Proceedings of the 2007 IEEE Symposium on
Foundations of Computational Intelligence (FOCI 2007)

A Novel Approach for a Routing Algorithm Based on a Discrete Time Hopfield Neural Network

C. J. A. Bastos-Filho
Department of Computing Systems, UPE
50720-001
Recife - PE – Brazil
cjabf@dsc.upe.br

R. A. Santana
Department of Computing Systems, UPE
50720-001
Recife - PE - Brazil
Robson_poli@yahoo.com.br

A. L. I. Oliveira, *IEEE Senior Member*
Department of Computing Systems, UPE
50720-001
Recife - PE - Brazil
alio @dsc.upe.br

***Abstract*—This articles proposes a new approach to accelerate the routing algorithm based on Hopfield Neural Network. We showed that one can calculate the best route in terms of cost in a network using a discrete equation instead of the common used differential formulation. We also demonstrated that the formulation based on discrete parameters outperforms the well known formulation in terms of simulation time.**

***Index Terms*— Communication network, Hopfield neural network, Shortest path, Routing.**

I. Introduction

Routing algorithms have been hardly discussed in the scientific community, mainly because the routing process impacts drastically on communications networks performance. Ideal routing algorithm comprises finding the best path between source and destination nodes, enabling high quality transmission and avoiding penalties caused by physical layer impairments. There are different ways to find the optimal route. Some algorithms determine the routes based on the shortest path (SP) [1], minor delay [2], higher signal-to-noise ratio (in All-Optical Networks case) [3], load balance [4], among others.

Moreover, to maintain the Quality of Service (*QoS*) computations have to be carried out in real time and should be adaptative. To provide this flexibility, many techniques from Computational Intelligence (CI) have been tested. The most used techniques are Artificial Neural Networks (*ANN*) [2,4-10], Ant Colony Optimization [11-12], Genetic Algorithms [13-14], and Hybrid algorithms combining those techniques [15-16].

ANN are very good candidates for solving the problem due to its high computational speed [2]. Hopfield and Tank described an ANN with feedback configuration suitable for solving constrained optimization problems, especially the Traveling Salesman Problem (TSP) [5]. Therefore, this ANN configuration is called Hopfield neural network (*HNN*). The use of HNN to find the shortest path between two nodes in a communication network was initiated by Rauch and Winarske [6]. However, this propose requires a prior knowledge of the network topology. To outperform this limitation, Ali and Kamoun [2] proposed a novel adaptive algorithm, where the weight matrix just carries convergence information. The information about the link cost and the topology is added through the bias as shown in Fig. 1. Additional papers report techniques to avoid involved loops and problems in the algorithm convergence [7-8]. However these algorithms are based on the same differential equation proposed by Ali and Kamoun [2].

In this paper, we propose a new approach based on discrete and finite difference equation to speed up the convergence of the HNN. The rest of the paper is organized as follows. In section II, we describe the ANN based routing algorithm proposed by [2]. In section III, we show our contribution on the solution method and in section IV we provide a detailed description of the proposed algorithm and of a software tool that we have developed for simulations. In section V, we present the simulation results and compare our approach based on difference equations to the formulation given by [2]. In section VI, we present our conclusions.

II. Hopfield Neural Networks For Routing in Communications Networks

The block diagram of the Hopfield neural network (HNN) is depicted in Fig. l. The processing elements are the neurons and they are full-connected, *i.e.* every output of each neuron is connected to inputs of all other neurons via synaptic weights. To solve routing problems in communication networks, Ali and Kamoun [2] proposed that each link in the communication network between two adjacent nodes has one neuron associated. For example, a link from node *x* to node *i* is associated to a neuron that have the *xi* description. The output of a neuron V_{xi}, depends on its input U_{xi}, according the sigmoidal function as expressed in (1). Notice that the input of each neuron corresponds to the sum of all the outputs of the neurons of the HNN multiplied by a factor represented by the matrix $T_{xi,yj}$ added to an external bias I_{xi}. The $T_{xi,yj}$ element represents the synaptic weights connecting the output of the neuron *yj* to the sum point in the input of the neuron *xi*.

$$V_{xi} = \frac{1}{1 + e^{-\lambda_{xi} U_{xi}}} \qquad \forall (x,i) \in \overline{N} \mathrm{X} \overline{N} \,/\, x \neq i \tag{1}$$

The parameter λ_{xi} determines the computation time to convergence and the correctness of the algorithm. In our

simulations we used the same λ_{xi} for all neurons in the HNN. Therefore, we call this parameter λ in the rest of the paper, instead of λ_{xi}. The behavior of the sigmoid logistic function is shown in the Fig. 2 for different values of λ. As higher as the parameter λ is, the logistic function tends to a step function.

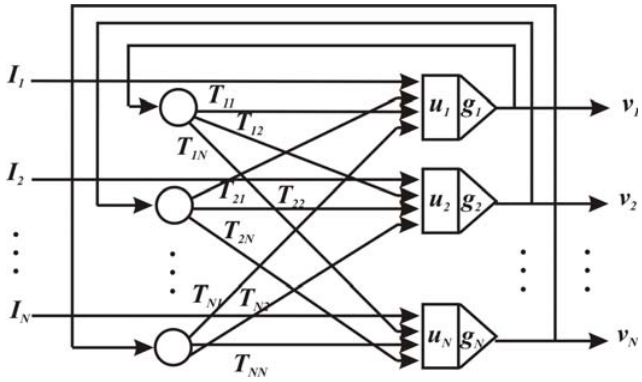

Fig. 1. Hopfield Neural Network Configuration.

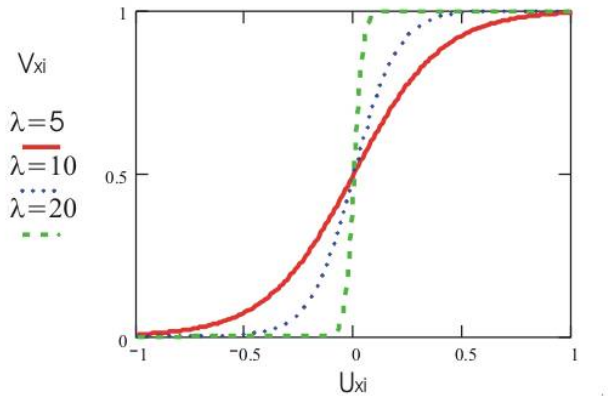

Fig. 2. Sigmoid Logistic function behavior for different values of λ.

If every link in the network has a nonnegative cost associated C_{ij}, the goal of the HNN is to find the path that minimizes the cost from some source node *s* to a destination node *d* through the communication network. Thus, the HNN should indicate a directed path as an ordered sequence of nodes connecting *s* to *d*, so the sum of all the costs connecting these nodes provides the lower possible cost. The path that provides minimum cost is defined as L_{sd}. In most cases, the cost matrix is symmetric ($C_{xi} = C_{ix}$), though exists some papers considering a asymmetric cost matrix ($C_{xi} \neq C_{ix}$) [4]. Notice that elements C_{ii} are nulls because one node cannot be connected to itself.

The matrix ρ_{xi} defines if the arc *xi* exists in the topology of the communication network used in the simulation. If the arc *xi* exists then $\rho_{xi} = 0$, otherwise $\rho_{xi} = 1$.

When the simulation converges, *i.e.* the change of every output values V_{xi} in an interactions are below a predefined criteria, an adjustment is done in each output. If an output has a value greater than 0.5 it is adjusted to "1", otherwise it is adjusted to "0". The final value of the V_{xi} will define if the arc related with the neuron *xi* belongs or not to the shortest path L_{sd}.

$$V_{xi} = \begin{cases} 1, arc_{xi} \in L_{sd} \\ 0, otherwise \end{cases} \tag{2}$$

Beside this, each cell can be externally excited by input bias I_{xi}. These system inputs can be used to set the general level of excitability of the whole network and represent the actual data provided by the user to the neural network. We used the following expression for the bias [2]:

$$\begin{aligned} I_{xi} = &-\frac{\mu_1}{2} C_{xi}\left(1-\delta_{xd}\delta_{is}\right) - \frac{\mu_2}{2}\rho_{xi}\left(1-\delta_{xd}\delta_{is}\right) \\ &-\frac{\mu_4}{2} + \frac{\mu_5}{2}\delta_{xd}\delta_{is} \\ &\forall(x \neq i), \forall(y \neq i) \end{aligned} \tag{3}$$

where δ is the Kronecker function and μ_1, μ_2, μ_4 and μ_5 are constants.

The synaptic matrix $T_{xi,yj}$ and the energy function of the HNN are described as [2]:

$$T_{xi,yj} = \mu_4\delta_{xy}\delta_{ij} - \mu_3\delta_{xy} - \mu_3\delta_{ij} + \mu_3\delta_{jx} + \mu_3\delta_{iy} \tag{4}$$

$$\begin{aligned} E = &\frac{\mu_1}{2} \sum_{\substack{x=1 \\ (x,i)\neq(d,s)}}^{n} \sum_{\substack{i=1 \\ i\neq x}}^{n} C_{xi}V_{xi} + \frac{\mu_2}{2} \sum_{\substack{x=1 \\ (x,i)\neq(d,s)}}^{n} \sum_{\substack{i=1 \\ i\neq x}}^{n} \rho_{xi}V_{xi} \\ &+ \frac{\mu_3}{2} \sum_{x=1}^{n} \left\{ \sum_{\substack{i=1 \\ i\neq x}}^{n} V_{xi} - \sum_{\substack{i=1 \\ i\neq x}}^{n} V_{ix} \right\}^2 \\ &+ \frac{\mu_4}{2} \sum_{x=1}^{n} \sum_{\substack{i=1 \\ i\neq x}}^{n} V_{xi}\left(1-V_{xi}\right) + \frac{\mu_5}{2}\left(1-V_{ds}\right) \end{aligned} \tag{5}$$

where *E* is the energy of the HNN and μ_3 is a constant.

These parameters have specific functions on the energy function: μ_1 minimizes the total cost; μ_2 prevents nonexistent links from being included in the chosen path; μ_3 is zero for every node in the valid path; μ_4 forces the HNN to converge to a stable state and μ_5 is introduced to ensure that the source and the destination nodes belong to the solution. Ahn *et al* proposed others terms to avoid loops [7], but we did not considered them in this work.

Therefore, if the system is stable in Liapunov sense, then iterations lead to smaller output changes. As a consequence, after a convenient number of iterations the HNN reaches some

minima of the system energy. Ali and Kamoun [2] resolve the system using the differential equation described below.

$$\frac{dU_{xi}}{dt} = -\frac{U_{xi}}{\tau} + \sum_{y=1}^{n}\sum_{\substack{j=1 \\ j\neq y}}^{n} T_{xi,yj}V_{yj} + I_{xi} \tag{6}$$

Notice that the sum of second and the third terms represents the energy variation of the HNN. Therefore, one can rewrite (6) as follow:

$$\frac{dU_{xi}}{dt} = -\frac{U_{xi}}{\tau} - \frac{dE}{dV_{xi}} \tag{7}$$

Ali and Kamoun [2] use the Runge-Kutta method to solve the equation. To simplify and accelerate the resolution we propose in the next section a new approach based on a discrete difference equation.

III. HNN Approach Based on Energy Finite Difference Equation

Some authors have proposed to solve Hopfield neural networks using discrete time energy equations [17-18]. However, none of them applied it to the routing problem in communications networks. Thus, we adapted the discrete time formulation to HNN modeled to solve the routing problem. We believed that it could accelerate the calculus of routes. Therefore, instead of solving the systems using a differential equation that requires sophisticated methods (like Runge-Kutta, *i.e.*, Eq. (7)), we propose a simple approach based on a simple difference equation based on discrete time reference. The equation used to calculate the next input value of the neurons is shown below.

$$\begin{aligned} U_{xi}[n+1] &= U_{xi}[n] - AU_{xi}[n-1] - BU_{xi}[n-2] \\ &+ C\Big(\sum_{y=1}^{n}\sum_{\substack{j=1 \\ j\neq y}}^{n} T_{xi,yj}V_{yj}[n] + I_{xi}[n]\Big) \end{aligned} \tag{8}$$

where $U_{xi}[n+1]$ is the next input of the neuron *xi* calculated based on the output values of all the neurons of the network $V_{yj}[n]$, the external bias $I_{xi}[n]$ and on its own input in previous instants $U_{xi}[n]$, $U_{xi}[n-1]$ and $U_{xi}[n-2]$. A, B and C are constants that regulate the weight of the previous inputs.

IV. Algorithm Description and Simulation Tool

We developed a simulation tool in *Java* to calculate the route based on HNN that follows the flow chart shown in Fig. 3. The first step is to get the parameters μ_1, μ_2, μ_3, μ_4 and μ_5 to determine the weight matrix. After that, the simulations parameters are required. Using this data the software calculates the topology and the bias matrixes. So, the neurons are initialized (the neurons input are set with a little noise to accelerate the convergence). Default values of the simulation parameters are presented in the table I.

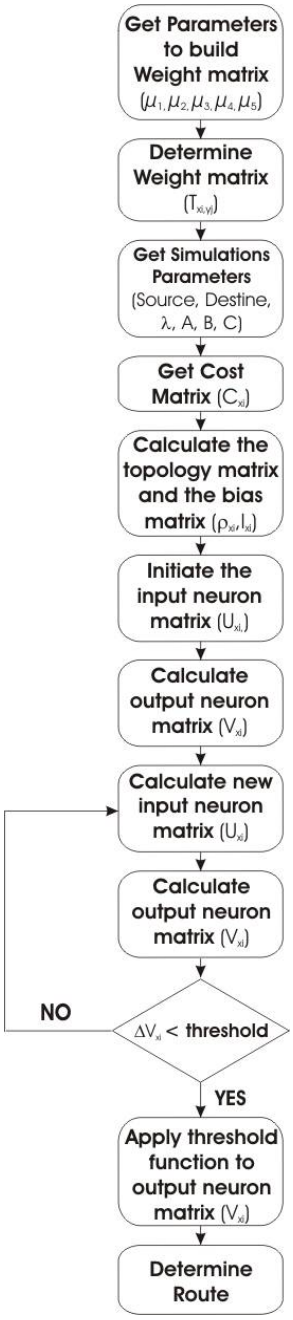

Fig. 3. Flow chart of the discrete time Hopfield Neural Network algorithm for routing in communications networks.

The value of the logistic function parameter used in the simulations is also presented in table I. It is well known that higher values for λ leads to faster convergence of the algorithm, despite it can lead to error in the route

determination.

TABLE I
PARAMETERS USED IN SIMULATIONS AND THEIR DEFAULT VALUES.

PARAMETER	VALUE (DEFAULT)
μ_1	950
μ_2	2500
μ_3	1500
μ_4	475
μ_5	2500
A	0.0001
B	0.00001
C	0.00001
λ	1

V. SIMULATIONS RESULTS

In this section we report on a number of simulations performed to compare the routing method proposed in this paper with the method proposed by Ali and Kamoun [2]. The routing algorithms based on Hopfiel neural networks are also compared to Dijkstra's algorithm [1]. We have considered three computer networks topologies in our simulations. They are depicted in Figures 3, 4 and 5, respectively.

In each simulation set we are interested in comparing the three routing methods regarding the number of optimal routes found as well as simulation time. In addition, for the methods based on HNN we have compared the number of iterations needed for convergence.

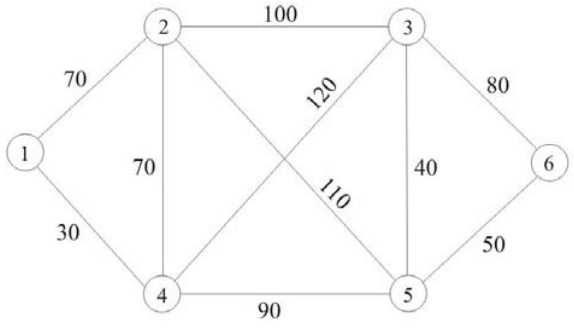

Fig. 3. Network 1: one of the computer network topologies used in the simulations. The numbers represent the costs of the links.

One simulation set for a given computer network topology consists of a number of simulations considering the three routing algorithms. In each simulation, the source and destination nodes are randomly selected. Next, Dijkstra's algorithm is used to compute the shortest path between the source and destination nodes. Finally, both the continuous version of the HNN routing method [2] and the discrete version proposed in this paper are executed independently to obtain the shortest path. The paths obtained by the HNN methods are compared to the one computed via Dijkstra's algorithm.

A simulation set is executed considering the same values for the parameters of the HNNs. After a simulation set is executed, we obtain the total simulation time for each of the HNN-based routing methods. In addition, we obtain the mean and standard deviation of the number of iterations needed for each of the HNN-based routing methods. Finally, we compute, for the HNN-based methods, the percentage of the simulations in which these method produced an optimal path, that is, a path identical to that produced by Dijkstra's algorithm for the same simulation (that is, a simulation using the same source and destination nodes).

Tables 2, 3 and 4 report some simulation results for the computer network 1, depicted in Figure 3. Each line of tables 2, 3 and 4 report the results obtained for a given simulation set. Each simulation set comprised 100 simulations with randomly selected source and destination nodes. In all simulations, the HNN-based methods employed the default values of the parameters μ_1, μ_2, μ_3 , μ_4 and μ_5 given in Table 1. We varied the values of parameters A, B, C and λ to analyze their influence on performance.

Table 2 shows the accuracy of the HNN methods with respect to the results furnished by Dijkstra's algorithm. The results of table 2 show that for most values of the parameters the HNN algorithms obtain the same paths obtained by Dijkstra's algorithm (that is, HNNs obtained 100% accuracy).

TABLE II
COMPARISON OF HNN-BASED METHODS IN TERMS OF ACCURACY (OF THE PATHS FOUND) FOR NETWOK 1 (FIGURE 3)

HNN PARAMETERS	**Accuracy**	
	HNN (Runge-Kutta)	HNN (discrete-time)
A=1e-4, B=1e-5, C=1e-5, λ=1	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=2	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=5	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=10	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=20	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=50	100.00%	100.00%
A=2e-4, B=2e-5, C=2e-5, λ=10	100.00%	100.00%
A=3e-4, B=3e-5, C=3e-5, λ=10	100.00%	100.00%
A=4e-4, B=4e-5, C=4e-5, λ=10	100.00%	100.00%
A=5e-2, B=5e-3, C=5e-3, λ=10	100.00%	82.00%

The mean (and standard deviation) number of iterations for convergence of each HNN-based method for network 1 are presented in Table 3. It can be observed that the number of iterations depends significantly on the values of the parameters. By comparing the best results obtained by each method (in bold) we conclude that the proposed method is able to converge using less iterations.

TABLE III
COMPARISON OF HNN-BASED METHODS IN TERMS OF ITERATIONS FOR CONVERGENCE FOR NETWOK 1 (FIGURE 3)

HNN PARAMETERS	Iterations	
	HNN (Runge-Kutta)	HNN (discrete-time)
A=1e-4, B=1e-5, C=1e-5, λ=1	4163.79 (1059.71)	4277.45 (998.71)
A=1e-4, B=1e-5, C=1e-5, λ=2	2569.23 (762.26)	2575.49 (664.21)
A=1e-4, B=1e-5, C=1e-5, λ=5	1265.14 (407.82)	1267.40 (400.77)
A=1e-4, B=1e-5, C=1e-5, λ=10	713.69 (241.67)	721.45 (229.99)
A=1e-4, B=1e-5, C=1e-5, λ=20	**464.93** (192.78)	460.52 (183.60)
A=1e-4, B=1e-5, C=1e-5, λ=50	10157.82 (19921.19)	8321.70 (18238.32)
A=2e-4, B=2e-5, C=2e-5, λ=10	835.0 (340.93)	487.72 (201.69)
A=3e-4, B=3e-5, C=3e-5, λ=10	778.57 (269.36)	**332.73** (126.98)
A=4e-4, B=4e-5, C=4e-5, λ=10	765.52 (262.48)	2751.17 (201.69)
A=5e-2, B=5e-3, C=5e-3, λ=10	4371.56 (1173.78)	29557.71 (24574.91)

Table 4 compares the routing methods in terms of simulation time. The results show that for most combinations of the parameters (expect that in the last line of the table) the method proposed in this paper is much faster than HNN solved with the Runge-Kutta method [2]. Moreover, the smaller simulation time for the proposed method was much smaller than that of HNN solved via Runge-Kutta (entries in bold in table 4). For network 1, the smaller simulation time obtained by Runge-Kutta was 8250ms whereas the proposed method achieved 2021ms.

TABLE IV
COMPARISON OF ROUTING METHODS IN TERMS OF SIMULATION TIMES FOR NETWOK 1 (FIGURE 3)

HNN PARAMETERS	Simulation time (ms)	
	HNN (Runge-Kutta)	HNN (discrete-time)
A=1e-4, B=1e-5, C=1e-5, λ=1	73735	26687
A=1e-4, B=1e-5, C=1e-5, λ=2	45922	16078
A=1e-4, B=1e-5, C=1e-5, λ=5	22782	7921
A=1e-4, B=1e-5, C=1e-5, λ=10	12719	4562
A=1e-4, B=1e-5, C=1e-5, λ=20	**8250**	2875
A=1e-4, B=1e-5, C=1e-5, λ=50	172890	47125
A=2e-4, B=2e-5, C=2e-5, λ=10	14641	3015
A=3e-4, B=3e-5, C=3e-5, λ=10	13797	**2031**
A=4e-4, B=4e-5, C=4e-5, λ=10	13703	16828
A=5e-2, B=5e-3, C=5e-3, λ=10	81422	192015

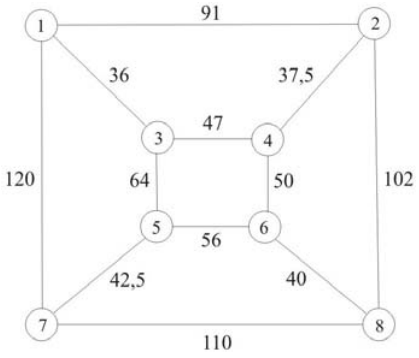

Fig. 4. Network 2: one of the computer network topologies used in the simulations. The numbers represent the costs of the links.

The simulation results for network 2 (Fig. 4) are show in Tables 5, 6 and 7, which follow the same structure of tables 2, 3, and 4, respectively. We have also tested the proposed method on a third computer network topology (network 3), which is depicted in Figure 5. The simulation results for network 3 are presented in Tables 8, 9, and 10. The same observations made for network 1 can be made for networks 2 and 3, namely, the proposed method is able to produce optimal routes with much smaller simulation times than the HNN-based method solved via Runge-Kutta [2].

TABLE V
COMPARISON OF HNN-BASED METHODS IN TERMS OF ACCURACY (OF THE PATHS FOUND) FOR NETWOK 2 (FIGURE 4)

HNN PARAMETERS	Accuracy	
	HNN (Runge-Kutta)	HNN (discrete-time)
A=1e-4, B=1e-5, C=1e-5, λ=1	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=2	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=5	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=10	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=20	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=50	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=100	100.00%	100.00%
A=2e-4, B=2e-5, C=2e-5, λ=10	100.00%	100.00%
A=3e-4, B=3e-5, C=3e-5, λ=10	100.00%	100.00%
A=4e-4, B=4e-5, C=4e-5, λ=10	100.00%	100.00%
A=5e-4, B=5e-5, C=5e-5, λ=10	100.00%	100.00%
A=1e-3, B=1e-4, C=1e-4, λ=10	100.00%	100.00%
A=5e-3, B=5e-4, C=5e-4, λ=10	100.00%	100.00%
A=5e-2, B=5e-3, C=5e-3, λ=10	100.00%	95.00%

TABLE VI
COMPARISON OF HNN-BASED METHODS IN TERMS OF ITERATIONS FOR CONVERGENCE FOR NETWOK 2 (FIGURE 4)

HNN PARAMETERS	Iterations	
	HNN (Runge-Kutta)	HNN (discrete-time)
A=1e-4, B=1e-5, C=1e-5, λ=1	4433.95 (1372.24)	4623.90 (1451.87)
A=1e-4, B=1e-5, C=1e-5, λ=2	2705.53 (836.33)	2802.07 (883.75)
A=1e-4, B=1e-5, C=1e-5, λ=5	1420.08 (385.44)	1463.95 (404.93)
A=1e-4, B=1e-5, C=1e-5, λ=10	834.88 (245.88)	858.07 (252.09)
A=1e-4, B=1e-5, C=1e-5, λ=20	495.75 (146.07)	502.75 (152.82)
A=1e-4, B=1e-5, C=1e-5, λ=50	**253.49** (77.82)	260.12 (74.1)
A=1e-4, B=1e-5, C=1e-5, λ=100	4634.77 (14266.76)	192.06 (66.89)
A=2e-4, B=2e-5, C=2e-5, λ=10	786.28 (248.24)	490.88 (160.81)
A=3e-4, B=3e-5, C=3e-5, λ=10	830.34 (246.89)	391.53 (124.89)
A=4e-4, B=4e-5, C=4e-5, λ=10	832.49 (233.22)	327.94 (94.86)
A=5e-4, B=5e-5, C=5e-5, λ=10	822.15 (242.44)	267.20 (90.22)
A=1e-3, B=1e-4, C=1e-4, λ=10	828.03 (213.48)	**162.08** (48.73)
A=5e-3, B=5e-4, C=5e-4, λ=10	805.86 (234.75)	552.26 (4970.55)
A=5e-2, B=5e-3, C=5e-3, λ=10	869.77 (247.34)	4030.47 (13558.24)

TABLE VII
COMPARISON OF ROUTING METHODS IN TERMS OF SIMULATION TIMES FOR NETWOK 2 (FIGURE 4)

HNN PARAMETERS	Simulation time (ms)	
	HNN (Runge-Kutta)	HNN (discrete-time)
A=1e-4, B=1e-5, C=1e-5, λ=1	259093	77844
A=1e-4, B=1e-5, C=1e-5, λ=2	149906	46266
A=1e-4, B=1e-5, C=1e-5, λ=5	81859	25188
A=1e-4, B=1e-5, C=1e-5, λ=10	46156	15375
A=1e-4, B=1e-5, C=1e-5, λ=20	27360	9000
A=1e-4, B=1e-5, C=1e-5, λ=50	**14359**	4672
A=1e-4, B=1e-5, C=1e-5, λ=100	257406	**2579**
A=2e-4, B=2e-5, C=2e-5, λ=10	45828	8672
A=3e-4, B=3e-5, C=3e-5, λ=10	48500	6766
A=4e-4, B=4e-5, C=4e-5, λ=10	48500	5703
A=5e-4, B=5e-5, C=5e-5, λ=10	48078	4532
A=1e-3, B=1e-4, C=1e-4, λ=10	48422	2703
A=5e-3, B=5e-4, C=5e-4, λ=10	46969	9250
A=5e-2, B=5e-3, C=5e-3, λ=10	51672	69203

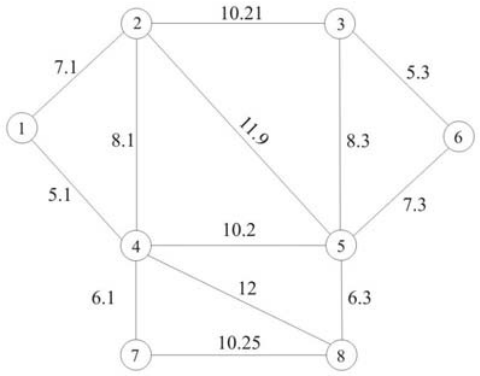

Fig. 5. Network 3: one of the computer network topologies used in the simulations. The numbers represent the costs of the links.

TABLE VIII
COMPARISON OF HNN-BASED METHODS IN TERMS OF ACCURACY (OF THE PATHS FOUND) FOR NETWOK 3 (FIGURE 5)

HNN PARAMETERS	Accuracy	
	HNN (Runge-Kutta)	HNN (discrete-time)
A=1e-4, B=1e-5, C=1e-5, λ=1	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=2	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=5	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=10	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=20	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=50	100.00%	100.00%
A=2e-4, B=2e-5, C=2e-5, λ=10	100.00%	100.00%
A=3e-4, B=3e-5, C=3e-5, λ=10	100.00%	100.00%
A=1e-3, B=1e-4, C=1e-4, λ=10	100.00%	100.00%
A=5e-2, B=5e-3, C=5e-3, λ=10	100.00%	86.00%

TABLE IX
COMPARISON OF HNN-BASED METHODS IN TERMS OF ITERATIONS FOR CONVERGENCE FOR NETWOK 3 (FIGURE 5)

HNN PARAMETERS	Iterations	
	HNN (Runge-Kutta)	HNN (discrete-time)
A=1e-4, B=1e-5, C=1e-5, λ=1	4674.93 (2433.65)	4742.56 (2572.75)
A=1e-4, B=1e-5, C=1e-5, λ=2	2663.18 (1197.25)	2692.09 (1214.80)
A=1e-4, B=1e-5, C=1e-5, λ=5	1241.41 (420.57)	1246.74 (418.65)
A=1e-4, B=1e-5, C=1e-5, λ=10	752.1 (311.52)	748.62 (307.55)
A=1e-4, B=1e-5, C=1e-5, λ=20	3884.67 (12652.35)	472.11 (152.88)
A=1e-4, B=1e-5, C=1e-5, λ=50	23597.75 (24862.92)	10874.62 (20295.35)
A=2e-4, B=2e-5, C=2e-5, λ=10	764.47 (319.06)	4870.60 (14195.32)
A=3e-4, B=3e-5, C=3e-5, λ=10	**738.99** (268.14)	6753.91 (16719.99)
A=1e-3, B=1e-4, C=1e-4, λ=10	828.03 (213.48)	**162.08** (48.73)
A=5e-2, B=5e-3, C=5e-3, λ=10	751.8 (281.56)	529.62 (4972.73)

TABLE X
COMPARISON OF ROUTING METHODS IN TERMS OF SIMULATION TIMES FOR NETWOK 3 (FIGURE 5)

HNN PARAMETERS	Simulation time (ms)	
	HNN (Runge-Kutta)	HNN (discrete-time)
A=1e-4, B=1e-5, C=1e-5, λ=1	253860	78687
A=1e-4, B=1e-5, C=1e-5, λ=2	145579	45125
A=1e-4, B=1e-5, C=1e-5, λ=5	68235	20922
A=1e-4, B=1e-5, C=1e-5, λ=10	41063	12484
A=1e-4, B=1e-5, C=1e-5, λ=20	209032	7281
A=1e-4, B=1e-5, C=1e-5, λ=50	1258469	166000
A=2e-4, B=2e-5, C=2e-5, λ=10	41765	78953
A=3e-4, B=3e-5, C=3e-5, λ=10	**40297**	109797
A=1e-3, B=1e-4, C=1e-4, λ=10	48422	**2703**
A=5e-2, B=5e-3, C=5e-3, λ=10	40953	8406

Finally, Table 11 presents the best simulation results in terms of simulation times obtained by the proposed method and HNN solved via Runge-Kutta for each network topology. These results of table 11 are the best results of tables 4, 7, and 10. Notice that as the complexity of the network (given by the number of nodes and the node degree) increases, the simulation time of the HNN solved by Runge-Kutta increases at a significant pace. Conversely, for the method proposed in this paper, the results of table 11 show that the increase in simulation time with computer network complexity is much smaller. Therefore, these simulations show that the proposed method is more efficient than the solution based on Runge-Kutta [2] and more adequate to handle more complex computer networks.

TABLE XI
COMPARISON OF ROUTING METHODS IN TERMS OF THE SMALLER SIMULATION TIMES OBTAINED FOR EACH COMPUTER NETWORK TOPOLOGY

Network	**Number of nodes**	**Node degree**	**Simulation Time (ms)**	
			HNN (Runge-Kutta)	**HNN (discrete time)**
Network 1 (Fig. 3)	6	3.33	8250	**2031**
Network 2 (Fig. 4)	8	3.00	14359	**2579**
Network 3 (Fig. 5)	8	3.25	40297	**2703**

VI. CONCLUSIONS

In this paper we have proposed a routing algorithm based on discrete-time Hopfield neural networks (HNN). The proposed method is based on a previous formulation based on HNNs which uses a differential equation solved through the Runge-Kutta method. In contrast, our method is based on a discrete difference equation.

We have carried out a number of simulations using three computer networks topologies to evaluate the proposed method. The results have shown that the proposed method is able to find optimal paths much faster than the method based on an HNN solved via Runge-Kutta. Furthermore, our simulations have shown that simulation time of the method based on continuous version of the HNN increases with the complexity of the computer networks much faster than in the case of the method proposed in this paper.

REFERENCES

[1] E. W. Dijkstra, 'A Note on Two Problems in Connection with Graphs', *Numerische Mathematik*, vol. 1, pp. 269-271, 1959.

[2] M. K. 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. F. Martins-Filho, C. J. A. Bastos-Filho, E. A. J. Arantes, S. C. Oliveira, L. D. Coelho, J. P. G. Oliveira, R. G. Dante, E. Fontana and F. D. Nunes, "Novel Routing Algorithm for Transparent Optical Networks Based on Noise Figure and Amplifier Saturation", *In Proc. of SBMO/IEEE MTT-S IMOC 2003*, vol.2, pp. 919-923, 2003.

[4] N. Kojic, I. Reljin, and B. Reljin, 'Neural Network for finding Optimal Path in Packet-Switched Network', *In: Proc. of IEEE 7th Seminar on Neural Network Applications in Electrical Engineering, NEUREL 2004,* Serbia and Montenegro, pp. 91-96. September 2004.

[5] J. J. Hopfield, and D. W Tank, '"Neural' computations of decision in optimization problems", *Biol. Cybern*., vol. 52, pp. 141- 152, 1985.

[6] H. E. Rauch, and T. Winarske, 'Neural Networks for Routing Communication Traffic', *IEEE Cont. Syst. Mag.*, pp. 26-30, April 1988.

[7] C. W. Ahn, R. S. Ramakrishna, C. G. Kang, and I. C. Choi, 'Shortest Path Routing Algorithm using Hopfield Neural Networks', *IEE Electronics Letters*, vol. 37, no. 19, pp. 1176-1178, 2001.

[8] D. C. Park, and S. E. Choi, 'A neural network based multi-destination routing algorithm for communication network'. *In Proc. Joint. Conf. Neural Networks*, Anchorage, pp. 1673-1678, USA, 1998.

[9] L. Zhang and S. C. A. Thomopoulos, 'Neural network implementation of the shortest path algorithm for traffic routing in communication networks', *In Proc. Int. Joint Con& Neural Networks*, pp. II. 591, June 1989.

[10] N. Shaikh-Husin, M; K. Hani, and T. G. Seng, 'Implementation of Recurrent Neural Network Algorithm for Shortest Path Calculation in Network Routing', *In: Proc. of IEEE International Symposium on Parallel Architectures, Algorithm and Networks, ISPAN 2002,* pp. 91-96, 2004.

[11] K. M. Sim and W. H. Sun, 'Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions', *IEEE Trans. On Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 33, no. 5, pp. 560-572, September 2003.

[12] G. D. Caro and M. Dorigo, 'AntNet: Distributed stigmergetic control for communications networks', *J. Artif. Intell. Res.*, vol. 9, pp. 317–365, 1998.

[13] S.H. Ngo et al., 'Adaptive routing and wavelength assignment using ant-based algorithm', *In Proc. of 12th IEEE ICON*, vol. 2, pp 482-486, Singapore, November, 2004.

[14] D. Bisbal et al., 'Dynamic Routing and Wavelength Assignment in Optical Networks by Means of Genetic Algorithms', *Photonic Network Communications*, vol. 7, no. 1, pp. 43-58, 2004.

[15] Y.-W. Yuan, H.-H. Zhan, and L.-M. Yan, ' An Adaptative QoS Route Selection Algorithm Based on Genetic Approach in Combination with Neural Netwok*, In. Proc. of the Second Int. Conf. on Machine Learning and Cybernetics*, Xi´an, 2003.

[16] V. T. Le, X. Jiang, S. H. Ngo, S. Horiguchi, "Dynamic RWA Based on the Combination of Mobile Agents Technique and Genetic Algorithms in WDM Networks with Sparse Wavelength Conversion", *in Proc. of 19th IEEE IPDPS*, 2005.

[17] G. G. Yen, and A. N. Michel, 'Stability analysis and synthesis algorithm of a class of discrete-time neural networks', *Mathematical and Computer Modelling*, vol. 21, no. 1-2, pp. 1-29, January, 1995.

[18] S. Guo, L. Huang, and L. Wang, 'Exponential stability of discrete-time Hopfield neural networks', *Computers & Mathematics with Applications*, vol. 47, no. 8-9 , pp. 1249-1256, April-May 2004.