Network Life Time Optimization in Wireless Sensor Networks Through Energy Efficient Delay Leap Routing with Bio-Inspired Computing

[1]Dr. Vishant Kumar
[1]Professor, [1]Department of CSE,
J. B. Institute of Technology, Dehradun,
U.K. India

[2]Dr. Joginder Singh Chauhan
[2]Professor
[2]Department of CSE, I.P.S. Saharanpur,
U.P. India

***Abstract*: Energy efficiency is one of most important parameter to consider while designing the wireless sensor networks. In today's world controlling the media traffic is one of the major goal facing contemporary scientists which is encouraging applications in Multimedia in such a way to ensure Quality service should be delivered to the end users. To ensure Quality service scientists are keeping in mind one end user to another end user delay leap, delay jitter leap, minimum bandwidth etc. In this paper, we proposed energy efficient delay leap routing scheme based on Hopfield type neural network model using artificial neural network. We report a development of a Hopfield type neural network model to solve minimum cost delay leap multicast routing problem. The delay leap optimized paths from source to various destinations are obtained recursively, which are then combined together using a union operator to ensure that all the links are participating in multicast route path only once. The following sections describe briefly energy saving scheme for network life time optimization, the experiments performed and results achieved in this work, and the main conclusions drawn from our experimental results.**

Index Terms:— Wireless Sensor Network (WSN), Artificial Neural Network (ANN), Hopfield Neural Network (HNN).

I. INTRODUCTION

A sensor network is a network of many smart devices, called sensor nodes, which are distributed in order to perform an application-oriented global task. The primary component of a sensor network is the sensor. Sensor nodes are used to monitor some real world physical phenomenon like pressure, temperature, humidity, presence/ absence of something, vibration, intensity, sound, pressure, motion and pollutant etc. Each of the sensor nodes is small and inexpensive but smart enough to perform several tasks. The smart tiny sensor nodes are equipped with microcontroller, a radio transmitter and an energy source. The most important design and implementation requirements of a typical sensor network are energy efficiency, routing decision type, memory capacity, computational speed and bandwidth. Sometimes a special central node is deployed to control all the operations of the network. Wireless sensor network, widely known as WSN, has become a wide area of research nowadays as the processing and storage technology became mature and feasible recently. We can now think of deploying hundreds of thousands of cheap sensor nodes to a target area to sense some special types of information. Sensor networks are used in a wide range of areas such as sensing ocean floor activities, volcanic activities, and phenomena in Mars etc. In most of the cases, sensor nodes are deployed in places where the physical condition is very adverse and human cannot go there. For this reason, there is no pre-existing network topology available in the target area. The major problem of the sensor network is the energy consumption. The battery technology is much lagging behind than the processing and storage technology. For this reason, the primary research on sensor network is how to efficiently use the power consumption of each of the sensor nodes to maximize the total lifetime of the network. In most of the cases, it is not feasible or even impossible to recharge the battery of the sensor nodes. The power consumption of the sensor nodes mostly depends on the routing of information in the network. Other means of power consumption include sensing and processing of the information. There is not much scope to reduce the power consumption related to sensing and processing. So, most of the research about sensor networks mainly concentrates on the power consumption due to the routing of information and there is still much scope to improve. Compression could reduce the amount of data transmitted in sensor network. However, data compression is out of the scope of this research and hence we will not discuss about data compression.

Wireless sensor networks are potentially one of the most important technologies of this century. Recent advancement in wireless communications and electronics has enabled the development of low-cost, low-power, multifunctional miniature devices for use in remote sensing applications. The combination of these factors has improved the viability of utilizing a sensor network consisting of a large number of intelligent sensors, enabling the collection, processing analysis and dissemination of valuable information gathered in a variety of environments. A sensor network is composed of a large number of sensor nodes which consist of sensing, data processing and communication capabilities. Sensor network protocols and algorithms must possess self-organizing capabilities. Another unique feature of sensor networks is the cooperative effort of sensor nodes. Sensor nodes are suitable with an onboard processor. Instead of sending the raw data to the nodes responsible for the fusion, they use their processing abilities to locally carry out simple computations and transmit only the required and partially processed data. Sensor networks are predominantly datacentric rather than address-centric. So sensed data are directed to an area containing a cluster of sensors rather than particular sensor addresses. Given the similarity in the data obtained by sensors in a dense cluster, aggregation of the data is performed locally. That is, a summary or analysis of the local data is prepared by an aggregator node within the cluster, thus reducing the communication bandwidth requirements. Aggregation of data increases the level of accuracy and reduces data redundancy. A network hierarchy and clustering of sensor nodes allows for network scalability, robustness, efficient resource utilization and lower power consumption. The fundamental objectives for sensor networks are reliability, accuracy, flexibility, cost effectiveness and ease of deployment.

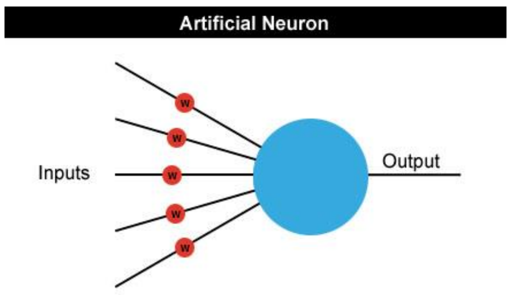
II. ANN: ARTIFICIAL NEURAL NETWORKS

A neural network is mans crude way of trying to simulate the brain electronically. So to understand how a neural net works we first have a look at how the old grey matter does its business.

Our brains are made up of about 100 billion tiny units called *neurons*. Each neuron is connected to thousands of other neurons and communicates with them via electrochemical signals. Signals coming into the neuron are received via junctions called *synapses*, these in turn are located at the end of branches of the neuron cell called *dendrites*. The neuron continuously receives signals from these inputs and then performs a little bit of magic. What the neuron does (this is over simplified I might add) is sum up the inputs to itself in some way and then, if the end result is greater than some threshold value, the neuron fires. It generates a voltage and outputs a signal along something called an *axon*. Just have a good look at the illustration and try to picture what is happening within this simple little cell.

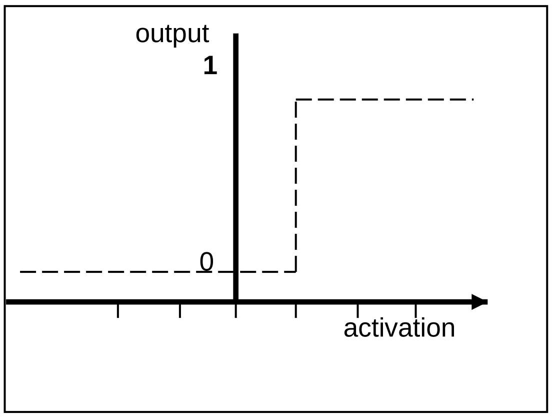
Neural networks are made up of many artificial neurons. An artificial neuron is simply an electronically modeled biological neuron. How many neurons are used depends on the task at hand. It could be as few as three or as many as several thousand.

One optimistic researcher has even hard wired 2 million neurons together in the hope he can come up with something as intelligent as a cat although most people in the AI community doubt he will be successful. There are many different ways of connecting artificial neurons together to create a neural network but I shall be concentrating on the most common which is called a *feed forward* network.



Each input into the neuron has its own weight associated with it illustrated by the red circle.

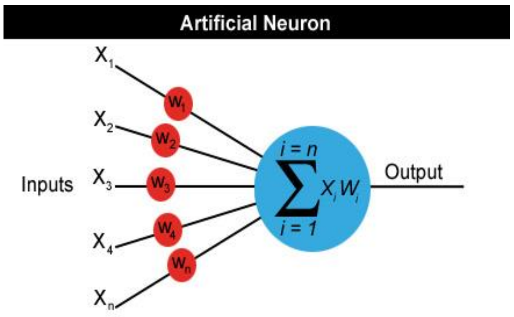
A weight is simply a floating point number and it's these we adjust when we eventually come to train the network. The weights in most neural nets can be both negative and positive, therefore providing excitory or inhibitory influences to each input. As each input enters the nucleus (blue circle) it's multiplied by its weight. The nucleus then sums all these new input values which gives us the *activation* (again a floating point number which can be negative or positive). If the activation is greater than a threshold value - lets use the number 1 as an example - the neuron outputs a signal. If the activation is less than 1 the neuron outputs zero. This is typically called a *step* function as shown in figure below:



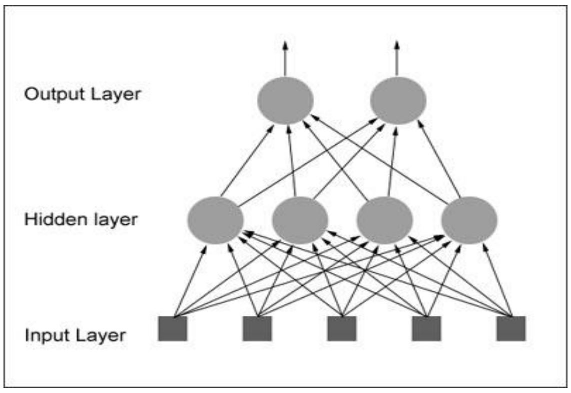
A neuron can have any number of inputs from one to n, where n is the total number of inputs. The inputs may be represented therefore as x_1, x_2, x_3… x_n. And the corresponding weights for the inputs as w_1, w_2, w_3… w_n. Now, the summation of the weights multiplied by the inputs we talked about above can be written as $x_1w_1 + x_2w_2 + x_3w_3 \ldots + x_nw_n$, . So, the activation value is

$$a = x_1w_1+x_2w_2+x_3w_3... +x_nw_n .$$

Fortunately there is a quick way of writing this down which uses the Greek capital letter sigma S, which is the symbol used by mathematicians to represent summation.



Well, we have to link several of these neurons up in some way. One way of doing this is by organizing the neurons into a design called a *feed forward network*. It gets its name from the way the neurons in each layer feed their output forward to the next layer until we get the final output from the neural network. This is what a very simple feed forward network looks like:



Each input is sent to every neuron in the hidden layer and then each hidden layer's neuron's output is connected to every neuron in the next layer. There can be any number of hidden layers within a feed forward network but one is usually enough to suffice for most problems you will tackle. Also the number of neurons I've chosen for the above diagram was completely arbitrary. There can be any number of neurons in each layer, it all depends on the problem.

III. ENERGY EFFICIENT DELAY LEAP ROUTING

a. INTRODUCTION

In today's world controlling the media traffic is one of the major goal facing contemporary scientists which is encouraging applications in Multimedia in such a way to ensure Quality service should be delivered to the end users. To ensure Quality service scientists are keeping in mind one end user to another end user delay leap, delay jitter leap, minimum bandwidth etc. The various minimum Steiner tree heuristics have been reported in literature [1-3]. The multicast tree, a NP complete problem, for communication network was first formulated by Kompella et al. [4]. Noronha and Tobagi [5] proposed an algorithm based on integer programming to construct the optimal source-specific leap-constrained minimum Steiner tree. The Leap shortest multicast algorithm [6, 7] was used to solve delay constrained tree optimization problem. The algorithm started by computing a least delay tree rooted at a source and spanning all group members and iteratively replaced the superedges in the tree by cheaper superedges not in the tree. The paper [8] summarized a tradeoff algorithm between the minimum Steiner tree and the least delay tree. Barathkumar and Jaffe [9] studied the algorithms to optimize the cost and delay of the routing tree. Rouskas and Baldine [10] studied the problem of constructing multicast trees subject to both delay and delay jitter constraints. Salama et al. [11] compared the performance of shortest path broadcast tree algorithm and a heuristic for tree cost.

On the other hand Hopfield Neural Network (HNN) model is a generally using for solving constrained optimization problem. The Hopfield Neural Network is a parallel, distributed information processing structure consisting of many processing elements connected via weighted connections [12]. The objective function was then expressed as quadratic energy function and the associated weights between neurons were computed using the gradient descent of energy function. The formulation of energy function associated with Hopfield Neural Network for shortest path computation was first proposed by Ali and Kamoun [13]. Hopfield Neural Network has been used for computation of shortest path for routing in computer networks and communication systems [13-15]. The Hopfield Neural Network was used to solve Quality service delivery Constrained for real time multicast routing [16-19]. The optimization of multicast tree using Hopfield Neural Network with delay and delay jitter has been reported in [19].

This scheme reports a development of a Hopfield type neural network model to solve minimum cost delay leap multicast routing problem. The delay leap optimized paths from source to various destinations are obtained recursively, which are then combined together using a union operator to ensure that all the links are participating in multicast route path only once.

b. DESCRIBING DELAY LEAP MULTICAST ROUTING

The multicast network can simply be represented as a weighted connected network like *A = (V, W)*, where *V* denotes the set of vertices (nodes) and *W* denotes the set of arcs (links). Let *K* is a subset of *V* i.e. $K \subseteq V$ forms the multicast group with each node of *K* is a group member. The node $p \in V$ is a multicast source for multicast group *K*. A multicast tree $Tr(p, K) \subseteq W$ is a sub-graph of *A* that spans all nodes in *K*, while it may include non-group member nodes along a path in the tree. The link between nodes *i* and *j*, $w = (i, j) \in W$ has its properties, cost C_{ij} and delay D_{ij} as real positive values. The link cost C_{ij} may be the monitory cost incurred by the use of the network link or may be some measure of network utilization. The cost coefficients C_{ij} for the nonexistent arcs are defined as infinity. The delay D_{ij} represents the time needed to transmit information through link that includes transmission, queuing and propagation delays.

Delay leap multicast routing problem can be defined as to find a tree rooted at the source *s* and spanning to all the member of *K* such that the total cost of the links of the tree is minimum and the delay from source to each destination is not greater than the required delay constraint. Therefore, the end-to-end delay Leap multicast routing problem is defined as –

Minimize cost of multicast tree

$$Cost\,(T_r(p,K)) = \sum_{i,j \in T_r(p,K)} C_{ij} \qquad (3.1)$$

Subjected to End-to-end delay constraint

$$D_k(P_T(p,k)) = \sum_{i,j \in P_T(p,k)} D_{ij} \le \Delta \quad \text{for } k = 1, K \qquad (3.2)$$

The $P_T(p,k)$ is a path between source node *s* and the destination node *k*. The path $P_T(p,k)$ is an ordered sequence of nodes connecting from source node *k* to destination node *m*, indicating the traverse of data from '*p*' to '*k*' as-

$(p \rightarrow i \rightarrow j \rightarrow k \;\; ..... \rightarrow r \rightarrow d)$.

c. DELAY LEAP MULTICAST ROUTING WITH HOPFIELD NEURAL NETWORKS

The delay leap multicast tree using Hopfield neural networks is obtained by recursively obtaining the delay leap shortest paths from source to each destination in the multicast group and combining them by the union operator. The union operator ensures that a link is appearing only once in the multicast tree.

d. EXPERIMENTAL RESULTS

The total cost associated with path $P_T(p,k)$ is therefore expressed as –

$C_{pk} = C_{pi} + C_{ij} + C_{jk} + \ldots\ldots + C_{rm}$
(3.3)

The shortest path problem is aimed to find the path $P_T(p,k)$ that has the minimum total cost C_{pk}. It can be explored using Hopfiled neural network through the following procedure -

A $(N \times N)$ matrix $\boldsymbol{B} = [B_{ij}]$ is used where *N* is the number of nodes. The diagonal elements of matrix *B* are taken as zero. The each element in the matrix is representative of a neuron described by double subscripts *i* and *j* representing the node numbers. Therefore, only $N(N-1)$ neurons are to be computed and the neuron at the location (x, i) is characterized by its output B_{xi} defined as follows –

$$B_{xi}^{k} = \begin{cases} 1 & \text{if arc from node } i \text{ to node } j \text{ is in path otehervise} \\ 0 & \end{cases} \qquad (3.4)$$

Let define p_{xi} as-

$$p_{xi} = \begin{cases} 1 & \text{if arc from node } x \text{ to node } i \text{ does not exist otherwise} \\ 0 & \end{cases} \qquad (3.5)$$

The optimized path (minimum cost path) can only be obtained by minimizing constrained parameters. On minimizing the constrained parameters through an annealing schedule, the probability of visiting lower energy states increases. The energy function must favor states that correspond to valid paths between the specified source-destination pairs. Among these, it must favor the one which has the optimized path (minimum cost). An energy function satisfying such requirement is given by [13] –

$$E^k = E_1^k + E_2^k + E_3^k + E_4^k + E_5^k \qquad (3.6)$$

Such that,

$$E_1^k = \frac{W_1}{2} \sum_{x=1}^{n} \sum_{i=1}^{n} C_{xi} V_{xi}^k \qquad (3.6.1)$$

$$E_2^k = \frac{W_2}{2} \sum_{x=1}^{n} \sum_{\substack{i=1 \\ i \neq x}}^{n} P_{xi} V_{xi}^k \qquad (3.6.2)$$

$$E_3^m = \frac{W_3}{2} \sum_{x=1}^{n} \left\{ \sum_{\substack{i=1 \\ i \neq x}}^{n} V_{xi}^k - \sum_{\substack{i=1 \\ i \neq x}}^{n} V_{ix}^k \right\}^2 \qquad (3.6.3)$$

$$E_4^m = \frac{W_4}{2} \sum_{i=1}^{n} \sum_{\substack{x=1 \\ x \neq 1}}^{n} V_{xi}^k (1 - V_{xi}^k) \qquad (3.6.4)$$

$$E_5^k = \frac{W_5}{2} (1 - V_{kp}^k) \qquad (3.6.5)$$

Where, W_1 weight to force the minimum cost of the path by accounting cost of existing links, W_2 weight to prevent the nonexistent links being included in the chosen path, W_3 weight to ensure that if a node has been entered in, it will also be exited, W_4 weight to force the state of neural network to converge to one of the corner of the hypercube defined by $V_{xi} \in \{0,1\}$, W_5 weight to enforce the construction of path originating at source *p* and terminate at destination *k*.

The V^k obtained above represents the output matrix of unicast route or shortest path from source *p* to destination *k*. Final output of multicast tree is obtained by the union of unicast routes from source to various destinations. The element V_{xi} in multicast tree is obtained as –

$$V_{xi} = V_{xi}^1 \bigcup V_{xi}^2 \bigcup V_{xi}^3 \bigcup \ldots\ldots\ldots\ldots \bigcup V_{xi}^k \qquad (3.7)$$

We are here introducing a new energy term E_6^k, where one end to another end delay constraint is referred as penalty, can be added for delay leap optimized path. This energy function term is therefore expressed as –

$$E_6^k = \frac{W_6}{2} \sum_{x=1}^{n} \sum_{\substack{i=1 \\ i \neq x \\ (x,i) \neq (k,p)}}^{n} DL_{xi}^k V_{xi}^k \leq \Delta \qquad (3.8)$$

Where, W_6 weight to enforce that the delay of the constructed path is less than or equal to specified delay constant.

The total energy function for k^{th} destination E^k including delay leap is therefore defined as-

$$E^m = E_1^m + E_2^m + E_3^m E_4^m + E_5^m + E_6^k \tag{3.9}$$

Using eqn (3.6) and (3.9),

$$E^m = \frac{W_1}{2}\sum_{x=1}^{n}\sum_{\substack{i=1\\ i\neq x}}^{n} C_{xi}V_{xi}^k + \frac{W_2}{2}\sum_{x=1}^{n}\sum_{\substack{i=1\\ i\neq x}}^{n} p_{xi}V_{xi}^k$$

$$+\frac{W_3}{2}\sum_{x=1}^{n}\left\{\sum_{\substack{i=1\\ i\neq x}}^{n} V_{xi}^k - \sum_{\substack{i=1\\ i\neq x}}^{n} V_{ix}^k\right\}^2 + \frac{W_4}{2}\sum_{i=1}^{n}\sum_{\substack{x=1\\ x\neq i}}^{n} V_{xi}^k(1-V_{xi}^k)$$

$$+\frac{W_5}{2}(1-V_{kp}^k) + \frac{W_6}{2}\sum_{x=1}^{n}\sum_{\substack{i=1\\ i\neq x\\ (x,i)\neq(k,p)}}^{n} DL_{xi}^k V_{xi}^k \tag{3.10}$$

To explore the optimized Energy function depicted in eq. (3.10), the Hopfield net with the probabilistic update rule can be used. The probabilistic distribution of states will be stationary or independent of time for a network to be in stochastic equilibrium. On decreasing the constraint parameter according to the probabilistic annealing schedule the network produces a new energy landscape, which contains the minimum energy states with respect to the previous one. This process continues until the network reaches to the global minimum energy state. This state refers to optimized delay leap in shortest path.

The shortest path with delay leap has been given by the eq. (3.10). The probability distribution of these states can be given as

$$P(s) = \frac{1}{z} e^{-\frac{E^k(p)}{\xi_\upsilon}}. \tag{3.11}$$

Where, *z* represents the partition function, *P(p)* represents the probability of visiting state *p* and $E^k(p)$ represents the energy function of state *p*.

According to this probability distribution the probability of visiting the lower energy states decreases as the network approaches to higher energy states and it also shows that for the larger value of constraint parameter ξ_υ, the probability of visiting the lower energy states decreases. Now, as the constraint parameter is reduced as per the annealing schedule of mean field approximation, the probability of visiting the lower energy states increases. Finally at the allowable lower limit of constraint parameter i.e. $\xi_\upsilon \approx 0$, the probability of visiting the lower energy states approaches to 1 (i.e. highest probability), so that the network settles in the minimum energy state, which describes the optimized delay leap shortest path amongst the multicast routing. Since, the implementation of simulated annealing requires computation of stationary probabilities at equilibrium state for each annealing schedule. To speed up this process, we may use mean field approximation [16] in which the stochastic update of bipolar units is replaced with deterministic states. The basic idea of mean field approximation is to replace the fluctuating activation values of each unit by their average values [17]. Activation of a i^{th} unit can be given as

$$\langle x_i \rangle = \left\langle \sum_i W_{ij} S_i \right\rangle,$$

$$\text{or } \langle x_i \rangle = \sum_i W_{ij}\langle S_i \rangle, \tag{3.12}$$

where, $\langle S_i \rangle$ is the average of the states of i^{th} node and W_{ij} represents the weight with ith and jth node.

$$\text{and} \qquad \langle S_i \rangle = \tanh\left(\frac{x_i}{\xi_\upsilon}\right). \tag{3.13}$$

In mean field approximation the activation of i^{th} unit x_i is replaced by $\langle x_i \rangle$, so that using equation (3.13), we have

$$\langle S_i \rangle = \tanh\left(\frac{1}{\xi_\upsilon}\sum_j W_{ij}\langle S_j \rangle\right). \tag{3.14}$$

The set of these equations is a result of minimization of an effective energy defined as a function of constraint parameter [18]. Thus, eq. (3.14) may be expressed as

$$\langle S_i \rangle = \tanh\left[-\frac{1}{\xi_\upsilon}\frac{\partial E^m(\langle S \rangle)}{\partial(\langle S_i \rangle)}\right]. \tag{3.15}$$

where the change in energy function for the average states of i^{th} unit is given by

$$\frac{\partial E^m(\langle S \rangle)}{\partial(\langle S_i \rangle)} = -\sum_{i\neq j} W_{ij}\langle S_j \rangle. \tag{3.16}$$

The non-linear deterministic equations given by eq. (3.15) are solved iteratively. As the constraint parameter is lowered to the minimum value, the steady equilibrium values of $\langle S_i \rangle$ will be obtained. At the allowable lower limit of constraint parameter the probability of visiting the minimum energy states has the maximum value i.e. $P(p)=\frac{1}{z}e^{-\frac{E^k(p)}{\xi_\upsilon}}\approx 1$, so that the network will achieve one of the minima of energy landscape. At this global minimum of the energy landscape the average value of the state of the network will represent the optimized energy function for delay leap optimized path in multicast routing.

i.e. $$\langle S_i \rangle = kD_{op}\,, \tag{3.17}$$

where m is the proportionality constant and D_{op} is the delay leap optimized path.

CONCLUSION

Hopfield neural network model with stochastic annealing is hereby successfully applied to obtain minimum cost delay leap constrained parameter for multicast tree. The multicast tree is obtained by recursively obtaining the delay leap optimized path from source to various destinations and combining them by union operator. The union operator ensures that a link is appearing only once in the multicast tree. The minimum energy function is obtained with minimization of constrained parameter as per a defined annealing schedule, which increases the probability of visiting lower energy states. Finally, the goal of minimization of objective function (minimum cost delay leap route) is achieved by using mean filed approximation with stochastic annealing process of reducing constrained parameter. It is suggested here, that in future, the effectiveness of the developed scheme should be tested to obtain optimum multicast trees for different sets of source and destinations on any weighted connected network.

ACKNOWLEDGMENT

I would like to thank professor and associate director Dr. Ashish Chaturvedi, ASCS, Arni University, Kathgarah, state of Himachal Pradesh, India for providing the research assistance throughout.

REFERENCES

[1] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for Steiner trees. Acta Informatica, vol. 15, no. 2, pp. 141-145, 1981.

[2] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. Mathematica Japonica, vol. 24, no. 6, pp. 573-577, 1980.

[3] V. R. Smith. The computation of nearly minimal Steiner trees in graphs. International Journal of Mathematical Education in Science and Technology, vol. 14, no. 1, pp. 15-23, 1983.

[4] V. Kompella, J. Pasquale, and G. Polyzos. Multicast routing for multimedia communication. IEEE/ACM Transactions on Networking, vol. 1, no. 3, pp. 286-292, 1993.

[5] C. Noronha and F. Tobagi. Optimum routing of multicast streams. Proceedings of IEEE INFOCOM '94, pp. 865-873, 1994.

[6] Q. Zhu, M. Parsa, and J. G. L. Aceves. A source-based algorithm for delay-constrained minimum-cost multicasting. Proceedings of IEEE INFOCOM '95, pp. 377-385, 1995.

[7] M. Parsa and Q. Zhu. An iterative algorithm for delay-constrained minimum-cost multicasting. IEEE/ACM Transactions on Networking, vol. 6, no. 4, pp. 461 – 474, 1998.

[8] L. Wei and D. Estrin. The trade-offs of multicast trees and algorithms. Proceedings of the Third International Conference on Computer Communications and Networking (IC 3N '94), pp. 17-24, 1994.

[9] K. B. Kumar and J. Jaffe. Routing to multiple destinations in computer networks. IEEE Transactions on Communications, vol. 31, no. 3, pp. 343-351, 1983.

[10] G. N. Rouskas and I. Baldine. Multicast routing with end-to-end delay and delay variation constraints. Proceedings of IEEE INFOCOM '96, pp. 353-360, 1996.

[11] H.. F. Salama, D. S. Reeves and Y. Viniotis. Evaluation of multicast routing algorithms for real-time communication on high-speed networks. IEEE Journal on Selected Areas in Communications, vol. 15, no. 3, pp. 332 – 345, 1997.

[12] J. J. Hopfield and D. W. Tank. Neural computation of decisions in optimization problems. Biological Cybernetics, vol. 52, pp 141-152, 1985.

[13] M. K. Ali and F. Kamoun. Neural networks for shortest path computation and routing in computer networks. IEEE Transactions on Neural Networks, vol. 4, no. 6, pp 941-953, 1993.

[14] H. E. Rauch and T. Winarske. Neural network implementation of the shortest path algorithm for routing communication traffic. IEEE Computer System Magazine, pp. 26-30, 1988.

[15] L. Zhang and S. C. A. Thomopoulos. Neural network implementation of the shortest path algorithm for traffic routing in communication network. Proceedings of International Conference on Neural Networks, 1999.

[16] S. Kirkpatrick, C. D. gelatt and M. P. Vecchi. Science. Vol. 220-671, 1983.

[17] G. L. Bilbro, W. E. Snyder, S. J. Garnier and J. W. Gault. IEEE Trans. Neural Networks. 3 no.1- 131, 1992.

[18] S. Haykin. Neural Networks: A Comprehensive Foundation. New York, Macmillan College. 338, 1994.