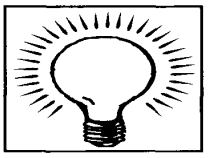


The Use of Artificial Neural Networks for Optimal Message Routing

A human information processing system is composed of many neurons switching at speeds about a million times slower than computer gates. Yet humans are more efficient than computers at computationally complex tasks such as speech understanding, visual recognition, etc. Artificial neural networks are designed to exploit this unique computational power of the human brain.

Chia-Jiu Wang and Paul N. Weissler

Given a network in which message traffic is being routed along communication links between nodes making up the network, the problem arises of how to determine the optimal paths to route the message through traffic. In determining the optimal path, links are considered to have a cost representing the desirability of using that particular link to route a message. The optimal path then becomes the path in which the total cost of routing a message between a source and destination node is minimized. While there are many conventional solutions available, none of them are without flaws. In order to find a better solution, the possibility of using an artificial neural network of some type as a message routing controller is explored in this article. Several neural network architectures are examined for their suitability to the problem of optimal message routing. Two different neural network architectures are implemented and tested against a network simulator in order to ascertain if their performance is adequate to allow them to be used as a routing controller in a network. A comparison between the neural solution and the conventional routing algorithms is also presented.

The routing of packets from a source node to a destination node is an important issue in the design of a communication network consisting of multiple nodes and links, because it affects several performance measures of interest. The objective of a routing algorithm is to optimize some performance measure, such as mean packet delay or network throughput.

There are many routing algorithms in use with different levels of sophistication and efficiency [1]. Routing can be done in a centralized, distributed, or localized manner. In centralized algorithms, all route choices are made at a central node, while in distributed algorithms, the computation of routes is shared among the network nodes with information exchanged between nodes as necessary. In localized routing algorithms, each node needs to have the current network connectivity and computes the routes to all possible destination nodes based on this connectivity information. In order to have the most current network connectivity, all network nodes broadcast their connectivity to neighboring nodes to each other. The Dijkstra algorithm [1] can be considered as a localized algorithm.

The centralized routing method requires a special node in the network which periodically receives information from all other network nodes and, based on this global information, it sets up and updates routing tables for all nodes. This method suffers from high communication overhead and reliability problems, since failure of the central control node results in shutdown of the entire network.

The distributed routing approach can reduce some problems in centralized routing. In this case, each node makes its own routing decisions based on the local information it receives from its neighboring nodes. Looping of packets and deadlocks might happen due to inconsistent routing paths. The most commonly used distributed routing algorithm is the Bellman-Ford algorithm [1].

In the localized routing algorithm, such as the Dijkstra algorithm, all network nodes broadcast their network connectivity to neighbor nodes, so that each network node can react quickly to changes in the network, but does incur the communication cost of broadcasting such changes.

Either centralized, distributed, or localized routing algorithms can be operated in a static or adaptive manner. In static routing algorithms, the path used by each origin-destination pair is fixed regardless of traffic conditions and network changes. In adaptive routing algorithms, the paths used to route new messages between origins and destinations change occasionally in response to traffic conditions and network changes, i.e., failed links, increase or decrease of link cost.

CHIA-JIU WANG is an associate professor of electrical and computer engineering at University of Colorado, Colorado Springs.

PAUL N. WEISSLER has worked for Boeing Electronics Company, Lexico Enterprises, and Kaman Science Corporation as a software engineer.

This research was sponsored in part by the BMDO and managed by the Office of Naval Research under contract N00014-92-J-1761. Part of this paper was presented at IEEE MILCOM '92.

Motivation

The recent resurgence of interest in neural networks has its roots in the recognition that the brain performs computation in a different manner than do conventional digital computers. Computers are extremely fast and precise at executing sequences of instructions designed for specific algorithms. A human information processing system is composed of many neurons (i.e., processing elements) switching at speeds about a million times slower than computer gates. Yet humans are more efficient than computers at computationally complex tasks such as speech understanding, visual recognition, etc. Artificial neural networks are designed to exploit this unique computational power of human brain. From the technology point of view, Very Large Scale Integration (VLSI) circuits can put tens-to-hundreds of processing elements in one chip, such as Intel Electronic Trainable Artificial Neural Networks (ETANN) 80170NX. For some problems, the neural VLSI chip can indeed produce solutions better and faster than the conventional approach [10]. It is the intent of this article to investigate the possibility of the using neural network approach to solve the message routing problem. For a communication network, each node can be equipped with a neural network, and all network nodes can train and later use the neural networks to obtain optimal or near-optimal routes for messages at the same time. In other words, all neural networks work collectively to solve the message routing problem. The neural approach for routing is neither centralized, distributed, nor localized. A brief review of neural networks and their applications in communications is presented in the next section. Following that, the neural network solutions to the routing problems are presented. The following section gives the simulation results and performance comparisons. A comparison between the neural approach and other popular routing algorithms such as Bellman-Ford's and Dijkstra's algorithms is then presented. The practical significance of this new routing algorithm is discussed and further research work is suggested in the conclusion.

Neural Networks

Overview of Neural Networks

Neural networks [10] are information processing systems that consist of nonlinear processing elements and weighted connections. Each layer in a neural network consists of a collection of processing elements. Each processing element collects the value from all of its input connections, performs a predefined mathematical operation, and produces a single output value. There seem to be numerous ways of classifying neural networks for different purposes. From the architectural point of view, neural networks can be classified as single layer feedforward networks, multiple layer feedforward networks, recurrent network (i.e., with feedback), and bidirectional neural networks. Another meaningful basis for classifying neural networks is the learning mode. Supervised and unsupervised learning are the main forms of learning. Supervised learning is often used in single and multiple layer networks. Unsupervised learning is often used in single layer networks. Some neural networks use batch learning. Batch learning means that the complete design information is available *a priori*. Then, neural networks are designed by recording or encoding these design information into desired equilibria. For example, the Hopfield network is a single layer network with feedback and can be used to represent a dynamic system.

Neural networks offer interesting alternative solutions to many problems. Useful applications have been designed, built, and commercialized, and much research continues in hopes of extending current success. Neural network applications emphasize areas where they appear to offer a more appropriate approach than traditional computing has. Neural networks offer possibilities for solving problems that require pattern recognition, pattern mapping, dealing with noisy data, pattern completion, associative look-ups, and systems that learn or adapt during use. Some optimization problems can also be addressed with neural networks. The range of potential applications is impressive.

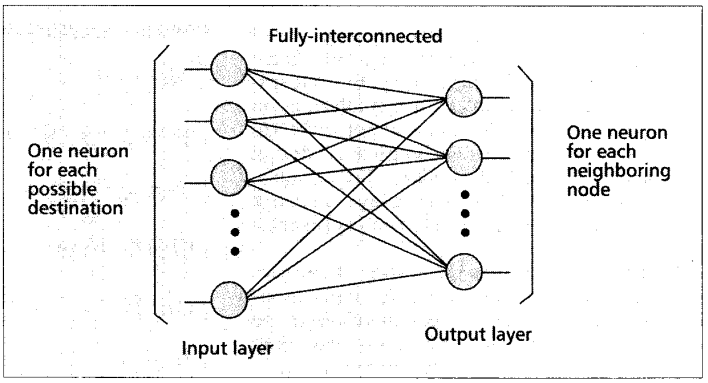
Neural Networks in Communications

There are two neural network architectures that have been proposed for communication networking applications: the feedforward multilayer network and the feedback optimization network. A feedforward network has been used to create a controller for service quality control in the asynchronous transfer mode (ATM) network [2]. Brown [3] and Troudet [4] have shown that a feedback neural network can be used to schedule packet transmissions through a crossbar network for maximum throughput. Morris and Samadi [5] have described their research in using feedforward networks for admission control and using feedback networks for switch control. The feedback network is used as an optimizer in all these applications. Jensen, Eshera, and Barash [6] also showed an application example of using a feedforward neural network as a controller for adaptive routing in communication networks. The major function performed by a routing algorithm is the selection of routes for various origin-destination pairs. Once the route is selected, the delivery of messages is straightforward and accomplished through a variety of protocols and data structures.

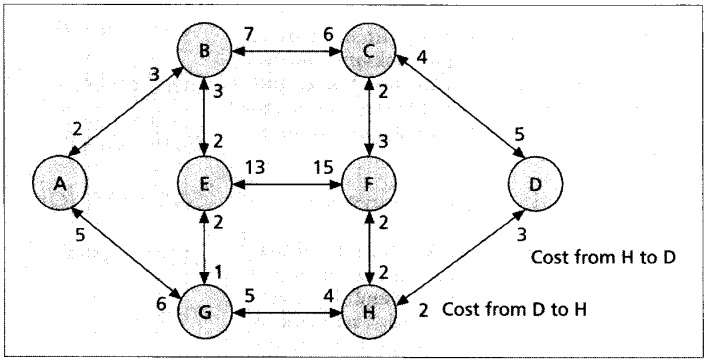
Neural Network Approach to Routing

The network proposed by Jensen, Eshera, and Barash offers a definite solution to the problem of optimal message routing [6]. We call this network the JEB network for simplicity. Figure 1 shows a JEB network, which is simply a standard, two-layer, feedforward network. As can be seen from Fig. 1, the architecture requires one layer which contains neurons for each possible message destination and another layer which contains neurons for each directly connected neighbor of the node. Figure 2 presents an example network. The JEB network for node F in Fig. 2 is present-

Neural networks offer interesting alternative solutions. Useful applications have been designed, built, and commercialized, and much research continues in hopes of extending current success.



■ Figure 1. *Jensen, Eshera, and Barash network.*



■ Figure 2. *An example of a message network.*

ed in Fig. 3. The JEB network architecture uses Hebbian learning for training the net. The exact equation is given below. All activation and update functions are identity functions.

$$w_{ij} = \eta a_i o_j$$

where w_{ij} is the weight from neuron i to neuron j; a_i is the input to neuron i in the input layer; o_j is the output of neuron j in the output layer; η is the learning rate (set to 1).

For the JEB net, training occurs every time a message is received at a node, meaning that the network is being continuously trained while being used. This allows the individual neural network located at each node to keep abreast with changes in the network topology. In order for this training to occur, each message packet must contain the necessary information for the neural net to be updated. In this case, this information includes the origin of the message, the destination of the message, the neighboring node from which the message was received, and the cost of the path the message picked up. In reality, the cost of a path in one direction could be different in the opposite direction, i.e., the cost of a link is asymmetric. Hence the cost picked up by each message is the cost of links in the direction toward the message origin. For any two nodes it needs two messages heading for different directions to pick up the cost of each direction.

The exact training procedure is summarized here:

- Assert a one on the input node (a neuron in the input layer) corresponding to the origin of the message packet. Assert zeros on all other input nodes.
- Assert the cost of the path the message picked up on arriving at the current node on the output node (a neuron in the output layer) corresponding to the neighboring node through which the message passed to reach the current node.
- Update the weight of the line connecting the input node to the output node using the Hebbian learning algorithm.

To actually determine the appropriate message routing path for an outgoing message, the following steps are followed.

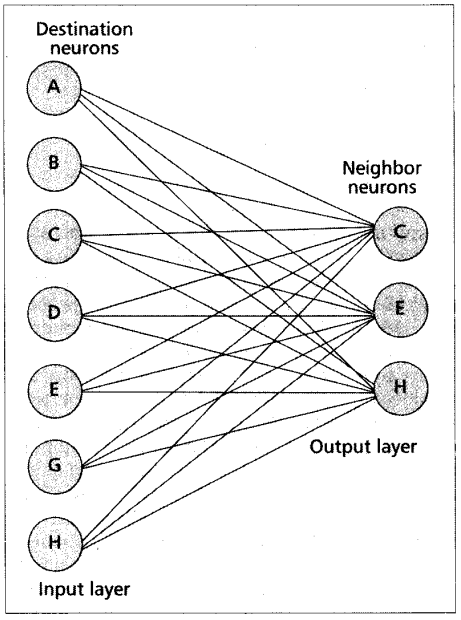
- Assert a one on the input node corresponding to the destination of the message packet. Assert zeros on all other input nodes.
- Pick the lowest value output neuron. This indicates that this neighbor the message should be routed through for the minimal routing cost.

This procedure would be repeated at each network node until the final destination was reached. Note that the neural networks for each node are kept up to date about the status of the network as long as there are messages passing through them. Some experimental results with the JEB net architecture are given in [6], and show that the JEB neural routing performs better than random routing or hot potato routing, but worse than the static routing paradigm by 3 percent. The static routing paradigm is a lookup table method constructed with optimal paths at the start of run.

JEB Network Problems and Proposed Solutions

There are several routing problems that need to be overcome if the JEB network is to function adequately for the task of message routing. These problems and the proposed solutions are presented below and implemented later in our simulator.

Problem 1 — Node sends but never receives messages. Since the JEB nets are only updated when a message is received by a node, in this case with



■ Figure 3. *Example JEB net for node F of Fig. 2.*

no messages ever being received, the node in question would never be updated with the new message network topology. Under certain circumstances, this could easily lead to a definite performance degradation of the network should this particular node be sending out numerous messages.

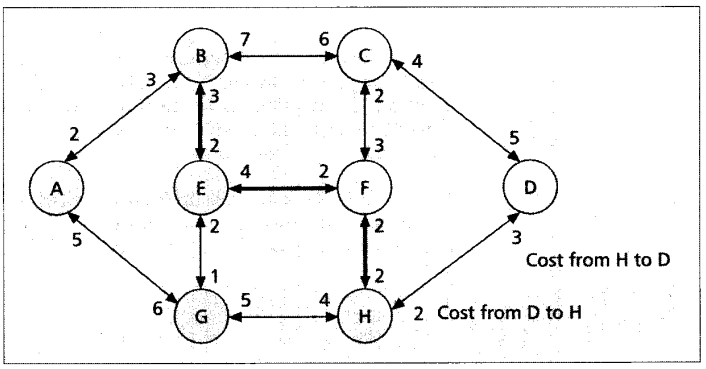
Solution 1 — To solve this problem, a scheme called "message echoing" is proposed and implemented in our simulator. With this attribute, any node receiving a message would send back a special class of message, called "echo message" to the originating node. This echo message would contain the message originating node, the destination node, and the cost it takes the original message to reach the destination node. The originating node, upon receiving the echo message, would update its JEB net using the cost contained in the echo message. Note that the originating node would need to first determine which neighboring node it used to send the original message and update that destination/neighbor neuron combination.

Problem 2 — Cost change occurs on path between nodes. This problem occurs when a cost increase in a link takes place. Should this change dictate a change in the optimal routing path, only nodes which receive messages through a link in which the cost change has taken place will be aware of this change. This occurs due to the fact that nodes unaware of the link cost change will use a different path, thus not providing updated information on the link cost. An example is given below to illustrate this problem. Referring to Fig. 2, a message is transmitted from node B to node H. Assuming all JEB networks are currently set such that the optimal path is always chosen for any message, the message would take the path B->E->F->H with a cost of 6 (as indicated by the heavy lines in Fig. 4). Now the cost on the link E->F is changed to 15 and F->E is changed to 13, as shown in Fig. 5. Once again a message is sent from B to H. This path will still be B->E->F->H, however, with a cost of 19 this time. Because this message picked up the new cost of link F->E, nodes F and H are made aware of the cost change. Nodes F and H update their networks with new cost only for destination B. Later on, a message from node H to node B will use the path H->G->E->B, so that node B does not have a chance to learn about the cost increase of link E->F. Hence, should a message from B to H be sent, the old path B->E->F->H will still be used. Any other cost changes on destination nodes are not updated with the new link cost information. Hence, the JEB net is not fully updated.

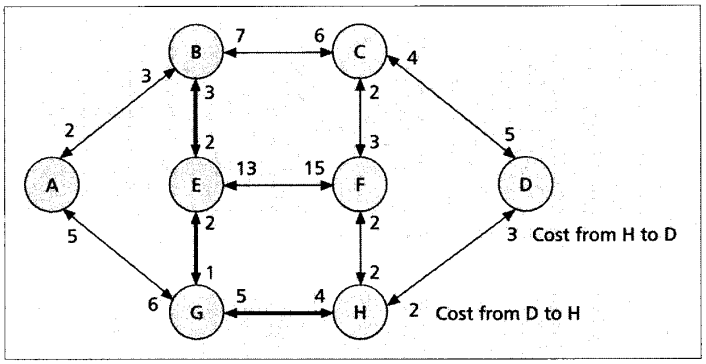
Solution 2 — Using only message echoing is not enough to completely solve problem 2. A method called "local link awareness" is used in cooperation with message echoing to solve problem 2. Local link awareness makes nodes aware of any cost changes to the links to which they are directly connected. To implement this, the node is alerted to the cost change and informed of the new cost. The node then causes its JEB net to update costs of all destinations which pass through the link whose cost has changed.

Making nodes aware of changes in their local links still does not resolve the JEB net update problem, (i.e., only the destination node ever getting updated), but it will allow for considerably better routing decisions to be made. For instance, referring to Fig. 2, a message going from B to H after a cost change on the E->F link, a new path would be followed if E is automatically made aware that a cost change has taken place. Given this awareness, B will still route the message to E as it will still think that it is the best path, but E will not route the message through the E->F link, as it would now be aware that the line is no longer a good path. Instead, the final path of B->E->G->H would be followed with a cost of 7 — a considerable improvement over a cost of 19.

Unfortunately, this improvement is not without a penalty. Since E is immediately aware of the cost change on link E->F, a message sent from B to H is never routed through the E->F link. This has the effect of ensuring that not only is node B kept unaware of the cost change, but also node H is kept in the dark. This results in a message being sent from H to B, using the path H->F->C->B with a cost of 11 rather than the better path, H->G->E->B. Originally, total cost for messages going from B to H and from H to B, after the link cost change, would be 29 (19+10). Making the nodes aware of cost changes reduces the total cost to 18 (7+11) — a definite improvement. With immediate local link cost awareness, any destinations that would normally have used the E-F link (not just nodes B, H), would pass their messages to node E (or F), which would then choose an alternate, cheaper route.



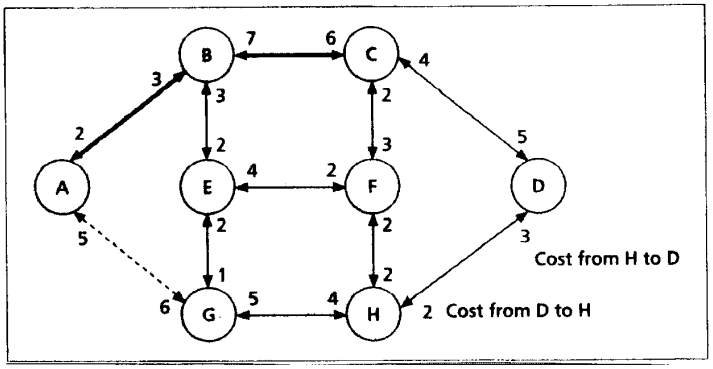
■ Figure 4. *A routing example using the JEB net.*



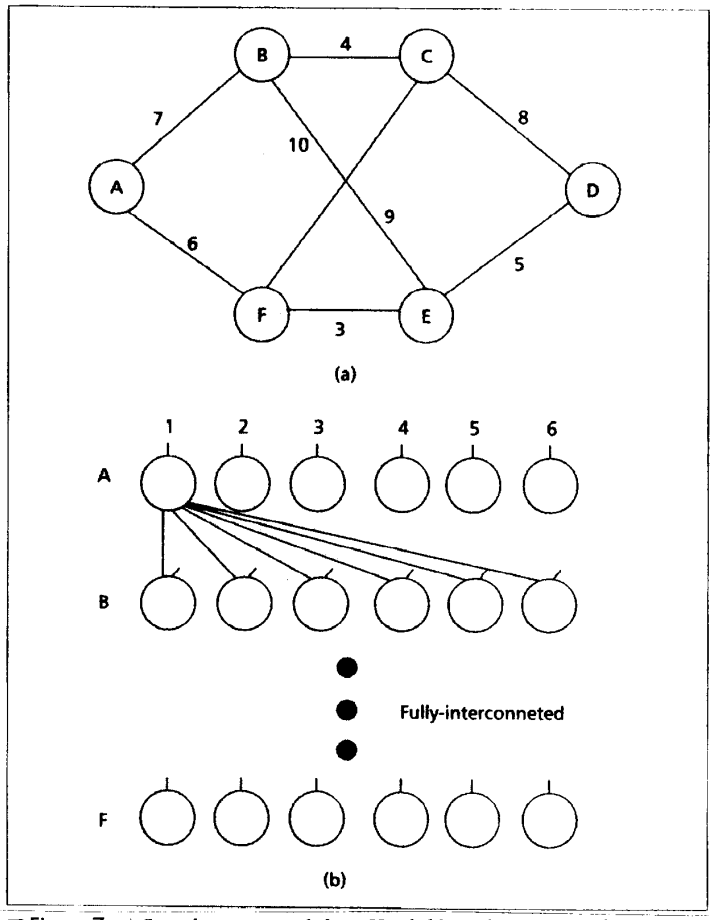
■ Figure 5. *A routing example after link cost change.*

Problem 3 — How to handle downed links — Since a downed link effectively blocks any message from passing through it, the JEB network at each node could never learn that a link was downed. Downed links might trap messages at a node and result in unroutable messages.

Solution 3 — To solve this problem requires the use of local link awareness, at least to the extent that nodes are aware that a link is down. Unfortunately, this only partially solves the problem of routing around downed links. In certain situations, it would be possible for a message to be undeliverable due to running into a downed link despite an alternative path being available. For example, referring to Fig. 6 where a message from C to G is being sent: the dashed line indicates that the link between A and G is down and the heavy lines indicate the partial path of the message from C to G.



■ Figure 6. *A routing example with downed link.*



■ Figure 7. *a) Sample test network for a Hopfield net; b) corresponding Hopfield network.*

As can be seen from Fig. 6, the network begins by attempting to route the message using the optimal path, which is C->B->A->G. However, upon reaching node A it encounters a downed link to G, making the message undeliverable from A. At this point, some sort of backtracking mechanism must be used to back up the route till reaching a node with an alternate route, as otherwise the network will report the message as undeliverable despite the fact that alternative paths are available.

Hopfield Network for Routing

The second neural network implemented to solve the routing problem is the Hopfield network. This approach is based on a solution proposed for the traveling salesman problem (TSP) as given in [8]. The message routing problem is considerably similar to the TSP problem in that it is concerned with finding an optimal route which travels between a source and destination node (city).

In order to map the optimal message routing problem onto a Hopfield neural network architecture, the same approach used in [8] to solve the TSP problem is taken to construct the network for this problem. This architecture requires n sets of n neurons where n is the number of nodes in the message network. One set of n neurons is assigned to each node. The output of that particular set of neurons determines what position in the path to be followed. This approach allows for either a centralized routing scheme or an isolated routing scheme (a special case of distributed routing) to be implemented. For a centralized scheme, the entire path found from the Hopfield network would be included with the message. In an isolated scheme, each node would have a Hopfield net which is used to find the next node the message needs to be passed to.

Figure 7 presents a sample message network and its corresponding Hopfield net. Assuming a message was being routed from A to D, using the path A->F->E->D, the output of the network would be interpreted to be as shown in Table 1.

The ones indicate these nodes forming a path, A->F->E->D, for a message sent from node A to D to follow. For the TSP problem, there would be a 1 in every row and column to form a complete closed path traversing all nodes. For a routing problem, it is not necessary nor desirable to visit all nodes. In order to satisfy the requirements of routing, the energy equation is constructed as follows.

$$E = A/2\,\Sigma_X\,\Sigma_i\,\Sigma_{i\neq j}\,V_{Xi}V_{Xj} + B/2\,\Sigma_i\,\Sigma_X\,\Sigma_{X\neq Y}\,V_{Xi}\,V_{Yi} + C/2\,(\text{valid path})^2 + D/2\,\Sigma_X\,\Sigma_{Y\neq X}\,\Sigma_i\,d_{XY}\,V_{Xi}\,(V_{Y,i+1} + V_{Y,i-1})$$

The evolution of Hopfield's net is described by the differential equation

$$du_{Xi}/dt = -u_{Xi}/\tau - A \Sigma_{j \neq i} V_{Xi} - B \Sigma_{Y \neq X} V_{Yi} - C(<\text{valid path}>) - D \Sigma_Y d_{XY} (V_{Y,i+1} + V_{Y,i+1})$$

In the above equations, the term <valid path> is 0 if the current output of the Hopfield network represents a valid routing path and 1 otherwise. A hard-limiter is used to calculate the outputs rather than the sigmoid function.

Simulation Results

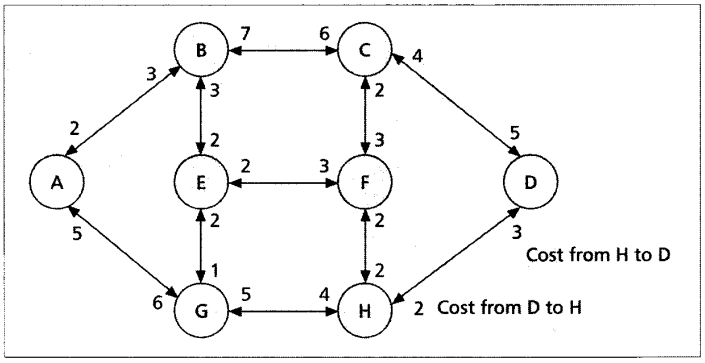
A network simulator was written to investigate the performance of the JEB net and the Hopfield net in routing messages. Network configurations shown in Figs. 8 and 9 are used to test the JEB neural router. The Hopfield neural router is tested using the network shown in Fig. 7a. A snapshot of the output of the simulator is included in the Appendix at the end of this aritcle. This simulator is capable of displaying various message network configurations and of running scripts to simulate message traffic. These scripts specify what messages are sent, any cost changes which occur, and what links go broken. To create these scripts, a random script generator was also written. This generator is capable of generating scripts of any desired length. To determine the mix of message types (send message, change cost, up/down link), percentages reflecting the chance of any particular type of message occurring can be passed into the program to generate a script. The following assumptions are used to generate the scripts and explain the simulation environment:

- The uniform message distribution is used to generate the message destination, implying that the probability of node *i* sending a message to node *j* is the same for all *i*, *j*, and *i j*.
- The propagation time is omitted; the link cost is the queuing cost only.
- All links are equally likely to become down, and all downed links are equally likely to become up.
- A command determines an action and, at one time, only one command is executed. The format of a command (an action) is shown in the Appendix.
- The link costs of test networks for the JEB neural net are asymmetric.
- The link costs of test networks for the Hopfield neural net are symmetric.
- The simulation results are obtained by averaging more than 30 runs of scripts with the same characteristics, i.e., same percentage of link cost changes, link down, link up, etc.)

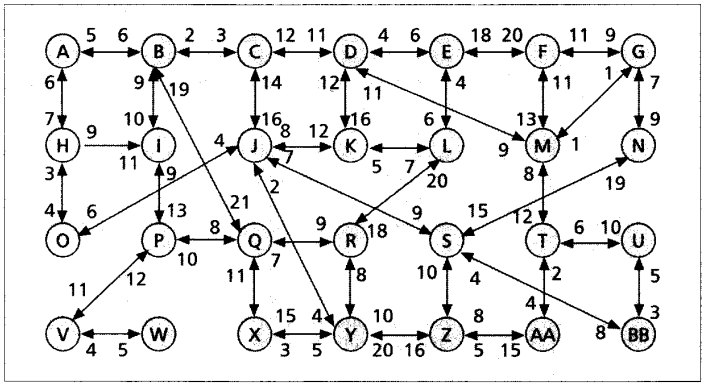
For the purpose of simulating different message routing conditions, several scripts were generated. Each script was generated randomly using a script generator. The command line for the script generator is shown in the Appendix. The script used during learning consists of 20,000 actions where an action is simply sending a message to a destination node, with no variations in the network. The value 20,000 is determined by the largest network size, i.e., test network two in Fig. 9. The scripts used for testing consist of a total of 200 actions, where an action could be either sending a message, changing a link cost or breaking a link. For these scripts, moderate (high) variation in cost is defined as a 15 percent (30 percent) chance of an action being a cost change. A moderate (high) chance of a downed link is defined as a 5 percent (10 percent) chance of an action causing a link to become broken. These percentages reflect the approximate number of actions of that particular type that would be present in a script. For example, with moderate variation in cost, 30 of the 200 actions would be cost change actions. By making use of these different types of scripts, networks with little or considerable fluctuation in their topology can be modeled. There are five scripts used for testing the network routing performance in our simulation as listed in Table 2.

	1	2	3	4	5
A	1	0	0	0	0
B	0	0	0	0	0
C	0	0	0	0	0
D	0	0	0	1	0
E	0	0	1	0	0
F	0	1	0	0	0

■ **Table 1.** *Example of a Hopfield output.*



■ Figure 8. *Simulation test network one.*



■ Figure 9. *Simulation test network two.*

Script 1	No variation in network
Script 2	Moderate variation in link costs, no downed links
Script 3	High variation in link costs, no downed links
Script 4	Moderate variation in link costs and moderate chance of downed links
Script 5	Moderate variation in link costs and high chance of downed links

■ **Table 2.** *Five scripts used for testing the network routing performance in the simulation.*

	Optimal initialization	Local link awareness	Message echoing
JEB 1	OFF	OFF	OFF
JEB 2	ON	OFF	OFF
JEB 3	ON	ON	OFF
JEB 4	ON	OFF	ON
JEB 5	ON	ON	ON
JEB 6	OFF	ON	ON

■ **Table 3.** *JEB net option settings.*

JEB Network Test Results

In testing the JEB network, various options were used or not used. These options are as follows:

Optimal Initialization — When turned on, the JEB nets would be initialized with the current optimal paths when the simulation starts up. If not on, nets are initialized only with the cost to the node's neighbors.

Local Link Awareness — When activated, nodes are aware of all cost changes to links to which the node is directly connected. If off, nodes can only learn of cost changes when a message is received.

Message Echoing — Destination nodes echo back to source nodes the cost of the path. Allows for both source and destination nodes to learn from a single message.

The JEB network solution was tested using the five scripts with different options. Table 3 presents the option settings for the JEB network.

We ran each script 30 times, for example, 30 versions of script 3, and computed the average value over these 30 runs. From our simulation results, it shows that JEB 5 has the best performance. Optimal initialization is an important factor for the JEB network to achieve satisfactory performance. Tables 4 and 5 summarize the test results obtained from running the shortest path algorithm, the static routing algorithm, the JEB 5 algorithm and the JEB 6 algorithm. Because the JEB 6 algorithm does not start with optimal initialization, we apply the learning script to the JEB 6 first before testing. The simulation results for all JEB 1 to JEB 4 are not included for brevity (they are available upon request) The numbers indicate the percentage of messages which were routed using the optimal or near-optimal path by the specified algorithm. For these tables, the near-optimal path is defined as being within one or two cost points of the optimal path cost as determined by the shortest path algorithm.

Test Results Discussion — Examining the test results of the JEB network reveals several interesting aspects of its operation. Note that to properly evaluate the performance of an algorithm, both the total cost of the routable messages and the number of unroutable messages should be examined.

First, it can be seen that the JEB nets which have gone through the learning phase give results as good as the JEB net with optimal initialization. This is to be expected, as a significant number of messages must be sent before the individual JEB nets really learn what the optimal paths are. The JEB network performs better when only moderate variation is present as opposed to high variation. When running with moderate variation, the JEB nets outperform the static routing by a fair amount, but do worse than the static routing when running with high variation. Actually, this behavior should be expected. Since the JEB nets depend on message traffic to update themselves, if there is a lot of variation as compared to the volume of message traffic, the JEB nets are unable to update themselves fast enough to keep up with the network fluctuations.

Also, the JEB nets perform considerably better than the static routing when downed links are present. In some cases, performance approaches the shortest path, with only a few more unroutable messages than the shortest path generates. Finally, it should also be noted that, in general, the JEB nets perform best with all options activated (optimal initialization, local link awareness, and message echoing).

Overall, the performance of the JEB network is satisfactory. This approach shows that it is able to adapt reasonably to changing network traffic and topologies while producing optimal or near-optimal path solutions.

Hopfield Test Results

The major advantage of the Hopfield network is that an analog circuit can be constructed to solve an optimization problem. But in simulation, the Hopfield network does not always generate valid solution especially when applied to large systems. Table 6 below summarizes the test results obtained from running the static routing algorithm and the Hopfield algorithm. The shortest path algorithm is also included in the table for comparison purposes. The numbers indicate the percentage of messages which are routed using the optimal or near-optimal path by the specified algorithm. For this table, the near-optimal path is defined as being within one or two cost points of the optimal path cost as determined by the shortest path algorithm.

Test Results Discussion — The Hopfield network is able to deliver reasonable results. Testing with script 1 yields 89 percent of the messages being routed along optimal or near-optimal paths. Even the worst performance under script 2 still has two-thirds of the messages being routed on optimal or near-optimal paths. These results are fairly good when one considers that no pre-initialization has been performed on the Hopfield network. The network,

starting from a cold start, is able to deliver the optimal path for the message in many cases.

Since the Hopfield network can be implemented by the analog circuit, which is one order faster than any digital table lookup implementation and is smaller in size, it could be applicable to a situation where time and space are critical and near-optimal solution is acceptable. In [9], it has demonstrated that the Hopfield network can be used to solve larger problems if the Lagrange multiplier method is used.

Comparison Between Neural and Other Routing Algorithms

The commonly used routing algorithms are the Bellman-Ford algorithm, Dijkstra algorithm, and the Floyd-Warshall algorithm. The Bellman-Ford algorithm iterates on the number of hops in a route. It is based on first finding optimal routes from a prescribed source node to all other nodes (subject to the constraint that no route contains more than one hop), then finding the optimal routes subject to the constraint no route contains more than two hops, and so forth. The amount of computation is at worst $O(N^3)$. Dijkstra's algorithm iterates on the length of a route. The general idea is to find the shortest routes in order of increasing route length. It first finds the shortest route from some node to the destination node and establishes this route. It then finds the next shortest route and establishes it, and so forth. The number of computation in the worst case is $O(N^2)$. The Floyd-Warshall algorithm iterates on nodes allowed as intermediate nodes in routes. It computes shortest routes between all pairs of nodes. The Floyd-Warshall algorithm is used in some networks with centralized routing such as TYMNET.

The JEB neural routing method can be considered an isolated routing algorithm in the sense that no information exchanged between nodes. In distributed routing algorithms, such as the distributed asynchronous Bellman-Ford algorithm, the computation of routes is shared among network nodes with information exchanged between them. Dijkstra's algorithm needs to know the current link status of the network for computation of the shortest routes. Hence each node needs to broadcast to all other nodes its connectivity to neighbors. Now we construct a table to compare the neural network approach, using JEB 6, with the distributed asynchronous Bellman-Ford algorithm and the Dijkstra algorithm. N is the number of nodes in the network.

The amount of computation of the JEB network is approximated in terms of the number of messages required to learn all the shortest routes in the network. For a network with N nodes, a node, say node i, needs to receive at most $N-1$ messages from node j to learn all possible routes from node i to node j. Hence a node has to receive at worst $(N-1)^2$ messages from all other nodes to learn the shortest routes for all other nodes. A network with N nodes, the total number of messages used during learning is $O(N^3)$. Since each node has a neural network, all network nodes can start the learning process at the same time. To avoid link contention, this learning process should be done in a distributed asynchronous manner, which is similar to the distributed asynchronous Bellman-Ford algorithm. The major advantage of the neural approach is no communication cost. Each node updates its JEB network upon detection of a cost change or a failed link through the incoming messages. Dijkstra's algorithm has the least computation load, but it has the largest communication overhead among the three approaches in Table 7.

	script1	script2	script3	script4	script5
Shortest path	100	100	100	100	100
Static routing	100	57	64	48	41
JEB 5	100	77	58	68	73
JEB 6	100	77	58	68	73

(JEB 6 is tested after running learning script.)

■ **Table 4.** *JEB net test results for test network one.*

	script1	script2	script3	script4	script5
Shortest path	100	100	100	100	100
Static routing	100	62	64	55	51
JEB 5	100	71	63	69	75
JEB 6	100	71	63	69	75

(JEB 6 is tested after running learning script with 20,000 actions.)

■ **Table 5.** *JEB net test results for test network two.*

	Script1	Script2	Script3	Script4	Script5
Shortest	100	100	100	100	100
Static	100	80	74	78	63
Hopfield	89	66	68	72	72

■ **Table 6.** *Test results summary for the Hopfield network.*

Complexity	Computation	Communication	Adaptivity
Asynchronous distributed Bellman-Ford	$O(N^3)$	Each node broadcasts to its neighbors of the latest route cost	Yes
Dijkstra	$O(N^2)$	Each node broadcasts to all other nodes of its connectivity to neighbor nodes	Yes
Neural nets (JEB 6)	$O(N^3)$ = no. of messages required in learning	None	Yes

■ **Table 7.** *A comparison of the neural approach and other routing algorithms.*

Conclusions and Future Work

The results obtained using the enhanced JEB network and Hopfield network show that the problem of optimal message routing can successfully be solved through the application of neural networks.

The problem of optimal message routing can successfully be solved through the application of neural networks.

The major problem of the JEB network is the unroutable messages even when there is a valid path. This problem could be solved through the use of backtracking mechanisms to back up the route until an alternate route can be found. Another possibility for solving the problem of messages getting trapped when using the JEB network might be making use of the second neural network. This second network could be used to store the network topology, but, for this network, only the interconnections would be stored, not the costs. This second neural network can be a two-level associative memory, which has been designed and tested through simulation on the New Jersey LATA network for automatic network restoration [11]. This would, in effect, give each node a current map of the available paths with the network, something which the JEB network does not provide.

To solve the unroutable message problem, each node would consult this second network before making its routing decision to verify that the node is not about to send the message into a trap. In order for this to be effective, the incoming message would need to contain the path it has followed up to this point. This would allow the node to determine the proper routing for the message by using the stored network topology and the path the message has already traveled.

Acknowledgment

The authors would like to thank the anonymous reviewers for their comments, which have improved the quality of this article significantly.

References

[1] D. Bertsekas and R. Gallager, Data Networks, (Prentice-Hall, 1987).
[2] A. Hirmatsu, "ATM Communications Network Control by Neural Network," Proc. of IJCNN 89, 1989, pp. I-259 I-266.
[3] T. X. Brown, "Neural Networks for Switching," *IEEE Commun. Mag.*, vol. 27, no. 11, Nov. 1989, pp. 72-81.
[4] T. P. Troudet and S. M. Walters, "Hopfield Neural Network Architecture for Crossbar Switch Control," *IEEE Tran. on Circuits and Systems*, vol. CAS-38, Jan. 1991, pp. 42-57.
[5] R. J. T. Morris and B. Samadi, "Neural Networks in Communications: Admission Control and Switch Control," ICC Conf. Record, June, 1991, pp. 21.6.1-21.6.7.
[6] J. E. Jensen, M. A. Eshera, and S. C. Barash, "Neural Network Controller for Adaptive Routing in Survivable Communications Networks," Proc. IJCNN 90, 1990, pp. II-29 and II-36.
[7] G. A. Carpenter and S. Grossberg, "ART 2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns," *Applied Optics*, vol. 26, no. 23, Dec. 1987, pp. 4919-4930.
[8] J. J. Hopfield and D. W. Tank, "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics*, vol. 52, 1985, pp. 141-152.
[9] J. E. Wieselthier, C. M. Barnhart, and A. Ephremides, "The Application of Hopfield Neural Network techniques to Problems of Routing and Scheduling in Packet Radio Networks," ONR technical research report, Nov. 1990.
[10] E. Sanchez-Sinencio and C. Lau, eds., "Artificial Neural Networks — Paradigms, Applications and Hardware Implementations" (IEEE Press, 1992).
[11] C-J. Wang and H. Y. Zhou, "Automatic Network Restoration using Two-Level Associative Memories," Proc. IEEE 1994 International Conference on Neural Networks, pp. 3565-3570.

Biographies

CHIA-JIU WANG [M '88] is an associate professor of electrical and computer engineering at the University of Colorado at Colorado Springs. His primary research interests are in network performance evaluation and modeling, distributive self-learning message routing, computational intelligence, and computer architecture. He received a B.S. in physics from National Central University, Taiwan, and M.S. and Ph.D. degrees in electrical engineering from Tatung Institute of Technology, Taiwan and Auburn University, respectively. Since 1988 he has been a member of the faculty at University of Colorado at Colorado Springs. His e-mail address is: cwang@vlsia.uccs.edu.

PAUL N. WEISSLER received B.S.E.E. and M.S.E.E. degrees from the University of Colorado at Boulder and at Colorado Springs. He has worked for Boeing Electronics Company, Lexico Enterprises, and Kaman Science Corporation as a software engineer. His research interests are computational intelligence, software engineering, and network simulation.

Appendix

1. The format of a command is given below.

Send message -	Send	<from node> <to node>	
Change cost -	Change	<from node> <to node>	<new cost>
Down a link -	Downlink	<from node> <to node>	
Down a node -	Downnode	<node>	
Up a link -	Uplink	<from node> <to node>	
Up a node -	Upnode	<node>	

2. The command for the script generator is :

scrptgen <length> <% send messages> <% cost changes> <%down links> <% down nodes>

3. A snapshot of the output of the network routing simulator: