Dynamic Channel Assignment Using Neural Networks

S. M. Hadi , Z. K. Ahmed
Department of Information Eng., College of Al-khawarzmi Eng. University of Baghdad
Department of Computer Science, College of Ibn Al-Hathem University of Baghdad

Abstract

This paper presents a proposed neural network algorithm to solve the shortest path problem (SPP) for communication routing. The solution extends the traditional recurrent Hopfield architecture introducing the optimal routing for any request by choosing single and multi link path node-to-node traffic to minimize the loss. This suggested neural network algorithm implemented by using 20-nodes network example. The result shows that a clear convergence can be achieved by 95% valid convergence (about 361 optimal routes from 380-pairs). Additionally computation performance is also mentioned at the expense of slightly worse results.

Keywords: Optimal Routing, Hopfield Neural Network Algorithm, Single and multi-link path

Introduction

Communication routing resource allocation problems are becoming increasingly relevant given the upsurge in demand of the internet and other telecommunication services. One such problem amounts to assigning arcs (links) in a connected network to requests from start-to-end nodes, given capacity constraints on the links such that a total additive cost (path-length) is minimized (1). A relatively simple routing problem, with only one request at a time, is the Shortest Path Problem (SPP), which can be solved exactly in polynomial time using Hopfield neural network algorithm were made

to provide an approximate solution to the (SPP) faster than with any other algorithm solution (2,3).

The idea was first presented by a Hopfield algorithm for the SPP network; their method exhibits an example of 16 nods that get a 100% convergence results (4). However this method has a drawback, which can be addressed by the neural network converge towards a valid solution that was connected by an exact number of nodes (16 nodes). This paper and by using a modified algorithm try to improve the result with an increasing number of nodes in the network graph.

We proposed nearly the same Hopfield algorithm in certain classes of graphs, when the number of nodes approaches 20 or more. An example of 20 nodes has been taken to be implemented which made it possible to find a path with as many as N hops, N being the number of nodes in the graph, which is also the highest number of hops the SP may have.

In our paper we present a modification of Hopfield neural network algorithm that aims to improve the reliability of the solutions, where reliability stands for succeeded and valid convergence. To achieve this, a new architecture, named Dependent Variables (DV) have been used. This architecture automatically guarantees an entire class of restrictions, thus considerably increasing the reliability of the method. At the same time, the number of neurons is equal to the number of arcs in the graph instead of being equal to the squared number of nodes as it is the case in any other NN methods.

Defining the Problem

Consider a graph G = (V, A) composed of N vertices V and a set of M arcs A. Associated with each arc(r, s) is a nonnegative number C_{rs} that stands for the cost from node r to node s. Non-existing arc costs are set to infinite (∞). Often, for clarity of exposition, namely when referring to figures, we will label each existing arc with a unique index and denote its cost simply by C_{ij}. Let P_{sd} be a path from a source node s to a destination node d, defined as a set of consecutive nodes (5), connected by arcs in set A:

$$P_{sd} = \{s, n_1, n_2 \ldots, d\}$$

There is a cost associated with each path P_{sd} that consists of all partial arc costs participating in the path. The shortest path problem consists

in finding the path connecting a given source – destination pair, (s,d), such that the cost associated with that path is minimum. Stated as an integer linear programming problem (6,7) this is:

To Minimize

$$\sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij} V_{ij} \qquad \ldots\ldots\ldots [1]$$

Subject to

$$\sum_{\substack{j=1 \\ j\neq i}}^{N} V_{ij} - \sum_{\substack{j=1 \\ j\neq i}}^{N} V_{ji} = 0 \qquad i = 1,\ldots\ldots\ldots,N \qquad \ldots\ldots\ldots [2]$$

And

$$v_{ij} \in \{0,1\}$$

Where V_{ij} is the participation of the arc (i, j) in the path which can only be 0 or 1, i.e., the arc whether participates entirely or doesn't participate at all in the path. Non-existing arc is not to be considered (8).

As an example of routing in a computer networks, consider the 20-nodes network graph showed in fig.(1), was used to evaluate the neural network's ability to handle larger networks (with larger nodes) and in this example it is possible to find a path with as many as 20 hops (number of links). The links, nodes, and cost capacity can be represented by a 20x20 matrix C with entries C_{ij} as shown in table (1), where the entry C_{ij} represents the capacity between source node i and destination j. When there is a 0 entry, it means that there is no direct link between the two nodes and communication requires multiple-link through adjoining nodes. In this example, the matrix C is symmetric but in general, this is not necessary.

The node-to-node routing problem requires finding the routing from the source i to the destination j, which minimizes a cost function such as the total number of links. The minimization procedure is implemented using a modification of the neural network traveling salesman.

The Network Routing

Our new example that have been built on a modified Hopfield algorithm can be explained by considering the 20 nodes network shown in fig. (1), which is require 20 control vectors assigned by 20 elements for each vector (U1, U2, U3,.....U20) in order to represent the location of the nodes for the path in the network. This algorithm start by giving the 20 control vectors (U) a reasonable initial estimate and then attempts to converge them to the final value of the control vectors, which is corresponds to the desired number of the nodes.

From the elements of the cost matrix and by using the minimum number of links for every source I to destination j we can determine the best minimum route from the minimum of the cost matrix. The cost-effective path through the network can be described by 20 vectors/with 20 elements for each vector V_1, V_2, V_3...V_{20}, where the rows represent the nodes of the network and the columns the position of the node in the path.

Given the source and the destination pair, the source vector and the destination vector have zero entries except for the source and destination nodes. The other vectors has zero entries for nodes with has no direct link to the source and destination node, and the values 1 for the nodes that have a direct link with the origin node and the destination node, and entries 0.05 at each of the other 18 nodes.

Assuming that it is desired to route from node 4 to node 19 in seven links or less. From visual inspection, it is apparent that candidate route with seven links goes through the eight nodes 4-3-7-6-11-16-20-19. The first and last nodes are the source and destination and the six interior nodes represent points in the seven links connection. Since the source U_1 and the destination U_8 are know the purpose of the routing algorithm that is to find a way to converge to the six intermediate vectors, U_2, U_3 ,U_4, U_5, U_6 , and U_7.

There is a quick way to determine the minimum number of links from every source i to the destination j from the elements of the capacity matrix C. Nonzero elements of the capacity matrix C represent source-to-destination pairs, where the minimum route requires only one link. Nonzero elements of the self-product of the capacity matrix (C_xC) represent pairs where the minimum route requires two links. In general, nonzero elements of the L^{th} product of the capacity matrix (C^L) represent pairs, where the minimum route requires L or fewer

links. Higher products can be grouped to reduce computation effort. Thus, C^4 is the product $C^2{}_xC^2$. This procedure has been programmed, and it is an effective way to reduce computation time by knowing in previous the minimum number of links for each source-destination pair.

The cost-effective path through the network can be described by eight 20-dimensional vectors V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8, where the rows represent the nodes of the network and the columns the position of the node in the optimal path from node 4 to node 19. Within each iteration of the new suggested ANN algorithm, a system of N differential equations in Equation (4) has to be evaluated; this step is then repeated until convergence. It should be noted here that the effort for solving the optimal routing problem rises with N_xM number of links.

The Energy Function

The modified neural network algorithm uses the same computation gradient approach to minimize the energy function that used in the 16 node example. This function has to be minimized and the lowest energy state is equivalent to the optimal route. The energy function can be stated as follows:

$$E_f = \frac{A}{2}\sum_{x=1}^{N}\sum_{\substack{y=1\\y\neq x}}^{N}\sum_{i=2}^{N-1} W_{xy}V_{xi}(V_{yi+1}+V_{yi-1}) + \frac{B}{2}\sum_{i=1}^{N}\sum_{\substack{i=1\\i\neq x}}^{N} P_{xi}V_{xi} + \frac{C}{2}\sum_{i=2}^{N-1}\sum_{x=1}^{N}\sum_{\substack{y=1\\y\neq x}}^{N} V_{xi}V_{yi}$$

$$+\frac{D}{2}\sum_{x=1}^{N}\sum_{i=2}^{N-1}\sum_{\substack{j=1\\j\neq i}}^{N} V_{xi}V_{xj} + \frac{E}{2}\sum_{x=1}^{N}\left(\sum_{i=1}^{N} V_{xi} - 1\right)^2 \quad \text{.....[3]}$$

There are five separate terms in the above equation, these are:

- The A-term minimizes the total cost of a path, taking into account the cost of existing links.
- The B-term excludes non-existing links. The term P_{ij} has the value 1 if the link between node i and j exists, otherwise it is 0.
- The C-term sum is zero if and only if each column contains no more than 1.
-The D-term sum is zero if and only if each row contains no more than 1.

- The E-term sum is zero if and only if there are N '1' in the entire matrix (ensures that exactly N entries equal to 1).
The dynamic (differential equation) of the neural network is converted by the following equations:

$$\frac{d}{dt}U_{xi} = -U_{xi} - A\sum_{\substack{y=1 \\ y\neq x}}^{N} W_{xy}(V_{yi+1} + V_{yi-1}) - BP_{xi} - C\sum_{\substack{y=1 \\ y\neq x}}^{N} V_{yi} - D\sum_{\substack{j=1 \\ j\neq i}}^{N} V_{xj} - E\left(\sum_{i=1}^{N} V_{xi} - 1\right) \quad [4]$$

Where u_{ij} denotes the total input to ij-th neuron then we have

$$U_{ij}^{t+\Delta t} = \frac{d}{dt}U_{ij} \cdot \Delta t + U_{ij}^{t} \qquad .....[5]$$

$$V_{ij} = 0.5\left(1 + \tanch\left(U_{ij}\right)\right) \qquad .....[6]$$

The New Suggested Algorithm

The algorithm of finding the optimal route contains many steps as follows:
Step1: Determine the minimum numbers of links from every source i to destination j.
Step2: Determine the initial condition for the control vectors.
Step3: Initialize all other coefficients.
Step4: While the stopping condition is false, do step 5-9.
Step5: Perform steps 6-8 N_xMin. No. of links times.
Step6: Choose a unit at random.
Step7: Change activity on the selected unit: this step depends on the selected energy function in which $u_{ij}^{new} = u_{ij}^{old} + \Delta t_x \, du_{ij}/dt$
Step8: Apply output function: $V_{ij} = 0.5(1+\tanch(U_{ij}))$
Step9: Check stopping condition.

The Experimental Results

In this section an attempt is made to determine the 20- node example results for seven-links simulation, the seven vectors in table (2) show the initial conditions for control vectors. U_1 and U_8 have zero entries except for the source node 4 and the destination node 19. U_2

has the values of 1 for the node 3 that has a direct link with the node 4 (source). U_7 has the values of 1 for the node 20 that have a direct link with the node 19 (destination). U_3, U_4, U_5, and U_6 have zero entries at the source and destination nodes (4 and 19) and entries 0.05 at each of the other 18 nodes.

The seven vectors in table (3) shows the optimal path from node 4 to node 19, where the rows represent the nodes of the network while the columns represent the position of the node in the path. Therefore, the optimal route for this example is: 4-3-7-6-11-16-20-19.

Neural Network Simulation

With 20-nodes example and two way traffic between each pair of the nodes there are 20_x19, i.e., 380 source-destination pairs, so the new suggested NN algorithm must be repeated 380 times for each step of serial simulation. The new suggested NN algorithm results with 20-nodes example are shown in table (4). While Figure (2) show the constraint energy evaluation for the 20 nodes simulation examples with reasonable convergence in 300 iterations to determine optimal route from node 1 to node 20.

All of the solutions resulting from simulations are 95% of the solutions were optimal and remaining 5% were worse by only one or two units (from 380 source-destination pairs given 361 optimal route and only 19 pairs are wrong solutions). In this simulation we given 95% valid convergence under the condition that the optimization parameters B, C, D, E used as a Lagrange parameters in Equation (3).

In general, the convergence is to one preferred route. For example, for the source node 1 to destination node 20 pair, the steady-state value has one preferred route, which traverses node 1-6-11-16-20 and the algorithm shows reasonable convergence in 300 iterations to determine the optimal route from node 1 to node 20. In the simulation of the 20-nodes network, the node 6 carries much of the traffic between the left and right half of the network. In table (4) it can be seen that three of the source–destination pairs (rows 4, 5, 7) show wrong solutions. By looking at the wrong solutions we can observe an important behavior of the new suggested NN algorithm.

Conclusions

This research presents a new neural network method that modified from Hopfield algorithm for determining the optimal routing in the computer networks. An evolution of this method, thought for the multi-destination routing problem, and in a single-destination version it extends the range of operation of the former method, achieving noticeable improved solutions even with a bigger number of nodes. In certain classes of the graphs all these solutions demand a number of neurons that squares the number of graph nodes.

In this optimization process, the optimal routing problem is formulated in terms of gradient minimization, where the routing problem requires choosing multilink paths for node-to-node to minimize loss, which is represented by the total number of links or other functions. The minimization procedure is implemented using a modification of the neural network traveling salesman algorithm. In comparisons, the new suggested NN algorithm is faster than the traditional approaches providing an approximate solution to optimal routing problem, where the time required by our new algorithm is about 52% of the total time required by the others.

References

1. Fritch, T. ; Mittler, M. and Tran-Gia, P. (1993) Neural Computing and Applications, Springer-Verlag, London Limited, 1, 2 : 124-146, 29 March.
2. Ford, L. R. and Fulkerson, D. R. (1992) Flows in networks , Addison-Wesley,
3. M. Schwartz and T. Stern, (1980) Com-28: 4.
4. Salman, A. A. (2005) Optimal Assignment of Communication Channel Using Neural Networks, published in the Engineering Journal, college of engineering / university of Baghdad, 11:1
5. L. Fausett, (1994) Fundamentals of neural networks architecture, algorithms, and applications, Prentice Hall International, Inc.
6. J. M. Zurada (1996) Introduction to artificial neural network systems , Second Jaico Impression, Jaico Publication House.

7. Rauch, H. E. and Winarske, T. (1988) Neural networks for routing communication traffic, IEEE Control Systems Magazine, 24, :26-31, April .
8. Zhang, L. and Thomopoulos, S. C. A. (1991) Neural network implementation of the shortest path algorithm for traffic routing in communication networks, In: Proceedings IEEE-IJCNN, Singapore.

Table (1) Cost Capacity Matrix for 20-Node Network in Fig (1)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	1	0	0	4	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	5	0	0	0	0	0	2	0	0	3	6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	2	0	8	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
10	0	0	0	0	0	3	0	0	0	0	0	0	0	2	0	0	0	0	0	0
11	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	1	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0
13	0	0	0	0	0	0	0	0	1	0	0	0	0	2	0	0	1	0	0	0
14	0	0	0	0	0	0	0	0	0	2	0	0	2	0	1	0	0	5	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	2
17	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	3	0

Table (2) Initial Control Vector for 20 nodes (seven links)

Element	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8
1	0	0	0.05	0.05	0.05	0.05	0	0
2	0	0	0.05	0.05	0.05	0.05	0	0
3	0	1	0.05	0.05	0.05	0.05	0	0
4	1	0	0	0	0	0	0	0
5	0	0	0.05	0.05	0.05	0.05	0	0
6	0	0	0.05	0.05	0.05	0.05	0	0
7	0	0	0.05	0.05	0.05	0.05	0	0
8	0	0	0.05	0.05	0.05	0.05	0	0
9	0	0	0.05	0.05	0.05	0.05	0	0
10	0	0	0.05	0.05	0.05	0.05	0	0
11	0	0	0.05	0.05	0.05	0.05	0	0
12	0	0	0.05	0.05	0.05	0.05	0	0
13	0	0	0.05	0.05	0.05	0.05	0	0
14	0	0	0.05	0.05	0.05	0.05	0	0
15	0	0	0.05	0.05	0.05	0.05	0	0
16	0	0	0.05	0.05	0.05	0.05	0	0
17	0	0	0.05	0.05	0.05	0.05	0	0
18	0	0	0.05	0.05	0.05	0.05	0	0
19	0	0	0	0	0	0	0	1
20	0	0	0.05	0.05	0.05	0.05	1	0

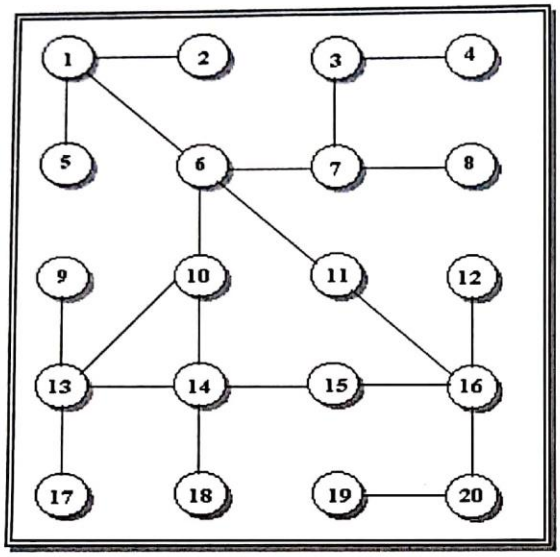

Fig. (1) Network Example with 20 Nodes.

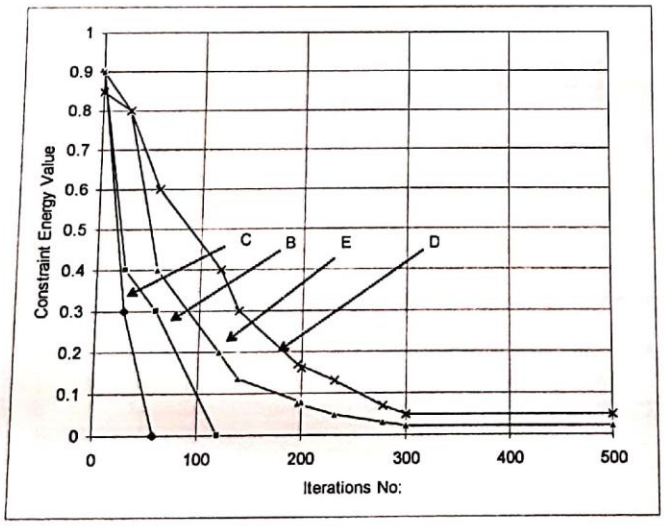

Fig. (2) Constraint energy evolution for 20 node example.

Table (3)Path Represented 20 nodes (seven links).

Element	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	1	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	1	0	0	0	0
7	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	1	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
16	0	0	0	0	0	1	0	0
17	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	1
20	0	0	0	0	0	0	1	0

Table (4) Results for a Network with 20 nodes (seven links)

	Source	Destination	Result	*Valuation*
1	1	19	1- 6 – 11 – 16 – 20 - 19	Optimal
2	1	20	1 – 6 – 11 – 16 - 20	Optimal
3	6	17	6 – 10 – 14 – 13 - 17	Optimal
4	4	13	4 – 3 – 10 – 14 - 13	Wrong
5	8	19	8 – 7 – 11 – 16– 20- 19	Wrong
6	10	17	10 – 14 – 13 - 17	Optimal
7	4	15	4 – 3 – 10 – 14 – 15	Wrong
8	2	6	2 – 1 - 6	Optimal
9	5	7	5 – 1 – 6 - 7	Optimal
10	12	19	12 – 16 – 20 - 19	Optimal

مجلة ابن الهيثم للعلوم الصرفة والتطبيقية المجلد 18 (2) 2005

تحديد افضل مسار ديناميكي بأستخدام الشبكات العصبية

سها محمد هادي ، زينب قاسم أحمد

قسم هندسة المعلومات ، كلية هندسة الخوارزمي ، جامعة بغداد

قسم علوم الحاسبات ، كلية التربية – أبن الهيثم ، جامعة بغداد

الخلاصة

يتناول هذا البحث مشكلة إنشاء وتحديد افضل مسار لغرض إرسال واستلام الرسالة من المرسل إلى المستلم بين أي حاسبتين في الشبكة لغرض الاتصال. حيث يعرض هذا البحث تقديم خوارزمية مقترحة تمثل استخدام الشبكات العصبية لتحديد افضل مسار بين أي مضيفين في الشبكة.

أن مشكلة تحديد افضل مسار تتطلب اختيار اكثر من ارتباط multi-link للمرور من عقدة إلى عقدة وبأقل خسارة أي (اقل كلفة) والتي تتمثل في عدد الارتباطات الكلي أو أي دالة أخرى من دوال المرور. لإيجاد افضل مسار تم بناء خوارزمية جديدة عن طريق إجراء تعديل على خوارزمية الشبكة العصبية التي أستخدمت لحل مشكلة البائع الجوال للحصول على اقصر الطرق وأفضلها. تم تطبيق الخوارزمية المقترحة على شبكة تتكون من 20 عقدة. الخوارزمية المقترحة أعطت نتائج جيدة مع الشبكة ذات 20 عقدة حيث إنها اقتربت بنسبة 95% من الحلول الصحيحة (أعطت افضل مسار لـ 361 زوج مرسل-مستلم من 380 زوج) .