Modeling Shortest Path Routing Problem with Hopfield Neural Network

Danial Esmaeili Aliabadi[1]*

Abstract
In this article, modeling of shortest path routing problem with artificial neural network is considered. In order to make our article clear, we choose different approach as common reports. First of all, we start with basic concepts on modeling and then articles that use the basic concept are introduced with their differences. Therefore, following researchers thought would be easier.

Keywords
Shortest Path Problem — Dijkstra Algorithm — Hopfield Neural Network

[1] *Sabanci University, Faculty of Engineering and Natural Science, Istanbul, Turkey*
***Corresponding author**: Danialesm@sabanciuniv.edu

Contents

Introduction

Routing problem has discussed hardly by the researchers because of its importance in theory and application. Some algorithms determine the routes based on the shortest path, minor delay, higher signal-to-noise ratio (in All-Optical Networks case), load balance, maximum reliability, among others. Shortest Path problem (SP) is one of routing problems that also used as building block in many more sophisticated problems like Maximum Flow and Minimum Cost.

Because of Artificial Neural Network (ANN) characteristics (like computational speed and ability to implemented on hardware), it could be a very good candidate to solve shortest path problems. One obvious application of such network might be in computer networks' routers to find a root which is not congested immediately and doing this directly affect their efficiency.

One type of ANN that used on solving SP is Hopfield neural network (HNN). Firstly, HNN is introduced by Hopfield and Tank (1985) for solving Traveling Salesman Problem (TSP). The use of HNN to find the shortest path between two nodes in a communication network was initiated by Rauch and Winarske (1988). However, this propose requires a prior knowledge of the network topology like number of nodes in the shortest path. To outperform this limitation, Ali and Kamoun (1993) proposed a novel adaptive algorithm, where the weight matrix just carries convergence information. Eventually, Bastos-Filho et al. (2007) proposed a new method to solve energy function faster.

1. Basic Concepts

The Hopfield neural computational circuit is shown in Fig.1. Each neuron is modeled as a nonlinear device with monotonic increasing sigmoid function with output of V_i that is correspond to its input U_i. The output could be any value between 0 and 1 based on input value. The sigmoid function is as follow.

$$V_i = g_i(U_i) = \frac{1}{1+e^{-\lambda_i U_i}} \tag{1}$$

Each neuron calculates its input based on external current (known as bias) I_i and multiplication of all outputs of all other neurons by their connection weight from that neuron T_{ij}. Matrix $T = [T_{ij}]$ also known as connection matrix.

$$U_i = I_i + \sum_{j=1}^{N} T_{ij} V_j \tag{2}$$

The dynamics of the Hopfield network are described by

$$\frac{dU_i}{dt} = \sum_{j=1}^{N} T_{ij} V_j - \frac{U_i}{\tau} + I_i \tag{3}$$

where τ is the circuit's time constant. Hopfield (1984) shown that for symmetric T with sufficiently large λ_i, the dynamics of the neurons follow a gradient descent of the quadratic energy function. Also, he shown that when $V_i \in \{0,1\}$ then for $\lambda_i = \infty$ the minima occurs at one of 2^N corners of hypercube. The dynamics of the ith neuron are described by

$$\frac{dU_i}{dt} = -\frac{U_i}{\tau} - \frac{\partial E}{\partial V_i} \tag{4}$$

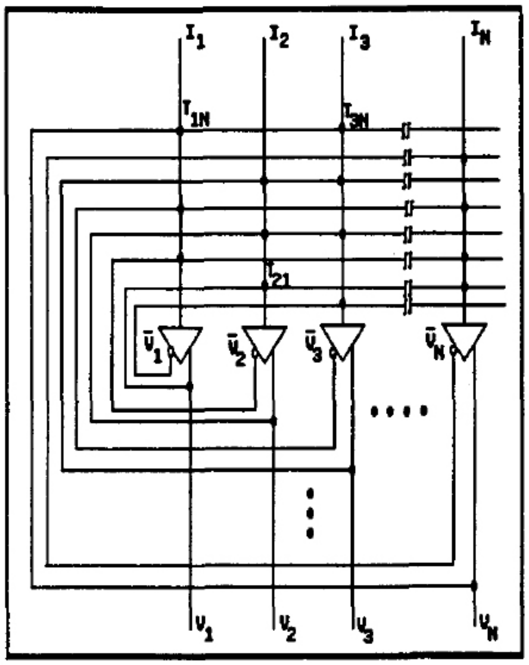


Figure 1. Hopfield Neural Network

1.1 Modeling HNN for SP

In this subsection, we discuss how to model general HNN for solving SP Problem. We define a network as a directed graph $G = (\overline{N}, \overline{A})$ and arc (i, j) has nonnegative cost of C_{ij}. A directed path of P^{sd} could be presented by sequence of nodes connecting s to d with length of $L^{sd} = \sum_{(i,j)\in P^{sd}} C_{ij}$. Thus, the goal would be finding a path from s to d with minimum length.

The best problem formulation for HNN uses this notation for output of each neuron which let it to be more flexible and did not assume anything beforehand about length of shortest path.

$$V_{xi} = \begin{cases} 1 & \text{if the arc from node } x \text{ to node } i \text{ is in the SP} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Therefore, proposed model is organized as (NxN) matrix without any diagonal element. Also due to checking feasibility of a path, we need to define binary parameter of ρ_{xi} with value of 1 when (x, i) does not exist.

In order to solve the SP problem, we need to define energy function that minimizing it, creates set of V_{xi} correspond to shortest path in the real network. It should take valid path into consideration and among all valid paths, it should choose the path with minimum length. The defined energy function is expressed as follow.

$$\begin{aligned} E = {} & \frac{\mu_1}{2}\sum_{x=1}^{N}\sum_{i=1\neq x}^{N} C_{xi}V_{xi} + \frac{\mu_2}{2}\sum_{x=1}^{N}\sum_{i=1\neq x}^{N} \rho_{xi}V_{xi} \\ & + \frac{\mu_3}{2}\sum_{x=1}^{N}\{\sum_{i=1\neq x}^{n} V_{xi} - \sum_{i=1\neq x}^{n} V_{ix}\}^2 \\ & + \frac{\mu_4}{2}\sum_{x=1}^{N}\sum_{i=1\neq x}^{N} V_{xi}(1 - V_{xi}) + \frac{\mu_5}{2}(1 - V_{ds}) \end{aligned} \tag{6}$$

In Eq.(6), the μ_1 minimizes total length of a path by taking into account the cost of existing links. The μ_2 term acts like penalty to prevent nonexistent links from being included in the chosen path. The μ_3 term is zero if for every node in the solution, the number of incoming arcs equal to number of outgoing arcs. Therefore it gives a cycle instead of path that has (d, s) in the shortest path from s to d. The μ_4 term pushes the state of the neural network to converge to one of the 2^{n^2-n} corners of hypercube, defined by $V_{xi} \in 0, 1$. Finally, the μ_5 term is zero when the output of the neuron at location (d, s) settles to 1.

1.2 The Connection Matrix and the Biases

In order to find proper bias vector and connection matrix, we should rewrite all formula based on new notation of V_{xi}.

$$\frac{dU_{xi}}{dt} = -\frac{U_{xi}}{\tau} + \sum_{y=1}^{N}\sum_{j=1\neq y}^{N} T_{xi,yj}V_{yj} + I_{xi} \tag{7}$$

$$\frac{dU_{xi}}{dt} = -\frac{U_{xi}}{\tau} - \frac{\partial E}{\partial V_{xi}} \tag{8}$$

$$V_{xi} = g_{xi}(U_{xi}) = \frac{1}{1 + e^{-\lambda_{xi}U_{xi}}} \tag{9}$$
$$\forall (x, i) \in \overline{N} \times \overline{N} / x \neq i.$$

By taking derivative of $\frac{\partial E}{\partial V_{xi}}$ from Eq.(6) and substituting in Eq.(8), we will obtain $\frac{dU_{xi}}{dt}$. Finally, by comparing with Eq.(7), we can find connection matrix and bias vector which are written as follow.

$$T_{xi,yj} = \mu_4\delta_{xy}\delta_{ij} - \mu_3\delta_{xy} - \mu_3\delta_{ij} + \mu_3\delta_{jx} + \mu_3\delta_{iy} \tag{10}$$

$$I_{xi} = \begin{cases} \frac{\mu_5}{2} - \frac{\mu_4}{2} & \text{if } (x, i) = (d, s) \\ -\frac{\mu_1}{2}C_{xi} - \frac{\mu_2}{2}\rho_{xi} - \frac{\mu_4}{2} & \text{otherwise} \end{cases} \tag{11}$$

where δ is the Kronecker delta defined by

$$\delta_{ab} = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

The advantage of current modeling is that it stores all data in biases rather than into neural interconnections. because we have tried to make a linear modeling for energy function.

Table 1. Parameters used in simulations and their default values

Parameter	Value
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

This representation brings flexibility because the network's topology and costs are stored in the biases and modifying internal coefficients HNN is inessential. The another advantage of the proposed representation is that it is not dependent to the specific source or sink. Therefore, by choosing proper bias values, it can find shortest path for any pair of nodes.

This is correspond to solving a system of $N(N-1)$ nonlinear differential equations where the variables are neuron's output voltages V_{xi}'s. To achieve this, Ali and Kamoun (1993) have chosen the fourth order Runge-Kutta method. Bastos-Filho et al. (2007) used discrete time energy equation. Their proposed equation for evolution of each neuron is as follow.

$$U_{xi}^{[n+1]} = U_{xi}^{[n]} - AU_{xi}^{[n-1]} - BU_{xi}^{[n-2]} + C\Big(\sum_{y=1}^{N}\sum_{j=1\neq y}^{N} T_{xi,yj}V_{yj}^{[n]} + I_{xi}^{[n]}\Big) \qquad (13)$$

where A,B, and C are constants that regulate the weight of the previous inputs.

2. Results and Discussions

In order to calculate the Eq.(13), we need to determine parameters. Bastos-Filho et al. (2007) used the values which are demonstrated in the Table 1.

Most of researchers claimed that HNN converges to the shortest path effectively. A reported sample by Ali and Kamoun (1993) is depicted in Fig.2.

Finally, Bastos-Filho et al. (2007) have reported that their algorithm converges in less number of iterations.

3. Implementation

Based on reviewed articles, the HNN is coded in C# and the result proved the capabilities of this network, especially in asymmetric graphs. My implemented code can model indirected graphs with Euclidean distance cost for available arcs. One major problem with this approach lied in solving energy function properly. Because of structure of energy function, we just used first order condition ($\frac{\partial E}{\partial V_{xi}}$) which may converges to just a stationary point that is not global minimum.

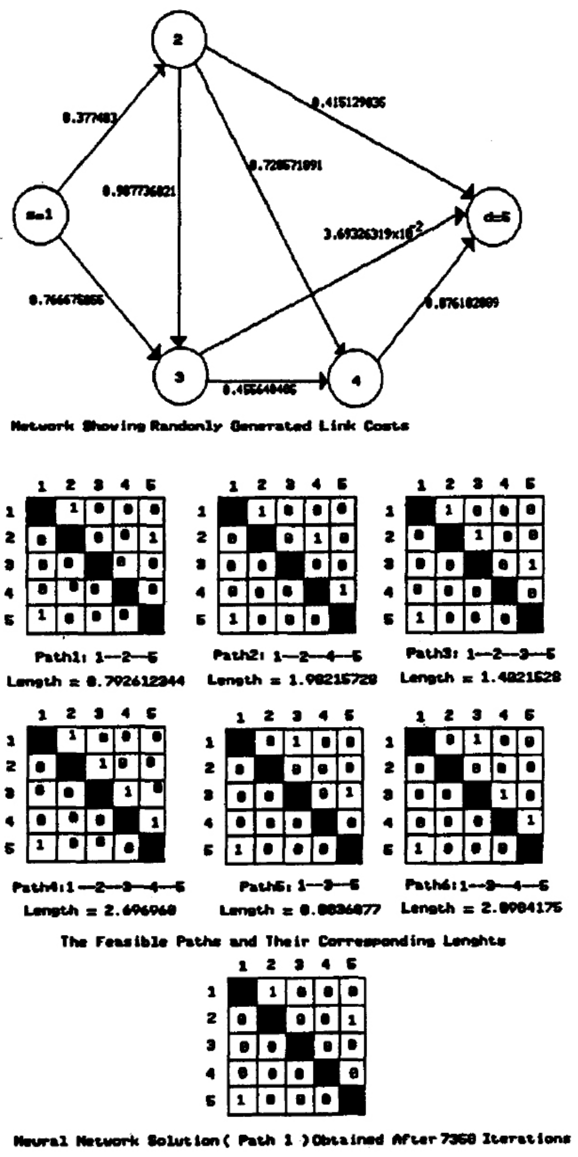
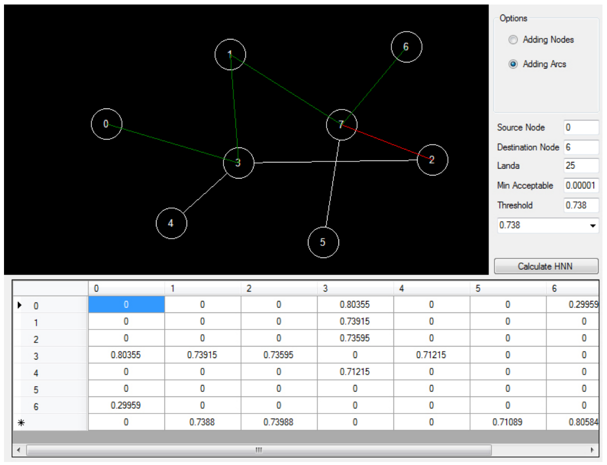


Figure 2. Typical results for the SP problem

Another minor issue about this network is in spare branches that affixed to the main path. For the sake of omitting these unnecessary branches we used breath-first search algorithm from source node to sink node over obtained subgraph from HNN. The result is depicted in Fig.3. The red arc is spare arc which is given by HNN and green path is the refined path of HNN.

4. Conclusion

In this review, modeling of shortest path problem with artificial neural network was considered. The researchers asserted that HNN can find shortest path effectively and because of characteristics of ANN sometimes it would be better to use such a network instead of classic algorithms such as Dijkstra.



Adding Nodes [✓ Adding Arcs]

Source Node	0
Destination Node	6
Landa	25
Min Acceptable	0.00001
Threshold	0.738
	0.738

	0	1	2	3	4	5	6
0	0	0	0	0.80355	0	0	0.29959
1	0	0	0	0.73915	0	0	0
2	0	0	0	0.73595	0	0	0
3	0.80355	0.73915	0.73595	0	0.71215	0	0
4	0	0	0	0.71215	0	0	0
5	0	0	0	0	0	0	0
6	0.29959	0	0	0	0	0	0
*	0	0.7388	0.73988	0	0	0.71089	0.80584

Figure 3. Implemented Application for solving SP with HNN

References

Hopfield, J.J., Tank, D.W.. Neural computation of decisions in optimization problems. Biological cybernetics 1985;52(3):141–152.

Rauch, H.E., Winarske, T.. Neural networks for routing communication traffic. Control Systems Magazine, IEEE 1988;8(2):26–31.

Ali, M.M., Kamoun, F.. Neural networks for shortest path computation and routing in computer networks. Neural Networks, IEEE Transactions on 1993;4(6):941–954.

Bastos-Filho, C., Santana, R., Oliveira, A.. A novel approach for a routing algorithm based on a discrete time hopfield neural network. In: Foundations of Computational Intelligence, 2007. FOCI 2007. IEEE Symposium on. IEEE; 2007, p. 363–369.

Hopfield, J.J.. Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings of the national academy of sciences 1984;81(10):3088–3092.