. If the system is stable successive iterations lead to smaller and smaller output changes and network reaches some minima of the system energy. Each cell is externally excited by input bias, I_i. These inputs can be used to set the general level of excitability of the whole network (shifting the input-output relation along the u_i axis), or to provide direct parallel inputs to drive specific neurons.

In hardware implementation, processing cells are realized as summing amplifiers, Fig. 1(b), with nonlinear transfer function $g(u_i) = v_i/u_i$, called also an *activation function*. In Hopfield's early work [3] the activation function was a simple threshold (*hard limiter*), and the model was a discrete one. Later on, Hopfield introduced continuous activation function. A common choice is the sigmoidal or logistic function, plotted in Fig. 1(b) upper right

$$g_i(u_i) = \frac{1}{1+e^{-a_i \cdot u_i}}. \tag{2}$$

The coefficient a_i determines the steepness of the sigmoidal function. The activation function $g(u_i)$ approaches the threshold function for large values of a_i, while smaller values of a_i produce more gentle slope, as indicated in Fig. 1(b). Synaptic weights T_{ij} are realized as resistors of resistances $R_{ij} = 1/|T_{ij}|$. This resistor is connected to the non-inverting output of amplifier for excitatory synapses $(T_{ij} > 0)$.

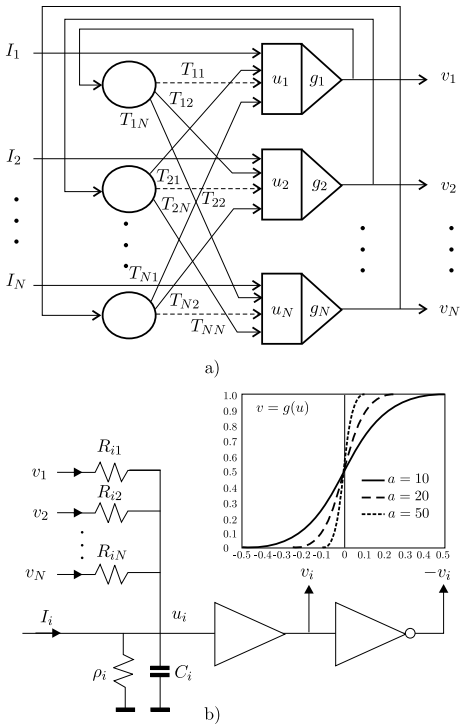

Fig. 1. Hopfield network: a) network structure, b) possible hardware realization of ith neuron (lower drawing) and its activation function (upper diagram).

In case of inhibitory synapses $(T_{ij} < 0)$, it is connected to the inverting output of amplifier. An additional RC section, ρ_i, C_i, defines the time constant of the cell and provides the integrative analog summation of the synaptic input currents from other neurons.

State equations of the circuit in Fig. 1(b) is described by

$$C_i \frac{du_i}{dt} = \sum_{\substack{j=1 \\ j \neq i}}^{N} T_{ij} v_j - \frac{u_i}{R_i} + I_i, \quad i = 1, 2, \ldots, N \tag{3}$$

where R_i is an equivalent resistance connected to the cell's capacitor C_i. Usually (but not necessarily) all neurons are identical, except synaptic conductance.

Hopfield network with N neurons may have $M = 2^N$ distinct states associated with an N-dimensional hypercube with sides $v_i \in \{0, 1\}$. Stable states are determined by the network weights and the current inputs. It was shown [4] that recurrent networks are stable if the matrix of weights is symmetrical with zeros on its main diagonal. In Hopfield realization [1] that means, $T_{ij} = T_{ji}$, and $T_{ii} = 0$ -

thus, in Fig. 1a weights T_{ii} are plotted as dashed lines. Under these assumptions, and if constants a_i are sufficiently large (for instance, $a_i > 100$), the stability of the network, in Liapunov sense, may be verified by observing the energy function, E, describing the state of the network:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ i\neq j}}^{N} T_{ij}v_j v_i - \sum_{i=1}^{N} I_i v_i. \tag{4}$$

The change in energy, due to the change in the state of neuron i, is

$$\frac{\partial E}{\partial v_i} = -\sum_{j=1}^{N} T_{ij}v_j - \sum_{i=1}^{N} I_i. \tag{5}$$

Comparing (5) and (3), the state equation (3) can be rewritten as

$$C_i\frac{du_i}{dt} = -\frac{u_i}{R_i} - \frac{\partial E}{\partial v_i}$$

or, in stable state, when signals are unchangeable: $du_i/dt = 0$, one can obtain

$$\partial E = -\frac{u_i}{R_i}\partial v_i. \tag{6}$$

From (6), and the activation function of the form (2), we can conclude that the term $u_i\partial v_i$ is strictly positive or zero, and the change in energy, ∂E, must be negative or zero - the network energy must either decrease or stay constant, hence, the system is stable.

Hopfield and Tank have applied the neural network, briefly described above, to the TSP n cities problem [1]. They used a neural net with $N = n \times n$ neurons and by minimizing its energy function they found the shortest path - the minimum of the energy function (the most stable state) corresponded to the shortest path. The routers belonging to this path have the final states equal to unity, as in matrix (1). Energy function must favor strongly stable states of the form as described by a matrix (1), and of the $n!$ such solutions it must favor those representing shortest paths. Possible energy function satisfying these requirements is of the form

$$\begin{aligned} E =& \frac{A}{2}\sum_X\sum_i\sum_{j\neq i} v_{Xi}v_{Xj} + \frac{B}{2}\sum_i\sum_X\sum_{X\neq Y} v_{Xi}v_{Yi} \\ &+ \frac{C}{2}\left(\sum_X\sum_i v_{Xi} - n\right)^2 + \frac{D}{2}\sum_X\sum_{Y\neq X}\sum_i d_{XY}v_{Xi}(v_{Y,i+1} + v_{Y,i-1}). \end{aligned} \tag{7}$$

The first three terms, if A, B and C are sufficiently large and positive, enforce valid tours, while the last term assures that the valid tour represents the shortest path. Subscripts are of the form 'modulo n', that means the nth city is adjacent in the tour to both city $(n-1)$ and city 1: $v_{Y,n+j} = v_{Y,j}$. In all terms double indices of the form Xi are used where the first (row) subscript corresponds to the city name and the second (column) subscript has the interpretation of the position of that city in a tour.

From relations (7), (4) and (3) the conductance matrix for the TSP problem solving, has the terms

$$T_{Xi,Yj} = -A\delta_{XY}(1-\delta_{ij}) - B\delta_{ij}(1-\delta_{XY}) - C - Dd_{XY}(\delta_{j,i+1}+\delta_{j,i-1}) \qquad (8)$$

where $\delta_{ii} = 1$, $\delta_{ij} = 0$, for $i \neq j$. and the bias currents are $I_{xi} = Cn$. The Hopfield network may be very efficient in solving the TSP problem despite of some difficulties. Namely, the choice of network parameters is not unique, and the solution obtained is not always just the shortest path but, it always belongs to one of the shortest valid paths [1].

3 Design of Neural Network for Routing in Packet-Switched Network

The optimal routing in communication network may be considered as the shortest path (SP) problem. For this purpose instead of distances, d_{ij}, describing the path between routers i and j, some other attributes called the link costs, C_{ij}, are associated to links, describing the transmission conditions between routers [2]. Then, the goal is to minimize the total cost, i.e., to find the 'path' between source (s)and destination router (d): $C_{si}+C_{ij}+C_{jk}+\cdots+C_{rd}$, that has the minimum length. The use of neural networks to find the shortest path between a given source-destination pair was described in several papers. In [5] the authors proposed a neural network of size $n \times m$ where $m < n$ denotes the number of routers forming the path. Serious limitation in their representation is that it requires a prior knowledge of the number of routers in the shortest path. Further improvements are derived in [6]. Authors used the Hopfield network and the neural network is designed to find the shortest path between given source-destination pair. However, for another source-destination pair, the neural configuration has to be changed. Furthermore, the cost term in their energy function is quadratic, associated to the term A, in (7), that means, associated to synaptic conductances. This solution is not appropriate for real case because in practice the link costs in communication network are time varying and, consequently, the conductances have to be changed, too.

Significant improvements in neural network algorithm are derived in paper offered by Ali and Kamoun [2]. For n routers problem their computational network

uses $n(n-1)$ neurons – the diagonal elements in the connection matrix $n \times n$ are removed – and find the shortest path from final stable neuron states. A suitable energy function is of the form

$$\begin{aligned} E =& \frac{\mu_1}{2} \sum_X \sum_{\substack{i \neq X \\ (X,i) \neq (d,s)}} C_{Xi} v_{Xi} + \frac{\mu_2}{2} \sum_X \sum_{\substack{i \neq X \\ (X,i) \neq (d,s)}} \rho_{Xi} v_{Xi} + \frac{\mu_5}{2}(1 - v_{ds}) \\ &+ \frac{\mu_3}{2} \sum_X \left(\sum_{i \neq X} v_{Xi} - \sum_{i \neq X} v_{iX} \right)^2 + \frac{\mu_4}{2} \sum_i \sum_{X \neq i} v_{Xi}(1 - v_{Xi}) \end{aligned} \tag{9}$$

Coefficients C_{Xi} are the link costs from router X to router i and the terms ρ_{Xi} describe the connection between routers: the value is 1 if routers are not connected, and 0 for connected routers. The term μ_1 minimizes the total cost; μ_2 prevents nonexistent links from being included in the chosen path; μ_3 is zero for every router in the valid path (the number of incoming links is equal to the number of outgoing links); μ_4 forces the state of the neural network to converge to one of the stable states – corners of the hypercube defined by $v_i \in \{0,1\}$. The state vi is close to 1 for router belonging to the valid path, otherwise the state is close to 0. The term μ_5 is zero when the output v_{ds} is equal to 1. This term is introduced to ensure the source and the destination routers belong to the solution (the shortest path). The main contribution in Ali-Kamoun's paper [2] is that synaptic conductances are constant, given by

$$T_{Xi,Yj} = \mu_4 \delta_{XY} \delta_{ij} - \mu_3 (\delta_{XY} + \delta_{ij} - \delta_{jX} - \delta_{iY}) \tag{10}$$

while the link costs and the information about the connection between routers are associated to the bias currents

$$\begin{aligned} I_{Xi} =& -\frac{\mu_1}{2} C_{Xi}(1 - \delta_{Xd}\delta_{is}) - \frac{\mu_2}{2} \rho_{Xi}(1 - \delta_{Xd}\delta_{is}) - \frac{\mu_4}{2} + \frac{\mu_5}{2} \delta_{Xd}\delta_{is} \\ =& \begin{cases} \dfrac{\mu_5}{2} - \dfrac{\mu_4}{2}, & \text{for } (X,i) = (d,s) \\ -\dfrac{\mu_1}{2} C_{Xi} - \dfrac{\mu_2}{2} \rho_{Xi} - \dfrac{\mu_4}{2}, & \text{otherwise} \end{cases} . \end{aligned} \tag{11}$$

In this way the neural network SP algorithm becomes very attractive for real time processes. Ali and Kamoun [2] applied this algorithm to optimize delay in computer networks under different network topologies and link costs. Recently, we applied similar algorithm [7] to the routing in full-connected bi-directional (full-duplex) networks, assuming both the symmetric $(C_{ij} = C_{ji})$ and non-symmetric $(C_{ij} \neq C_{ji})$ cost matrix, that means, assuming the same and different link costs (transport conditions) depending on the direction of traffic flow.

Our further investigation was addressed to the routing with reduction of packet loss. Namely, each link is characterized by its *capacity*, K_{ij} (in data units per second). Note that the link capacity from router i to router j and in opposite direction should not be the same, in general. On the other hand, each link has its *traffic density*, G_{ij} (also in data units per second), describing the actual traffic flow in particular direction. Certainly, the traffic density may not exceed the link capacity; i.e., it must be satisfied $K_{ij} \geq G_{ij}$. If, in some circumstances, incoming packets overflow the input buffer, these packets would be lost. In order to minimize this effect we introduce a new term observing the free space in a link capacity. This term is the difference $(K_{ij} - G_{ij})$ and the routing have to find a path with enough free space in link capacity. Since the neural network is designed to minimize the energy function, actual term in energy function have to be of the form $[1-(K_{ij}-G_{ij}]$. Following this reason we introduced new term, μ_6, in biases

$$\begin{aligned} I_{Xi} =& -\frac{\mu_1}{2}C_{Xi}(1-\delta_{Xd}\delta_{is}) - \frac{\mu_2}{2}\rho_{Xi}(1-\delta_{Xd}\delta_{is}) \\ & -\frac{\mu_4}{2} + \frac{\mu_5}{2}\delta_{Xd}\delta_{is} + \frac{\mu_6}{2}\Big[1-(K_{Xi}-G_{Xi})\Big](1-\delta_{Xd}\delta_{is}) \\ =& \begin{cases} \dfrac{\mu_5}{2} - \dfrac{\mu_4}{2}, & \text{for } (X,i) = (d,s) \\ -\dfrac{\mu_1}{2}C_{Xi} - \dfrac{\mu_2}{2}\rho_{Xi} - \dfrac{\mu_4}{2} + \dfrac{\mu_6}{2}\Big[1-(K_{Xi}-G_{Xi})\Big] \end{cases} . \end{aligned} \tag{12}$$

The term μ_6 minimizes the possibility of packet loss under the constraint $K_{ij} \geq G_{ij}$. If G_{ij} exceeds the link capacity, K_{ij}, this router is removed from the algorithm by setting an appropriate term ρ_{ij} into the matrix $\boldsymbol{\rho}$ defining the existence of links between routers (the routers connectivity) to the value of 1.

By embedding (10) and (12) into the state equations (3) the following expression suitable for computer simulation is obtained

$$\begin{aligned} C_i\frac{du_i}{dt} =& -\frac{u_i}{R_i} - \frac{\mu_1}{2}C_{Xi}(1-\delta_{Xd}\delta_{is}) - \frac{\mu_2}{2}\rho_{Xi}(1-\delta_{Xd}\delta_{is}) \\ & -\mu_3\sum_{Y\neq X}(v_{XY}-v_{YX}) + \mu_3\sum_{Y\neq X}(v_{iY}-v_{Yi}) \\ & -\frac{\mu_4}{2}(1-2v_{Xi}) + \frac{\mu_5}{2}\delta_{Xd}\delta_{is} + \frac{\mu_6}{2}\Big[1-(K_{Xi}-G_{Xi})\Big](1-\delta_{Xd}\delta_{is}). \end{aligned} \tag{13}$$

4 Simulation Results

Relations (13) and (2) are used for computer simulations derived in Matlab. The router positions in Descartes space, and relevant matrices defining the routers connectivity, $\boldsymbol{\rho}$, the link cost, $\boldsymbol{C}$, the link capacities, $\boldsymbol{K}$, and the traffic density, $\boldsymbol{G}$, have

been constructed. We have analyzed rather complex case of fully connected graph producing the maximal number of possible valid solutions, of the form (1), which is equal to $n!$. Links were considered as bidirectional: for cases of symmetrical transmission with $C_{ij} = C_{ji}$, as well as for more realistic case with $C_{ij} \neq C_{ji}$ (non-symmetrical case). The coefficients μ_1 to μ_6 and the amplifier coefficients, a, have been chosen for best results. Actually, we used: $\mu_1 = 950$; $\mu_2 = 2500$; $\mu_3 = 1500$; $\mu_4 = 475$; $\mu_5 = 2500$; $\mu_6 = 1000$; $a = 50$; assuming that C_i and R_i has the unit values. Very intensive simulations have been done. The results here presented are used for pointing out the algorithm itself.

Example 1. Fully-connected network with 8 routers, Fig. 2, with symmetrical link cost matrix given by (14). The optimal path between the router 1 and router 6, having the 'length' of 0.5, is 1-2-4-6 (with bolded entries in matrix $\boldsymbol{C}$), as depicted in Fig. 3. Note that this path is pure shortest one.

$$\boldsymbol{C} = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 0 & \mathbf{0.1} & 0.4 & 0.5 & 0.6 & 0.9 & 0.6 & 0.8 \\ \hline 0.1 & 0 & 0.8 & \mathbf{0.2} & 0.3 & 0.8 & 0.7 & 0.5 \\ \hline 0.4 & 0.8 & 0 & 0.5 & 0.2 & 0.6 & 0.3 & 0.8 \\ \hline 0.5 & 0.2 & 0.5 & 0 & 0.4 & \mathbf{0.2} & 0.9 & 0.4 \\ \hline 0.6 & 0.3 & 0.2 & 0.4 & 0 & 0.7 & 0.8 & 0.2 \\ \hline 0.9 & 0.8 & 0.6 & 0.2 & 0.7 & 0 & 0.5 & 0.2 \\ \hline 0.6 & 0.7 & 0.3 & 0.9 & 0.8 & 0.5 & 0 & 0.4 \\ \hline 0.8 & 0.5 & 0.8 & 0.4 & 0.2 & 0.2 & 0.4 & 0 \\ \hline \end{array} \tag{14}$$

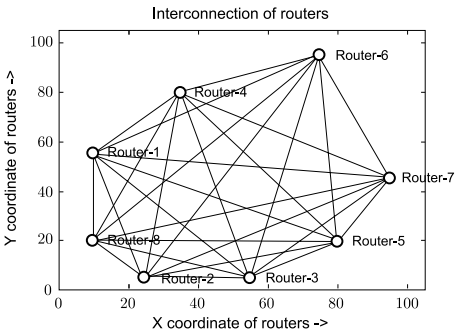

Fig. 2. A eight-routers network.

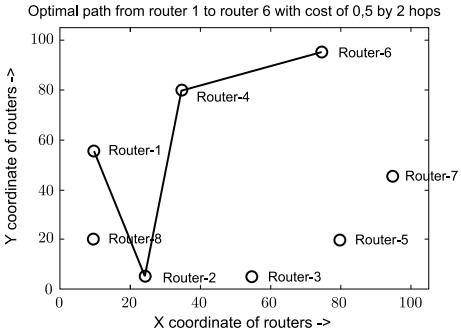

Fig. 3. Optimal path (the shortest one) from router 1 to router 6 (Example 1).

Example 2. Fully-connected eight-router network, as in Fig. 2, with non-symmetrical link cost matrix, described by (15). The optimal path between the source 1 and destination 6, is 1-3-4-7-6 (bolded entries in matrix $\boldsymbol{C}$), having the 'length' of 0.5, as represented in Fig. 4.

$$\boldsymbol{C} = \begin{array}{|cccccccc|}
\hline
0 & 0.08 & \mathbf{0.1} & 0 & 0 & 0.9 & 0.6 & 0.8 \\
0.2 & 0 & 0 & 0.9 & 0.1 & 0 & 0 & 0.6 \\
0.4 & 0 & 0 & \mathbf{0.1} & 0.1 & 0.6 & 0.6 & 0.1 \\
0.1 & 0 & 0.8 & 0 & 0 & 0.5 & \mathbf{0.1} & 0.8 \\
0 & 0.1 & 0 & 0.1 & 0 & 0.9 & 0.7 & 0.1 \\
0.8 & 0 & 0 & 0 & 0.5 & 0 & 0.4 & 0.7 \\
0.1 & 0.4 & 0.5 & 0 & 0 & \mathbf{0.2} & 0 & 0.2 \\
0 & 0.2 & 0.1 & 0.4 & 0.9 & 0 & 0.9 & 0 \\
\hline
\end{array} \tag{15}$$

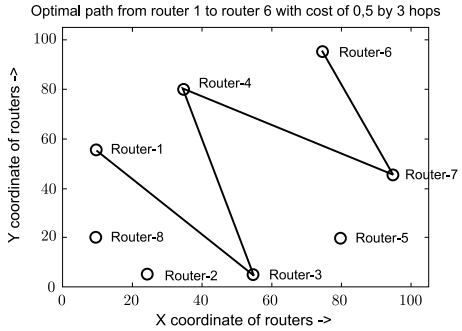

Fig. 4. Optimal path (the shortest one) from router 1 to router 6 (Example 2).

Example 3. Fully-connected network with 10 routers and non-symmetrical link cost matrix described by matrix (16). The algorithm has chosen the path 1-10-6 as optimal between routers 1 and 6 Fig. 5(a). Its length is 0.7. However, additional checking founded that the path 1-5-8-10-6 is the shortest one (bolded entries), having the length of 0.5. Such behavior of Hopfield neural network is even expected, according to initial paper [1]. Namely, the solution derived by Hopfield neural network may not be exactly the shortest path but it always belongs to one of the shortest valid paths. In our example, by changing the amplifier constants from $a = 25$ to $a = 3$, the network had found the exact shortest path, Fig. 5(b), but the computational time now was longer for nearly 15%.

$$\boldsymbol{C} = \begin{array}{|cccccccccc|}
\hline
0 & 0.8 & 0.4 & 0.5 & \mathbf{0.1} & 0.9 & 0.6 & 0.8 & 0.2 & 0.6 \\
0.1 & 0 & 0.8 & 0.2 & 0.3 & 0.8 & 0.7 & 0.5 & 0.4 & 0.7 \\
0.4 & 0.8 & 0 & 0.5 & 0.2 & 0.6 & 0.3 & 0.8 & 0.2 & 0.8 \\
0.5 & 0.2 & 0.5 & 0 & 0.4 & 0.8 & 0.9 & 0.4 & 0.8 & 0.1 \\
0.6 & 0.3 & 0.2 & 0.4 & 0 & 0.7 & 0.8 & \mathbf{0.2} & 0.4 & 0.8 \\
0.9 & 0.8 & 0.6 & 0.2 & 0.7 & 0 & 0.5 & 0.2 & 0.1 & 0.5 \\
0.6 & 0.7 & 0.3 & 0.9 & 0.8 & 0.5 & 0 & 0.4 & 0.9 & 0.2 \\
0.8 & 0.5 & 0.8 & 0.4 & 0.2 & 0.2 & 0.4 & 0 & 0.7 & \mathbf{0.1} \\
0.2 & 0.6 & 0.1 & 0.9 & 0.4 & 0.8 & 0.5 & 0.6 & 0.8 & 0.3 \\
0.4 & 0.7 & 0.8 & 0.1 & 0.1 & \mathbf{0.1} & 0.2 & 0.6 & 0.4 & 0.7 \\
\hline
\end{array} \tag{16}$$

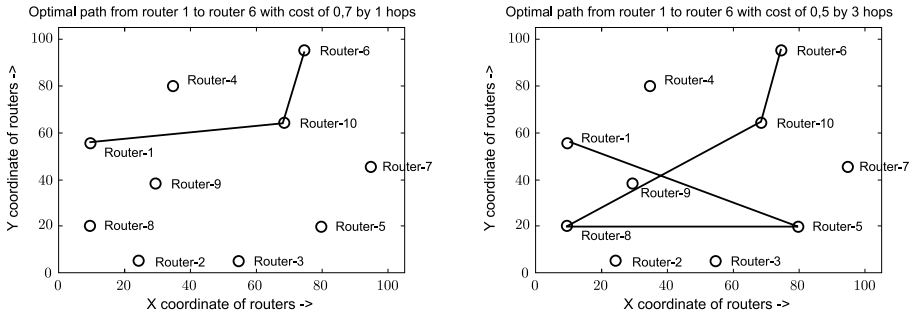

Fig. 5. (a) Initially chosen optimal path (not the shortest one) from router 1 to router 6 and (b) the shortest path obtained after changing amplifiers' constant (Example 3).

In previous examples 1-3 the algorithm was derived without the term μ_6, i.e., without the consideration of possible packet loss due to the input buffer overflow. The next example will introduce this term.

Example 4. Fully-connected network with 8 routers and non-symmetrical link cost matrix as in example 2, relation (15), and with link capacity and traffic density matrices, $\boldsymbol{K}$ and $\boldsymbol{G}$, given by (17) and (18), respectively. The optimal path between the source router 1 and the destination 6 is 1-3-7-6, Fig. 6(a), having the length of 0.9.

$$\boldsymbol{K} = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 0 & 0.2 & 0.41 & 0.8 & 0.4 & 0.8 & 0.6 & 0.1 \\ \hline 0.1 & 0.6 & 0.8 & 0.1 & 0.4 & 0.7 & 0.7 & 0.1 \\ \hline 0.8 & 0.2 & 0.5 & 0.4 & 0.8 & 0.5 & 0.9 & 0.3 \\ \hline 0.4 & 0.1 & 0.4 & 0.4 & 0.8 & 0.9 & 0.1 & 0.1 \\ \hline 0.1 & 0.5 & 0.4 & 0.5 & 0.1 & 0.1 & 0.2 & 0.6 \\ \hline 0.7 & 0.5 & 0.1 & 0.1 & 0.5 & 0.1 & 0.6 & 0.2 \\ \hline 0.6 & 0.9 & 0.7 & 0.7 & 0.3 & 1 & 0.2 & 0.4 \\ \hline 0.2 & 0.1 & 0.3 & 0.1 & 0.5 & 0.1 & 0.8 & 0.9 \\ \hline \end{array} \tag{17}$$

$$\boldsymbol{G} = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 0 & 0.1 & 0.1 & 0.1 & 0.2 & 0.9 & 0.7 & 0.2 \\ \hline 0.1 & 0 & 0.3 & 0.4 & 0.6 & 1 & 0.1 & 0.7 \\ \hline 0.5 & 0.9 & 0 & 0.8 & 0.4 & 0.6 & 0.8 & 0.7 \\ \hline 0.9 & 0.8 & 0.5 & 0 & 0.2 & 0.4 & 0.9 & 0.6 \\ \hline 0.1 & 0.1 & 0.5 & 0.7 & 0 & 0.2 & 0.6 & 0.7 \\ \hline 0.1 & 0.5 & 0.4 & 1 & 0.6 & 0 & 0.4 & 0.8 \\ \hline 0.2 & 0.5 & 0.1 & 0.9 & 0.9 & 0.9 & 0 & 0.5 \\ \hline 0.1 & 0.5 & 0.9 & 0.2 & 0.8 & 0.4 & 1 & 0 \\ \hline \end{array} \tag{18}$$

Let us assume increase of G_{13} from 0.1, as in (18), to 0.5. Now the traffic density $G_{13} = 0.5$ exceeds its link capacity $K_{13} = 0.41$. According to our algorithm, Eqs (12) and (13), the link 1-3 is excluded from the procedure. The optimal path

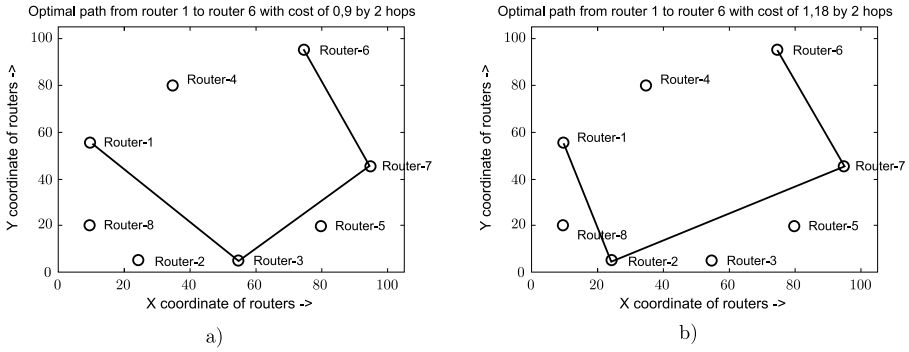

Fig. 6. (a) Optimal path from router 1 to router 6, including link capacity and traffic density matrices (b) Optimal path from router 1 to router 6 avoiding the packet loss in router 3 (Example 4).

between the source router 1 and the destination router 6 now is 1-2-7-6, Fig. 6(b), having the length of 1.18. This path is not the shortest one, but the packet loss in router 3 is avoided.

5 Conclusions

The routing problem in packet-switched communication networks can be solved as the optimization problem of the traveling salesman type. Links are to be described as appropriate 'costs', C_{ij}, containing relevant data describing transmission conditions from router i to router j (instead of simple distance between the nodes, as in case of classical TSP problem). Ali and Kamoun [2] derived efficient method for optimizing the routing in communication networks with assistance of Hopfield neural network. Recently, we have applied the Cellular Neural Networks for solving routing problems in networks of regular structure (known as Manhattan-Street Networks) [8]. Here, we have derived a new routing algorithm, inspired by Ali-Kamoun's paper, applicable for full-duplex networks of arbitrary structure. The algorithm assumes both symmetrical and non-symmetrical link cost matrices and takes into account the link capacities and traffic density. In this way the packet loss is minimized, as well. After performing intensive simulations, we have concluded that neural network always finds the optimal solution: the shortest path or some very close to this, but avoiding the packet loss due to the input buffer overflow.

The advantage of the Hopfield-type neural network is that this network can be realized as classic electronic device, assuring the very high speed due to inherent parallel processing. The data terms describing the communication network: the link 'costs', router connectivity, link capacities and the traffic density, as described in the algorithm, are associated to bias currents, not to synaptic conductance. In

this way the network becomes highly flexible and may be used even in real cases when network parameters and traffic conditions may rapidly vary in time.

References

[1] J. J. Hopfield and D. W. Tank, "'Neural' computations of decision in optimization problems," *Biol. Cybern.*, pp. 141–152, 1985.

[2] M. Ali and F. Kamoun, "Neural networks for shortest path computation and routing in computer networks," *IEEE Trans. on Neural Networks*, vol. 4, no. 6, pp. 941–953, 1993.

[3] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci.*, vol. 79, pp. 2554–2558, 1982.

[4] M. Cohen and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 13, pp. 815–826, 1983.

[5] H. Rauch and T. Winarske, "Neural networks for routing communication traffic," *IEEE Cont. Syst. Mag.*, pp. 26–30, Apr. 1988.

[6] L. Zhang and S. Thomopoulos, "Neural network implementation of the shortest path algorithm for traffic routing in communication networks," in *Proc. Int. Joint Conf. Neural Networks*, June 1989, p. II. 591.

[7] N. Kojić, I. Reljin, and B. Reljin, "Determination of optimal path using neural network," in *Proc. 48th Conf. ETRAN*, vol. 1, Čačak, Serbia and Montenegro, June 6–8, 2004, pp. 119–122, (in Serbian).

[8] P. Kostić, I. Reljin, and B. Reljin, "Cellular neural network for solving routing problems in manhattan street networks," *Int. Journal of Theoretical Electrical Eng.*, no. 6, pp. 106–113, 1996, Thessaloniki (Greece).