Neural Network based Dynamic Multicast Routing

N. Kojic¹, I. Reljin², B. Reljin³
¹ICT College of Vocational Studies, Zdravka Celara 16, 11000 Belgrade, Serbia, phone: +381 11 3290 828
²Faculty of Electrical Engineering, University of Belgrade, Bulevar kralja Aleksandara 73, 11000 Belgrade, Serbia, phone: +381 65 9044 003
³Innovation Center of the Faculty of Electrical Engineering, University of Belgrade, Bulevar kralja Aleksandara 73, 11000 Belgrade, Serbia, phone: +381 11 3370 123 nenad.kojic@ict.edu.rs

Abstract—The Hopfield neural network is suggested for dynamic multicast routing in communication network of arbitrary topology and under variable traffic conditions. A new algorithm takes into account not only the most important parameters describing the actual network state (the network topology, link and router bandwidths, estimated link delays, and the traffic density), but also the history of link/router occupancy. The goal of the paper is to find the Pareto optimal path for multicast routing case and to avoid possible packets loss, due to heavy traffic. The effectiveness of the new routing algorithm has been verified under various network topologies and traffic conditions.

Index Terms—Dynamic routing, multicast, Steiner tree problem, Hopfield neural network.

I. INTRODUCTION

Standard communication in computer networking comprises the transmission of data from one source (S) to single destination (D), which is known as the unicast (i.e., point-to-point or S-D pair) transmission. New services for entertainment and commercial applications (video-on-demand, video games, telemedicine, videoconference, triple play services, distance learning, multiservice networks, distributed databases, etc.) are based on multicasting, when one source sends the same content to a group of m destinations, (point-to-multipoint or S-mD, m≥1, case) [1]–[5]. Consequently, corresponding routing algorithms are known as the unicast or multicast routing. Historically, the routing algorithms have evolved from static routing, in which “good routes” are computed off-line, to more sophisticated dynamic routing, in which the routes are computed online, taking the actual traffic conditions into account [6], which is very important for ensuring the quality of service in IP networks [4], [5].

Most frequently, routing algorithms are based on the shortest path (SP) problem. In the simplest way, the physical path length (distances) may be considered, as used in classic optimization problem known as the traveling salesman problem (TSP). But, similarly as in road traffic, in telecommunications the shortest path is not always the optimal solution. Some other measures may be considered when determining optimal solution in routing. For instance, the number of hops between source and destination node [7], the mean transmission delay [8], [9], commercial cost, link and/or node bandwidth, average traffic density, the input buffer occupancy, etc., [1], [3].

The problem of finding the shortest path from a single source to a single destination has some well-known polynomial algorithmic solutions, such as Dijkstra’s [6] and Bellman-Ford’s [6], but the problem is computationally very hard, particularly in large-scale networks and constrained problems. In their well-known paper Hopfield and Tank [10] introduced an analog neural network (known as the HNN) suitable for solving different constrained and computationally hard optimization problems – among others the well-known TSP problem. Note that in real communication networks, due to particular limitations dictated by technological and/or commercial requirements, some parameters affecting the routing path are fixed. In this case, instead of standard optimization method more appropriate is to use Pareto optimization [11]. Moreover, in multicast routing, instead of the shortest path, the minimum Steiner tree (MST) problem [6] should be resolved for the given network.

In this paper the dynamic multicast routing algorithm, which takes into account many different parameters describing actual state within the network, such as the network topology (including its possible changes), node and link bandwidth, traffic dynamics, and link delays, is considered. The new algorithm uses Hopfield-like neural network for optimizing, in Pareto sense, the cost of the total Steiner tree [6], and minimizing the total cost wihit regard to different network parameters.

II. HOPFIELD NEURAL NETWORK IN ROUTING

The first step in the Hopfield approach to the optimization problem is to formulate an appropriate energy function E which depends on the outputs v={v_j} of nonlinear devices (cells, or neurons in NN terminology) driven by voltages...
u={u_t}, and synaptic weights \( T_{ij} \). Synaptic weights connect neurons and enable collective functioning of the network. Indices \( i \) and \( j \) relate to neurons. The energy function is composed by two parts: a term that needs to be minimized leading to the optimization of the energy function, and some penalty terms (constraints). The role of penalty terms is to increase the value of energy function if any of the problem constraints is not satisfied. From the energy function the dynamical equations describing the network state are derived as \( du/dt = -dE/d\bar{u} \) from which follows that the steady-state response (\( u=\{u_t\}=\text{const} \)) corresponds to the minimum of the network energy. Significant advantage of Hopfield approach is that the neural network can be realized by standard electronic components: amplifiers, resistors and capacitors. Due to intrinsic parallel work of neural network, hardware realization of HNN is capable to find near optimal solution of the TSP problem in a very short time [10].

Optimal routing in communication network may be considered as the SP problem and can be resolved by HNN. For this purpose instead of physical distances, \( d_{ij} \), between nodes \( i \) and \( j \), some other attributes called the link costs, \( C_{ij} \), are associated to links, describing the transmission conditions between nodes [12]. The goal is to minimize the total cost, \( C_{\text{tot}} \),

\[
C_{\text{tot}}= C_{ij} + C_{jk} + \ldots + C_{ID},
\]

i.e., to find the “minimal-length path” between the source and destination node going through internodes \( i, j, k, \ldots, r \). The use of HNN in finding the shortest path between a given \( S-D \) pair in communication network with \( n \) nodes was described in several papers.Rauch and Winarske [13] proposed a HNN of size \( n \times m \), where \( m < n \) denotes the number of nodes forming the path. The serious limitation in their representation is that it requires fixed and a priori known number of nodes, \( m \), in the shortest path. This drawback was corrected in [14], but for fixed \( S-D \) pair. For another \( S-D \) pair, the neural configuration has to be changed, which is hardly impractical. Furthermore, the cost terms in the energy function, which correspond to synaptic conductances, are quadratic. Consequently, such a solution is not appropriate for a real case because, in practice, the link conditions (costs) are time varying and, thus, the conductances in corresponding neural network have to be changed, too.

Significant improvements in routing using HNN are suggested by Ali and Kamoun [12]. For communication network with \( n \) nodes their computational network uses \( n(n-1) \) neurons – the diagonal elements in the connection matrix \( n \times n \) are removed. The squared matrix corresponds to neurons into the HNN, with locations \((x,t)\) relating to rows and columns, respectively.

For solving routing problem Ali and Kamoun proposed the energy function of the form

\[
E^{AK} = \frac{\mu_k}{2} \sum_{(i,j) \in (D,S)} C_{ij} (1-v_{ij}) + \frac{\mu_k}{2} \sum_{(i,j) \in (D,S)} \rho_{ij} (1-v_{ij}) + \frac{\mu_k}{2} \sum_{i,j} (\sum_{\alpha=1}^{m} v_{ij} - \sum_{\alpha=1}^{m} v_{ij})^2 + \frac{\mu_k}{2} \sum_{i,j} v_{ij} (1-v_{ij}) + \frac{\mu_k}{2} (1-v_{DS}),
\]

where terms \( C_{ij} \) denote the link costs from node \( i \) to node \( j \), the terms \( \rho_{ij} \) describe physical connection between nodes (the value of \( \rho_{ij} \) is set to 1 if nodes are not connected, and 0 for connected nodes), while \( v_{ij} \) are neurons’ outputs. Coefficients \( \mu_{k} \), \( k=1, \ldots, \nu \), control the influence of particular terms on the energy function. The term associated with \( \mu_k \) minimizes the total cost of the route \( S-D \), while other terms relate to constraints: \( \mu_k \) prevents the inclusion of nonexisting links into the solution; \( \mu_k \) determines valid paths, \( \mu_k \) force the state of the neural network to converge to one of the stable states – corners of the hypersphere defined by \( v_{ij} \in \{0,1\} \). The state \( v_{ij} \) is close to 1 for node belonging to the valid path otherwise the state is close to 0. The term associated to \( \mu_k \) is introduced to ensure the source and the destination nodes belong to the valid solution (i.e., to the shortest \( S-D \) path).

The main contribution in Ali-Kamoun’s paper [12] is that synaptic conductances in HNN are constant, given by

\[
T_{x,y}^{AK} = -\mu_1 (\delta_{xy} + \delta_{yx} - \delta_{ty} - \delta_{yt}) + \mu_2 \delta_{xy} \delta_{yx},
\]

where \( \delta_{xy} \) are the Kronecker delta, having values of \( \delta_{xy}=1 \) and \( \delta_{xy}=0 \) (\( i \neq j \)), while the link costs and the information about the connection between nodes are associated to bias currents

\[
I_{x,y}^{AK} = -\frac{\mu_3}{2} C_{xy} (1-\delta_{yD} \delta_{yS}) - \frac{\mu_4}{2} \rho_{xy} (1-\delta_{yD} \delta_{yS}) - \frac{\mu_5}{2} C_{xy} \delta_{yD} \delta_{yS} + \frac{\mu_6}{2} C_{xy} \delta_{yD} \delta_{yS} - \frac{\mu_7}{2} \rho_{xy} \delta_{yD} \delta_{yS} - \frac{\mu_8}{2},
\]

for \((x,y) = (D,S)\),

\[
I_{x,y}^{AK} = \begin{cases} -\frac{\mu_3}{2} C_{xy} \delta_{yD} \delta_{yS} + \frac{\mu_6}{2} C_{xy} \delta_{yD} \delta_{yS} - \frac{\mu_7}{2} \rho_{xy} \delta_{yD} \delta_{yS} - \frac{\mu_8}{2}, & \text{otherwise.} \end{cases}
\]

In this way their SP algorithm becomes very attractive for real time processing, because bias currents may be easily online controlled. Ali and Kamoun [12] successfully applied this algorithm to the minimum delay routing problem in computer networks under different network topologies and link costs. Several modifications and improvements of their approach have been introduced by Park and Choi [15], for the multicast routing problem. However, as noticed in [16], despite the improvements their method still has several drawbacks, for instance, multiple convergence points and poorer behavior with increasing number of graph nodes.

### III. THE NEW ALGORITHM FOR MULTICAST ROUTING BY NEURAL NETWORK

#### A. Dynamic Unicast Routing by Neural Network

For better understanding the new algorithm for dynamic multicast routing we will describe, first, similar algorithm applied to unicast routing. In [17] we suggested the following energy function
\[ E^{new} = E^{AK} + \frac{\mu_i}{2} \sum_{x \neq x} (1 - M_{x,i}) E_{ij} + \frac{\mu_i}{2} \sum_{x \neq x} r_{ij} V_{ij} + \frac{\mu_i}{2} \sum_{x \neq x} (1 - S_{x,i}) V_{ij}, \]
\[ \text{where the first term, } E^{AK} \text{ corresponds to that of the Ali and Kamoun’s, given by (2), while the new terms are introduced for avoiding the packet loss due to heavy and variable traffic.} \]

The term with \( \mu_i \) relates to the traffic conditions. Each link \( x \)-\( i \) is characterized by its capacity, \( K_{x,i} \), meaning by maximal data flow – the maximal number of data units per second (i.e., the bandwidth). On the other hand, each link has the actual data flow in particular direction, i.e., the traffic density, \( G_{x,i} \), expressed also in terms of data units per second. For reliable traffic the condition \( K_{x,i} \geq G_{x,i} \) must be satisfied, otherwise packets exceeding the input buffer will be lost. For avoiding this effect we introduce the difference \( (K_{x,i} - G_{x,i}) \), which represents the link capacity margin
\[ M_{x,i} = (K_{x,i} - G_{x,i}), \]

\( i.e., \) the free space in link capacity. The routing policy has to find a path with enough free space in link capacity (high value of \( M_{x,i} \)). If \( G_{x,i} \) approaches to \( K_{x,i} \) this link may be overloaded and thus the HNN should distimulate this link to be included into the final path. For that reason the term \( (1 - M_{x,i}) \) is used in the energy function (5).

Next parameters characterizing the links are their delays, \( \tau_{x,i} \). The term associated with \( \mu_i \) is introduced to minimize total delivery time of packets on the final path, for a given S-D pair. By minimizing this term an overall throughput in the network is increased and thus the network efficiency as well.

The last parameter in the energy function (5), which is controlled by \( \mu_i \), we are calling as the statistics of link occupancy. Corresponding parameter, \( S_{x,i} \), depends on the history of links occupancy in previously determined routes. Our assumption is that if some of the links are frequently used in past the probability of its overload in future increases. Data describing the link occupancy are recorded and parameters \( S_{x,i} \) are calculated in inverse sense regarding to occupancy: greater occupancy - smaller \( S_{x,i} \) (approaching to zero), while rarely used links have \( S_{x,i} \) close to unity. By using terms \( (1 - S_{x,i}) \) in energy function (5) frequently used links are distimulated to be included in the final path, in order to avoid the possible packet loss.

Except the optimization process for finding the final route under different conditions, as described above, we can apply even some crisp logic determining permitted paths under additional constraints, as follows. This crisp logic relates to the node connectivity matrix \( \rho = [\rho_{x,i}] \) associated to the constant \( \mu_i \) in (2). As noted before terms \( \rho_{x,i} \) describe network topology having values \( \rho_{x,i} = 1 \) or \( \rho_{x,i} = 0 \) (for non-existing and existing links, respectively). But, we can assume even some virtual node connectivity in order to avoid possible packet loss due to heavy and changeable traffic. For instance, if in some time periods the traffic density exceeds the node capacity, i.e., if \( M_{x,i} \leq 0 \), the term \( \rho_{x,i} \) will be temporarily set to \( \rho_{x,i} = 1 \), irrespectively of the physical node connection. Furthermore, we can consider the maximal allowed link delay, \( \tau_{\text{max}} \). Such a parameter may be important in some cases, for instance, for real-time video streaming with guaranteed QoS. If the transmission time (delay) \( \tau_{x,i} \) through some particular link exceeds \( \tau_{\text{max}} \) this link may be temporarily excluded from the path by setting \( \rho_{x,i} = 1 \). Finally, if it is necessary to hardly exclude the link \( x \)-\( i \) from the final path, network administrator can externally set the value \( S_{x,i} = 0 \), producing \( \rho_{x,i} = 1 \). The described crisp logic can be expressed by
\[ \rho_{x,i} = \begin{cases} 1, & \text{if } M_{x,i} \leq 0 \text{ or } \tau_{x,i} \geq \tau_{\text{max}} \text{ or } S_{x,i} = 0, \text{ as initially, otherwise.} 
\end{cases} \]

After forming the energy function, the bias current in our algorithm is defined as:
\[ I^{new}_{x,i} = I^{AK}_{x,i} - \frac{\mu_i}{2} (1 - M_{x,i}) (1 - \delta_{x,i} \delta_{\text{in}}) - \frac{\mu_i}{8} \tau_{x,i} (1 - \delta_{x,i} \delta_{\text{in}}) - \frac{\mu_i}{2} (1 - S_{x,i}) (1 - \delta_{x,i} \delta_{\text{in}}) \]

or, more precisely
\[ I^{new}_{x,i} = \begin{cases} \frac{\mu_i}{2} (C_{x,i} - \mu_i) - \frac{\mu_i}{2} \rho_{x,i} - \frac{\mu_i}{2} (1 - M_{x,i}) - \frac{\mu_i}{2} \tau_{x,i}, & \text{for } (x,i) = (D,S), \\
\frac{\mu_i}{2} (1 - S_{x,i}), & \text{otherwise.} 
\end{cases} \]

By embedding (5)–(9) into the network state equations the following expression, suitable for computer simulation, is suggested in [17]
\[ \frac{d\rho_{x,i}}{dt} = \frac{n_{x,i}}{\tau_{x,i}} \left( C_{x,i} (1 - \delta_{x,i} \delta_{\text{in}}) - \frac{\mu_i}{2} \rho_{x,i} (1 - \delta_{x,i} \delta_{\text{in}}) - \mu_i \sum_{Y \neq x} \delta_{Y,i} (v_{x,i} - v_{Y,i}) - \mu_i \sum_{Y \neq x} \delta_{Y,i} (v_{x,i} - v_{Y,i}) - \mu_i (1 - 2 \tau_{x,i}) - \frac{\mu_i}{2} \delta_{x,i} \delta_{\text{in}} - \mu_i \left( (1 - M_{x,i}) (1 - \delta_{x,i} \delta_{\text{in}}) - \frac{\mu_i}{2} \tau_{x,i} (1 - \delta_{x,i} \delta_{\text{in}}) \right) \right) \]

The term \( \tau_{x,i} = R \cdot C_{x,i} \) in (10) denotes the cell’s time constant which provides the network dynamics.

B. Dynamic Multicast Routing by Neural Network

The NN algorithm described by (5)–(10) was successfully applied to unicast routing in large networks under different traffic conditions [17], [18]. In multicast routing the goal is to optimize \( m \) paths from the source node \( S \) to the set of destination nodes: \( D^m = \{ D_1, D_2, ..., D^m \} \). In general, paths \( S-D_1, S-D_2, ..., S-D^m \) are not independent (several paths may share the same links). For the given destination \( D^m \) it is necessary to find relations between partial \( S-D \) solutions, \( S-D_1, S-D_2, ..., S-D^m \). First, instead of the unique terms \( v_{x,i} \), describing output voltages of neurons, we will observe
voltages $v_{xi}^m$ corresponding to links belonging to the multicast tree from $S$ to $D_i^m$. Furthermore, the new term (function) $f^m_w(v)$, describing the influence of other routes to particular path $S-D_i$, $j=1,2,...,n$, $j \neq m$, has to be used [19]

$$f^m_w(v) = \frac{1}{1 + \sum_j v_{xi}^j}, \quad j \neq m.$$  

(11)

From the new term $f^m_w(v)$ outputs of the neurons from different routes, belonging to the same links in the network, try to cooperate to minimize the overall cost associated to different routes, belonging to the same links in the network, via minimizing additional terms in (5) controlled by constants $\mu_i$ to $\mu_S$.

The partial energy function corresponding to the $S-D_i^m$ pair now reads

$$E^m = \frac{H_i}{2} \sum_{x \neq x_s} C_{xi}\cdot f^m_{xj}(v) + \frac{H_s}{2} \sum_{x \neq x_s} P_{xi}\cdot v_{xi}^m +$$

$$+ \frac{H_s}{2} \sum_{x \neq x_s} (v_{xi}^m - \sum_{i \neq x} v_{xi}^m)^2 + \frac{H_s}{2} \sum_{x \neq x_s} v_{xi}^m(1-v_{xi}^m) +$$

$$+ \frac{H_s}{2} (1-\delta_{xj}) + \frac{H_s}{2} \sum_{x \neq x_s} (1-S_{xi})\cdot v_{xi}^m f^m_{xj}(v) +$$

$$+ \frac{H_s}{2} \sum_{x \neq x_s} \tau_{xj}\cdot v_{xi}^m f^m_{xj}(v) + \frac{H_s}{2} \sum_{x \neq x_s} (1-S_{xi})\cdot v_{xi}^m f^m_{xj}(v),$$  

(12)

while the overall energy function for the set of all multicast routes is given as the sum of partial energy functions

$$E = \sum_mE^m.$$  

(13)

The dynamics of $i$th neuron now becomes

$$\frac{du^m_{xi}}{dt} = u^m_{xi} \cdot \frac{\partial E^m}{\partial u^m_{xi}} =$$

$$= \frac{u^m_{xi}}{\tau_i} - \frac{\mu_i}{2} C_{xi} \cdot f^m_{xj}(v) \cdot (1-\delta_{xm}) \cdot \delta_{jS} - \frac{\mu_s}{2} P_{xi} \cdot (1-\delta_{xm}) \cdot \delta_{jS} -$$

$$- \frac{\mu_s}{2} \sum_{y \neq x} (v_{yi}^m - \sum_{i \neq y} v_{yi}^m) + \frac{\mu_s}{2} \sum_{y \neq x} (v_{yi}^m - \sum_{i \neq y} v_{yi}^m) - \frac{\mu_s}{2} P_{xi} \cdot (1-2v_{yi}^m) +$$

$$+ \frac{\mu_s}{2} \delta_{xm} \cdot \delta_{jS} \cdot K_{MDS} \cdot f^m_{xj}(v) \cdot (1-\delta_{xm} \delta_{jS}),$$  

(14)

where, for simplicity, we denote the constant $K_{MDS}$ as

$$K_{MDS} = \left[ \frac{\mu_s}{2} (1-M_{xi}) + \frac{\mu_s}{2} \tau_a + \frac{\mu_s}{2} (1-S_{xi}) \right].$$  

(15)

IV. SIMULATION MODEL AND RESULTS

By using the algorithm described in previous section we derived the simulation model. For $n$-node network we started by initial square matrices $n \times n$ describing the link costs $C = \{C_{ij}\}$, network topology $\rho = \{\rho_{ij}\}$, and other terms introduced in (5-11): $K = \{K_{ij}\}$, $G = \{G_{ij}\}$, $T = \{t_{ij}\}$, and $S = \{S_{xi}\}$. Initially, the matrix $\rho$ was defined according to the node connectivity. Parameters in matrices $G, T, S$ are updated dynamically every time when the new route is established and may change the terms in matrix $\rho$ according to (7). In all simulations we considered the possibility of nonsymmetrical link costs and other parameters, i.e., the case $C_{ij} \neq C_{ji}, K_{ij} \neq K_{ji}, G_{ij} \neq G_{ji}, t_{ij} \neq t_{ji}, S_{xi} \neq S_{ji}$ is assumed. Identical neuron cells, with $t_{xi} = t = 1$ and the same sigmoid transfer function, are used. As usual in NN simulations, neurons are initially excited randomly with small input voltages close to 0.5. Constants $\mu_{1-S}$, which control the impact of particular term on the energy function, are determined by following suggestions given in [12], [15]-[18]. Starting from defined intervals for constants $\mu_i$ in [10], [12], we first analyzed stability of energy function for $\mu_i$, considering only one input, link costs $C$. We used suggested value for these constants as in [17], [18] and refined them empirically. We also used network topology and parameters, and final solution based on Dijkstra’s algorithm in [15], to additionally modifies constants values. Based on it, we iteratively changed constants values until we get final shortest route, as Dijkstra’s algorithm found. After that, we have included matrices $K, G, T$ and $S$, with their neutral values, and again modify all constants $\mu_i$ in order to find previously defined shortest path. In our network we used the following values $\mu_i = 2500, \mu_s = 3500, \mu_G = 480, \mu = 3000$, for constants controlling the constraints, while remaining constants, controlling the minimization part of the energy function take the values $\mu_i = 1500, \mu_s = 1200, \mu = 1000, \mu_k = 1800$. Note that if these constants are equal-valued $\mu_1 = \mu_2 = \mu_3 = \mu_k$ all terms will have the same influence on the minimization of energy function. Using smaller value corresponding term will be stimulated and vice versa. We found system is stable if these constants are within the limits [600, 2000].

When we tested our algorithm we started with the unicast case. By intensive simulations over a large number of randomly chosen different networks (meaning, different number of nodes, different node connections, and different values of parameters describing other network conditions), and for randomly chosen $S-D$ pairs, we found that energy function converges very fast to stable state. As an illustration, in Fig. 1 depicts the typical plot of energy function vs. the number of iterations, $k$, for 20-node network and unicast routing. The saturation of energy function is obtained after less than 100 iterations.

After the unicast case we tested our algorithm to multicast case. We analyzed randomly chosen network topologies with different number of routers (up to $n = 90$) and randomly loaded matrices $C, G, K, T$, and $S$. As a reference, in Fig. 2 the mean and standard deviation of iterations before saturation is depicted for the case 1S-3D: nodes are randomly chosen and randomly repeated 5 times.

It is evident that our algorithm still converges very fast. In the worst case of non-realistic (90-node) network the maximal number of iterations was less than 1400 (800 in
average), while for reasonable sized network (up to 40 nodes) the maximal number of iterations does not exceed 650. The efficiency of our algorithm was compared with already published algorithms for multicast routing. In their paper [15] Park and Choi analyzed the network as in Fig. 3 with 20 nodes and 45 bidirectional links described by their normalized costs. The 1S-3D case is assumed: the source router is S=2 and destinations are D={9,15,20}. Park and Choi considered only the link cost minimization and compared their results with those after applying Dijkstra’s SP, Kruskal’s MTS, and Ali-Kamoun’s [15]. Their results are presented in Table I – first four rows. In our algorithm, when minimizing only the link costs (expressed by matrix C) the entries in matrices K, T and S, are chosen to be neutral, with values of 0.5, while the matrix G is filled by zeros. In this case our algorithm found as optimal the routes depicted in Fig. 4. Determined routes are exactly the same as obtained by closed-form Dijkstra’s SP algorithm and the NN algorithm by Park and Choi [15]. Corresponding costs of partial routes are listed in the last row of Table I. Note that Park and Choi presented route costs in the form 2→9, 9→15 and 15→20 (left side of columns in Table I). We calculated and presented also the costs of routes from the source (node 2) to individual destinations (9, 15, 20) – right side of columns in Table I.

The possibility to find an optimal route is a main descriptor of the routing algorithm quality. The second descriptor is the execution time (processing speed). We compared our algorithm with other algorithms analyzed in [15] in terms of the number of iterations before saturation. The number of iterations for Ali-Kamoun’s, Park-Choi’s and our algorithm for routes S:2-D:9, S:9-D:15 and S:15-D:20 are approximately (7000, 3500, 140), (15000, 8000, 185) and (6500, 2000, 120) respectively. Our method was significantly faster than Ali-Kamoun’s and Park-Choi’s methods. In our approach, even for more than 50 randomly chosen 1S-3D routing in 20 node network, the average number of iterations was less than 200.

The execution time is compared also with results recently presented in Araujo et al. [16]. They analyzed 40-node network and random S-D pairs in unicast routing, and compared their results with those of Park and Choi. Comparative results are listed in Table II.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Connection</th>
<th>Park and Choi</th>
<th>Araujo</th>
<th>Our method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2→9</td>
<td>1098</td>
<td>1628</td>
<td>7489</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>9→15/2→15</td>
<td>1475/1235</td>
<td>1011/3646</td>
<td></td>
</tr>
<tr>
<td>Kruskal</td>
<td>15→20/2→20</td>
<td>671/1050</td>
<td>671/1236</td>
<td></td>
</tr>
<tr>
<td>Ali-Kamoun</td>
<td></td>
<td>1284</td>
<td>506380</td>
<td></td>
</tr>
<tr>
<td>Park-Choi</td>
<td></td>
<td>1946</td>
<td>1098</td>
<td></td>
</tr>
<tr>
<td>Our algorithm</td>
<td></td>
<td>1475/1235</td>
<td>671/1050</td>
<td></td>
</tr>
</tbody>
</table>

Note: * Park and Choi considered link costs 2→9, 9→15 and 15→20. Additionally, we presented link costs 2→15 and 2→20.

Except the shortest path(s) determination, our algorithm successfully responded to dynamic network conditions (described by link capacity margins, \( M_{li}=\left(K_{ci}-G_{ci}\right) \), and statistics of network occupancy, \( S_{li} \), and avoided links with large delays – according to new terms in energy function (5) and (11). Since similar results are not reported yet in literature we will present in Fig. 5 only some characteristic examples, describing the capability of our method. As a reference, the routes when assuming only link costs (as in Fig. 4) are depicted as thin solid lines. First, we included node bandwidths by randomly loaded matrix K with
elements within the range [0.45, 1.0]. One solution was
drawn by wide solid lines in Fig. 5-a. By inspecting the
matrix \( \mathbf{K} \) we found the bottleneck was the link 13-12
(belonging to previous routes to destinations 15 and 20),
having now the value \( K_{13,12} = 0.46 \) and thus the algorithm
avoided this bottleneck finding alternative routes.

Next example includes the traffic density, by matrix \( \mathbf{G} \),
also randomly loaded with values within [0, 0.8]. One
solution was drawn by dotted lines in Fig. 5(a). Now for
links 3-7 and 13-20 the link capacity margins \( M_{i,j}(G_{i,j}) \)
become negative and, according to constraint (7) these links
are hardly excluded.

Fig. 5. Multicast routes 2→9(15,20) under different network conditions:
(a) Based on matrices \( \mathbf{C} \) and \( \mathbf{K} \) (wide solid lines) and on matrices \( \mathbf{C}, \mathbf{K} \)
and \( \mathbf{G} \) (dotted lines); (b) Based on matrices \( \mathbf{C}, \mathbf{K}, \mathbf{G} \) and hardly excluded
links 12-16 and 11-15 (dots) and with addition of time delay matrix \( \mathbf{T} \)
(dash).

In Fig. 5(b) by dot lines are drawn routes with fixed
matrices \( \mathbf{C}, \mathbf{K} \) and \( \mathbf{G} \) as previous when links 12-16 and 11-
15 are hardly excluded, while by dash line are drawn routes
when delay matrix \( \mathbf{T} \) is included, with the constraint \( \epsilon_{max} = 0.45 \).
The main difference now is in the route 2→9. In
previous route 2-3-7-10-9 the total delay was 1.101 while
the new route 2-1-4-8-9 has the delay 0.875.

V. CONCLUSIONS

The paper considers the dynamic multicast routing in
large communication networks. A new algorithm based on
Hopfield neural network has been proposed. It takes into
account parameters relevant for such a complex problem.
Taking into consideration different network topologies,
including the changeable topology due to unpredictable
user’s connection to a multicast group, an overall cost
function is defined. The HNN energy function is derived to
include different parameters such as link distances (costs),
bandwidth, estimated link delays, link density and maximal
allowed link delays. We introduced also the link occupancy
statistics because it is expected that, if some links are used
frequently, this link may be overloaded and packets will be
lost. By minimizing the energy function Pareto optimal paths
have been found.

The simulation results are equal and even better to similar
ones, presented in available literature. Moreover, our
method is significantly faster than others, which is approved
through the convergence time, describing by the number of
iterations before the energy function saturation.

REFERENCES

Villaseñor V. Rangel, “Analysis of Topological and Geographical
Multicast Routing Algorithms on Wireless Ad Hoc Networks”,
Elektronika ir Elektrotechnika (Electronics and Electrical

Packet Switches”, IEEE Communication Magazine, vol. 40, no. 11,
http://dx.doi.org/10.1109/MCOM.2002.1046996

Adaptive Carrier Sense for Video Applications in Wireless Ad-hoc
Networks”, Elektronika ir Elektrotechnika (Electronics and Electrical
Engineering), no. 6, pp. 37–42, 2011.

Performance Evaluation”, Elektronika ir Elektrotechnika

Providing Quality of Service in Multiservice IP Networks”, Acta
Universitatis, Series: Electronics and Energetics, no. 17, pp. 261–


Samcovic, “The Analysis of Network Performance with Different
Routing and Interconnection Methods”, Elektronika ir
Elektrotechnika (Electronics and Electrical Engineering), no. 2,
pp. 43–46, 2011.

the Variable Throughput Rotes”, Elektronika ir Elektrotechnika

Packet Transmission Networks for the Service Quality”, Elektronika
ir Elektrotechnika (Electronics and Electrical Engineering), no. 2,


and routing in computer networks”, IEEE Trans. on
http://dx.doi.org/10.1109/72.288689


[14] L. Zhang, S. Thomopoulos, “Neural network implementation of the
shortest path algorithm for traffic routing in communication networks”,
[Online]. Available: http://dx.doi.org/10.1109/IJCNN.1989.118375

routing algorithm for communication network”, in Proc. of the Int.

path computation”, IEEE Trans. Neural Networks, vol. 5, no. 12,
http://dx.doi.org/10.1109/72.950136

[17] N. Koic, I. Reljin, B. Reljin, “Neural network for optimization of
routing in communication networks”, FACTA Universitas, Series:
[Online]. Available: http://dx.doi.org/10.2298/FUEE0602217K

[18] N. Koic, I. Reljin, B. Reljin, “Neural network for finding optimal
path in packet – switched network”, in Proc. of the 7th Seminar

approach to multicast routing in real-time communication networks”,
in Proc. of the ICNP 95, 1995, pp. 332–339.