Delay-Constrained Multicast Routing Using the Noisy Chaotic Neural Networks

Lipo Wang, *Senior Member*, *IEEE*, Wen Liu, *Student Member*, *IEEE*, and Haixiang Shi

Abstract—We present a method to compute the delay-constrained multicast routing tree by employing chaotic neural networks. The experimental result shows that the noisy chaotic neural network (NCNN) provides an optimal solution more often compared to the transiently chaotic neural network (TCNN) and the Hopfield neural network (HNN). Furthermore, compared with the bounded shortest multicast algorithm (BSMA), the NCNN is able to find multicast trees with lower cost.

Index Terms—Multicast routing, noise, chaos, constrained Steiner tree (CST), neural networks.

1 Introduction

Multimedia communications and data delivery [1], [2], [3], [4] motivated research in multicast routing, resource reservation, and network architecture development. Along with progresses in audio, video, and data storage technologies, multicast routing is becoming more and more popular in recent years. The multicast routing problem is also called the Steiner tree problem, which aims to minimize the total cost of a multicast tree, and is known to be NP-complete (nondeterministic polynomial-time complete) [5].

For a real-time application, the routing algorithm needs to find an optimal multicast route that can offer sufficient resources to guarantee the required quality of service (QoS) [6], [7], [8]. This problem is called QoS-constrained multicast routing and was also proved to be an NP-complete problem [5]. Furthermore, QoS requirements may be interdependent, i.e., one factor has effects on another, which makes the problem more complicated.

QoS-constrained multicast routing problems include routing with the bandwidth requirement [6], many-to-many multicast routing [9], [10], and dynamic routing (the node may join or leave the network at any instance of time) [11]. We focus on delay-constrained multicast routing (DCMR), which aims to find an optimal tree between a source and a set of destinations while each path is subject to a given delay constraint. This is also called the constrained Steiner tree (CST) problem.

1.1 Review of the CST Problem

Kompella et al. [12] presented the first heuristic for the CST problem. It is a source-based heuristic, which assumes that the source node can obtain topology information about the communication network through the routing protocol. This algorithm failed frequently in cases where the link delays and delay constraints take noninteger values [13]. Besides, this heuristic may be impracticable when applied to directed communication networks [13]. Widyono [14] proposed four CST heuristics based on the constrained Bellman-Ford (CBF) algorithm and a merge algorithm. The constrained adaptive ordering (CAO) heuristic is the best among them. Sun and Langendoerfer [15] proposed another heuristic for the CST problem. It constructs a CST by merging a minimum cost tree with a minimum delay tree, where the trees are found by Dijkstra's algorithm. Parsa et al. proposed the bounded shortest multicast algorithm (BSMA) [16], [17]. In the first place, BSMA also finds the minimum-delay multicast tree through Dijkstra's algorithm. Then, the cost of the so-called *superedge* to each destination is reduced monotonously through a kth-shortest path algorithm [18], while the update must satisfy the delay constraint until no further optimization in the total cost is possible. Salama et al. [13] evaluated all these heuristic algorithms (Kompella et al.'s, CAO, and BSMA) under a high-speed networking environment and found that the BSMA is the best in terms of the total cost of the final multicast tree. However, compared with the constrained optimal minimum Steiner tree (COPT) algorithm presented by Manyem [19], BSMA's final costs are around 3 percent-7 percent more expensive than optima found by the COPT, respectively. The COPT is implemented by a branch-and-bound technique and therefore needs a very long execution time. As a result, the COPT may not be applicable in real time.

1.2 Review of Neural Networks for Routing Problems

By defining proper energy (cost) functions and deriving associated weights between neurons, one can use neural networks (NNs) to solve routing problems [20]. After Hopfield presented the Hopfield NNs (HNNs) [21] in 1982, many researchers have been exploring and improving the performance of HNNs on versatile applications. However, the HNN is easily trapped at local minima, especially in large networks [22]. To overcome this, Nozawa [23] proposed a chaotic NN by adding negative self-feedbacks into HNNs. The transiently chaotic NN (TCNN) proposed by Chen and Aihara [24] has been further developed in

• *The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Block S1, Nanyang Avenue, Singapore 639798. E-mail: {elpwang, liuw0004, pg02782641g}@ntu.edu.sg.*

Manuscript received 28 Jan. 2007; revised 4 Apr. 2008; accepted 7 July 2008; published online 25 July 2008.
Recommended for acceptance by S.R. Murthy.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0038-0107.
Digital Object Identifier no. 10.1109/TC.2008.127.

recent years, such as introducing some time-dependent parameters for richer dynamics [25], [26]. Wang et al. [27], [28] proposed stochastic chaotic simulated annealing (SCSA) by noisy chaotic NNs (NCNNs) with the addition of a decaying stochastic noise into the TCNN. Compared with TCNNs, NCNNs can solve the traveling salesman problem (TSP) more efficiently [27], [28].

The formulation of the energy function for NNs on routing problems was first proposed by Ali and Kamoun [29] and then improved in [30] and [31]. Pornavalai et al. [32] applied the HNN on the CST problem of an eight-node communication network. They modified the energy function of the unconstrained unicast routing problem to suit the delay-constrained unicast routing problem in the first step and then changed this energy function to solve the DCMR problem. These two steps are realized through f-type neurons and LP-type neurons, which we will deliberate in detail in Section 3.1.

1.3 Noise and Chaos for Combinatorial Optimization

As we use NCNNs and TCNNs in this paper, we now briefly review effects of noise and chaos on optimization. Both chaotic dynamics and noise have gained extra attention from scientists. Aihara [33] reviewed chaos engineering from various aspects such as system models, applications, and hardware implementations of chaotic NNs. Delgado-Restituto and Rodriguez-Vazquez [34] surveyed techniques of integrated chaos generators at both system and circuit levels.

Bucolo et al. [35] brought forward the question "*does chaos work better than noise?*" In their work, effects of chaos and random noise on different applications were investigated and compared. The authors concluded that "*even a general answer cannot be formulated, the benefits of chaos are often evident.*" Pavlovic et al. [36] employed stochastic noise to enhance the HNN. The experiment result on the image classification problem confirmed the expected improvement. The NCNN model has both decaying noisy and chaotic dynamics [27], [28], which helps to improve the ability of the NN model to reach global optima.

The rest of the paper is organized as follows: In Section 2, we review the formulation of the CST problem. In Section 3, we review the NCNN model and the energy function for the CST problem, followed by the presentation of NN dynamics. Simulation results and performance comparison are given in Section 4. Finally, we conclude this paper in Section 5.

2 Problem Formulation

The DCMR problem [32], or the CST problem, aims to find a tree rooted at the source s and spanning to all destinations in group D such that

1. the total cost of the tree is minimum and
2. the delay from the source to each destination is not greater than a required delay constraint.

In the formulation proposed by Pornavalai et al. [32], an n-node communication network with D destinations is formulated onto $D \times n \times n$ matrices. Matrix m is used to compute the constrained unicast route from source node s to destination m ($m = 1, \ldots, D$). Each element in a matrix is treated as a neuron, and neuron mxi (element xi in matrix m) describes the link from node x to node i for destination m in the communication network.

P_{xi} is used to characterize the connection status of the communication network:

$$P_{xi} = \begin{cases} 1, & \text{if the arc from node } x \text{ to node } i \\ & \text{does not exist,} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The routing solution is described by the final output $V_{xi}^{(m)}$ of neuron mxi as follows:

$$V_{xi}^{(m)} = \begin{cases} 1, & \text{the arc from node } x \text{ to node } i \text{ is} \\ & \text{on the final tree for destination } m, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The cost of a link from node x to node i is denoted by $C_{xi} \geq 0$. For nonexistent arcs, $C_{xi} = 0$. L_{xi} denotes a real positive delay on the link from node x to node i. We assume that link costs and link delays are independent. For example, a cost could be a measure of the amount of buffer space or channel bandwidth used, and link delays could be a combination of propagation, transmission, and queuing delays. We also assume that the routing protocol will collect state information of the communication network (e.g., the group membership, available resources, and application requirements) and deliver this information throughout the communication network.

3 The NCNN for the CST Problem

The dynamic equations for NCNNs [27], [28] are

$$\begin{aligned} U_{xi}^{(m)}(t+1) = {} & kU_{xi}^{(m)}(t) + \sum_{y=1}^{N}\sum_{j=1}^{N} w_{yj,xi} V_{yj}^{(m)}(t) \\ & + I_{xi} - z_{xi}(t)\left[V_{xi}^{(m)}(t)) - I_0\right] + n(t), \end{aligned} \tag{3}$$

$$V_{xi}^{(m)} = g_{xi}^{(m)}\left(U_{xi}^{(m)}\right) = \frac{1}{1 + e^{-U_{xi}^{(m)}/\epsilon_{xi}^{(m)}}}, \tag{4}$$

where the variables are

- $U_{xi}^{(m)}$: internal state of neuron xi in matrix m,
- $V_{xi}^{(m)}$: output of neuron xi in matrix m,
- $\epsilon_{xi}^{(m)}$: the steepness parameter of the transfer function ($\epsilon \geq 0$).
- $z_{xi}(t+1)$: $(1-\beta_1)z_{xi}(t)$,
- $A[n(t+1)]$: $(1-\beta_2)A[n(t)]$,
- k: damping factor of the nerve membrane ($0 \leq k \leq 1$),
- I_{xi}: input bias of neuron xi,
- $z_{xi}(t)$: self-feedback neuronal connection weight ($z_{xi}(t) \geq 0$),
- I_0: positive parameter,
- $n(t)$: random noise with a uniform distribution in $[-A, A]$,
- $A[n]$: noise amplitude,
- β_1, β_2: damping factors for the time-dependent neuronal self-coupling and the random noise, respectively. ($0 \leq \beta_1, \beta_2 \leq 1$).

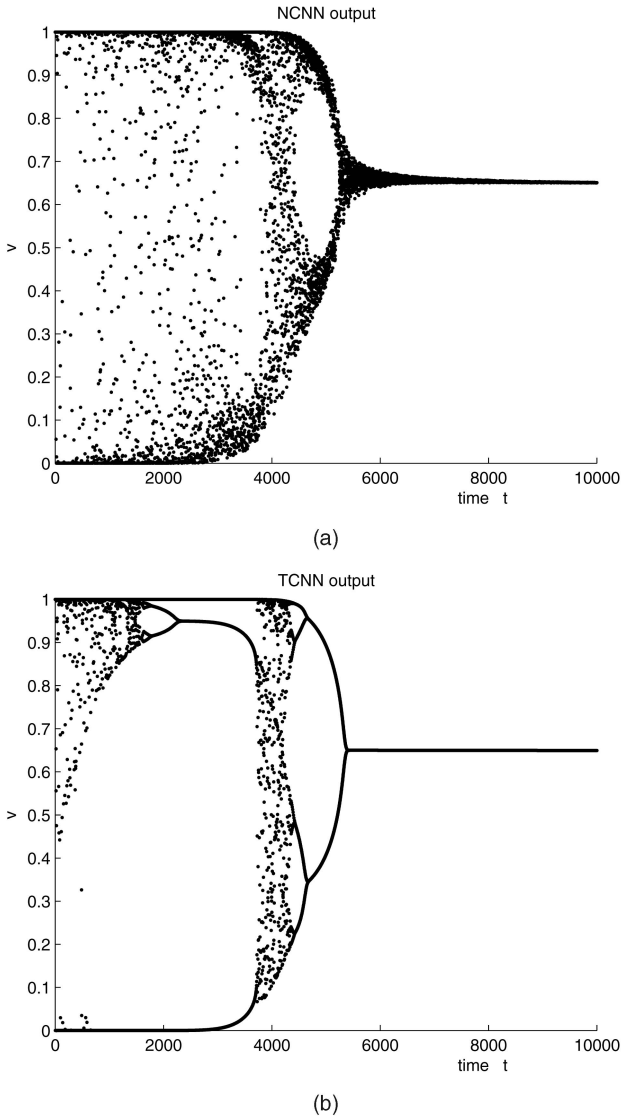


Fig. 1. A comparison of dynamics of (a) the NCNN and (b) the TCNN.

In the absence of noise, i.e., $n(t) = 0$, the NCNN reduces to the TCNN of Chen and Aihara [24]. When solving the combinatorial optimization problem through the NN method, the connection weights of the NN can be obtained from

$$\sum_{y=1}^{N}\sum_{j=1}^{N} w_{yj,xi} V_{yj}^{(m)}(t) + I_{xi} = -\frac{\partial E}{\partial V_{xi}^{(m)}}. \tag{5}$$

Here, E denotes the energy function (cost function), which depends on the problem to be solved. The minimum value of the energy function should correspond to the neuron outputs that represent an optimal solution of the application. For example, in our work, the final neuron outputs should stand for an optimal delay-constrained multicast tree.

Fig. 1 shows the dynamics of the TCNN and the NCNN, respectively. Compared with the TCNN, the NCNN explores a wider searching space because of the additional noise. Hence, the NCNN is more capable of jumping out of local minima and therefore providing better solutions.

3.1 Energy Function

Our energy function is based on the work of Pornavalai et al. [32]. In their model, the final outputs of neurons are first pushed toward zero or one through an energy term $\sum_{x=1}^{n}\sum_{i=1,i\neq x}^{n} V_{xi}^{(m)}(1 - V_{xi}^{(m)})$ and then set to zero (one) if less (greater) than a threshold 0.5. Here, we do not add this term to the energy function but rather use the average value of all the outputs as the threshold to determine the final neuron state, i.e., all outputs above the average value are set to one and zero otherwise.

The total energy function E is the sum of energy functions for the delay-constrained unicast routing to each destination:

$$E = \sum_{m\in D} E^{(m)}, \tag{6}$$

$$E^{(m)} = \mu_1 E_1^{(m)} + \mu_2 E_2^{(m)} + \mu_3 E_3^{(m)} + \mu_4 E_{4,LP}^{(m)}. \tag{7}$$

Here, $E^{(m)}$ is the energy function of matrix m, which is used to find the constrained unicast route from source node s to destination m. $\{\mu_i;\ i = 1, 2, 3, 4\}$ are the weighting coefficients.

E_1^m is the total cost of the unicast route [32]:

$$E_1^{(m)} = \sum_{x=1}^{n}\sum_{i=1,i\neq x}^{n} C_{xi} f_{xi}^{(m)}(V) V_{xi}^{(m)}, \tag{8}$$

$$f_{xi}^{(m)}(V) = \frac{1}{1 + \sum_{j=1,j\neq m,j\in D}^{n} V_{xi}^{(j)}}. \tag{9}$$

The connections among the neurons in different matrices are not directly from the outputs but through some special neurons, i.e., f-type neurons [32] that have an input-output function specified in (9). These f-type neurons help to reduce the cost term $E_1^{(m)}$ in proportion to the number of the unicast routes that are using the same link. Through the connections expressed by function $f_{xi}^{(m)}(V)$, outputs of the neurons from different matrices that represent the same link in the communication network try to cooperate together to minimize the cost of the whole multicast route.

$E_2^{(m)}$ [32] creates a virtual link from destination m to source s and requires that the number of incoming links is equal to the number of outgoing links for every node in the communication network:

$$E_2^{(m)} = \left(1 - V_{ms}^{(m)}\right) + \sum_{x=1}^{n}\left\{\sum_{i=1,i\neq x}^{n} V_{xi}^{(m)} - \sum_{i=1,i\neq x}^{n} V_{ix}^{(m)}\right\}^2. \tag{10}$$

$E_3^{(m)}$ [32] penalizes neurons that represent nonexisting links of the network:

$$E_3^{(m)} = \sum_{x=1}^{n}\sum_{i=1,i\neq x}^{n} P_{xi} V_{xi}^{(m)}. \tag{11}$$

$E_4^{(m)}$ [32] is used to satisfy the delay constraint, which is an inequality constraint:

$$\sum_{x=1}^{n}\sum_{i=1,i\neq x}^{n} L_{xi}V_{xi}^{(m)} \leq \Delta \quad V_{xi}^{(m)} \in \{0,1\}, \tag{12}$$

$$E_{4,LP}^{(m)} = \int h(z)dz,$$
$$h(z) = \begin{cases} 0, & \text{if } z \leq 0, \\ z, & \text{otherwise}, \end{cases}$$

where

$$z = \sum_{x=1}^{n}\sum_{i=1,i\neq x}^{n} L_{xi}V_{xi}^{(m)} - \Delta. \tag{13}$$

Here, Δ is the delay bound. A linear programming (LP)-type neuron [32] receives the delay of the current established route and the delay constraint as inputs. The transfer function of the LP-type neuron is denoted as $h(z)$. The LP neuron contributes positively only when the delay constraint is violated. In other words, the minimization of the energy function will force this term to approach zero, i.e., the delay of the path must not be greater than the delay constraint.

3.2 The Structure of the NN

Substituting the energy function in (3) with (7), we obtain the network dynamics, i.e., the difference equation of the NN, as follows:

$$\begin{aligned} U_{xi}^{(m)}(t+1) = {} & kU_{xi}^{(m)}(t) - \mu_1 C_{xi} f_{xi}^{(m)}(V)(1-\delta_{xm}\delta_{is}) \\ & - \mu_2\Big[-\delta_{xm}\delta_{is} + \sum_{y=1,y\neq x}^{n}\Big(V_{xy}^{(m)} - V_{yx}^{(m)}\Big) \\ & + \sum_{y=1,y\neq i}^{n}\Big(V_{iy}^{(m)} - V_{yi}^{(m)}\Big)\Big] \\ & - \mu_3 P_{xi}(1-\delta_{xm}\delta_{is}) - \mu_4 L_{xi}(1-\delta_{xm}\delta_{is})h(z) \\ & - z_i(t)\Big[V_{xi}^{(m)}(t) - I_0\Big] + n(t), \end{aligned} \tag{14}$$

where

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

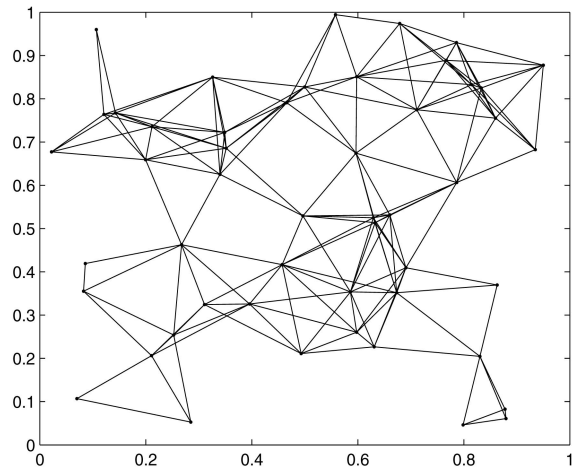
The connection strengths and the bias terms are derived through a simple comparison of (3) and (14):

$$w_{yj,xi} = -\mu_2\delta_{xy} + \mu_2\delta_{xj} - \mu_2\delta_{ij} + \mu_2\delta_{iy}, \tag{15}$$

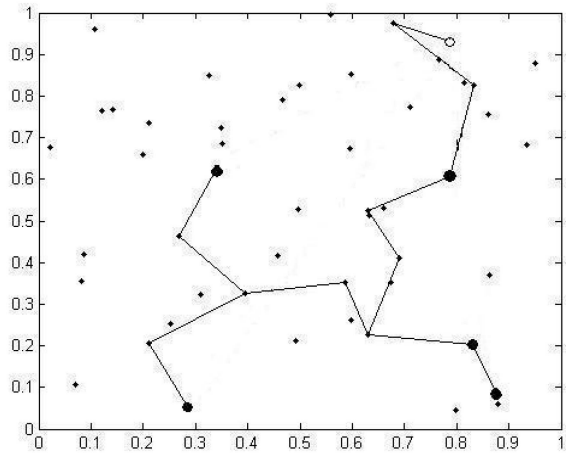
$$\begin{aligned} I_{xi} = {} & -\mu_1 C_{xi} f_{xi}^{(m)}(V)(1-\delta_{xm}\delta_{is}) + \mu_2\delta_{xm}\delta_{is} \\ & - \mu_3 P_{xi}(1-\delta_{xm}\delta_{is}) - \mu_4 L_{xi}(1-\delta_{xm}\delta_{is})h(z). \end{aligned} \tag{16}$$

Equation (16) can also be written concisely as

$$I_{xi} = \begin{cases} \mu_2, & \text{if } (x,i) = (m,s), \\ -\mu_1 C_{xi} f_{xi}^{(m)}(V) - \mu_3 P_{xi} & \\ \quad -\mu_4 L_{xi} h(z), & \text{otherwise } \forall (x \neq i). \end{cases}$$



(a)



(b)

Fig. 2. (a) A 50-node network used in our simulations, with an average degree (the number of links for a node) of four. (b) The delay-constrained optimal multicast tree found by the NCNN for the 50-node network with one source node (circle) and five destinations (black dots).

No terms in the connection matrix depend on the cost of links and the network topology. The costs are mapped onto the biases [37], so the connection matrix is independent of the changes in the flow of the communication network. The advantage is immense because one does not need to change the internal parameters of the NN to adapt to the changes in the environment. Thus, the hardware implement of the NN is a general one, independent of the topology or link costs of the communication network.

4 Simulation Results

The communication networks used in the simulation are constructed randomly with a graph generator (based on the method proposed by Waxman [38]). A communication network with n nodes is randomly placed on a Cartesian coordinate. The cost and delay of an arc from node x to node i, i.e., C_{xi} and L_{xi}, are randomly generated. Fig. 2a presents an example of a randomly generated 50-node network. Fig. 2b shows the delay-constrained optimal multicast tree found by the NCNN for the 50-node network.

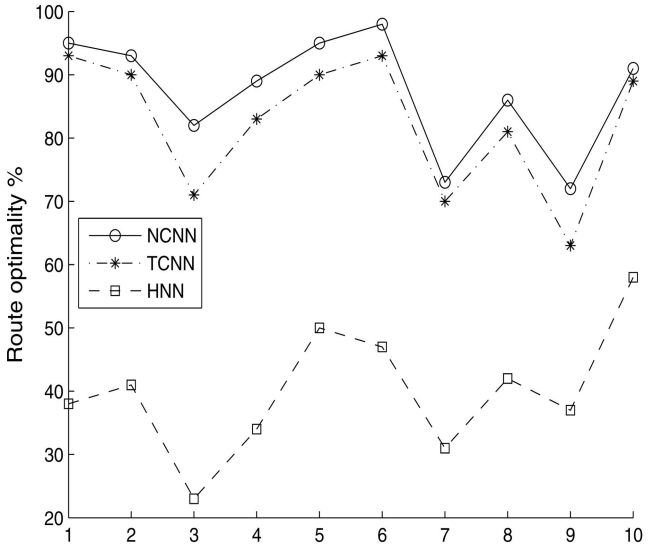


Fig. 3. Comparisons of the performance of NCNNs, TCNNs, and HNNs: the horizontal axis is the instance number. The route optimality denotes the percentage at which the network reached the optimal solution in the 1,000 runs.

The source node is shown as a circle, and the destination nodes are shown as dark dots.

The algorithm is implemented in VC++ and run on a Linux cluster (16-node dual Xeon 3.06-GHz, Intel IA32). Values for the coefficients in our model are chosen as in [28] to have rich dynamics and be able to converge in 8,000 steps: $\epsilon_{xi} = \epsilon = 0.004$, independent of neuron location xi. $k = 0.9999$, and $I_0 = 0.65$. Initial internal states $U_{xi}(0)$ are randomly generated between $[-1, 1]$. The algorithm is stopped when the value of the energy function does not change by more than a threshold value (0.0002) in three consecutive updates. The final neuron outputs are decoded to the multicast tree through the prior definition in (2). We assume that the average value of the neuron outputs is V_{Avg}; if $V_{xi} \geq V_{Avg}$, then $V_{xi} = 1$, i.e., the neuron is on, which means that the link from node x to node i in the communication network is chosen to be in the final optimal tree, and vice versa.

The initial self feedback $z_{xi}(0)$ [24], [28] and the initial amplitude of random noise $A[n(0)]$ [28] are set properly to keep the balance of every term in (3). β_1 determines the length of transient chaos [24], and β_2 determines the cooling schedules of noise [28]. We set $z_{xi}(0) = z(0) = 0.1$, $A[n(0)] = 0.02$, and $\beta_1 = \beta_2 = 0.0001$ in our simulations, unless otherwise specified.

We set the weighting coefficients in the energy function based on the principles described in [32] as follows:

$$\mu_1 = 150; \quad \mu_2 = 5{,}000; \quad \mu_3 = 5{,}000; \quad \mu_4 = 200.$$

For different instances, e.g., different network sizes, destination numbers, or delay bounds, these network parameters (actually only μ_1 and μ_4) are tuned through trial and error on a small scale. To improve the solution, we need to slightly reduce μ_1 if the cost of the final tree is large, and we need to increase μ_4 when the delay bound is strict. We will discuss the effects of various noise levels $A[n(0)]$ and different weighting constants μ_1 and μ_4, while fixing μ_2 and μ_3 at 5,000, later in this section. The parameters μ_2 and μ_3 are instance independent. The μ_2 term promotes completeness of the route, and the μ_3 term prevents the use of nonexisting links. To prevent infeasible solutions, we set μ_2 and μ_3 at 5,000, which is much greater than μ_1 and μ_4.

TABLE 1
Specifications of the Randomly Generated Geometric Instances

Instance	Nodes Number N	Destinations Number	Edges E	Delay bound Δ
Case #1	8	5	11	20
Case #2	16	5	22	20
Case #3	30	5	47	20
Case #4	80	5	254	25
Case #5	100	5	352	25
Case #6	50	5	171	20
Case #7	50	15		
Case #8	50	25		
Case #9	50	35		
Case #10	50	45		
Case #11	30	10	47	25
Case #12	80	50	254	25

The proposed algorithm is run 1,000 times on a randomly generated topology and randomly assigned link parameters for each instance (1-12). In all the cases, algorithm converged to stable states within 4,000-6,000 iterations. The statistical results are shown in Table 2.

The performance of the NCNN is compared with that of the TCNN and HNN (Fig. 3). From the HNN to the TCNN and from the TCNN to the NCNN, the network dynamics become more and more complex, and as a result, the model can reach the global optimal solution more frequently. However, the searching time increases. In a real application, one may choose the model, as well as the complexity of the dynamic, depending on the preference of the client.

We investigate the performance of NCNNs with different settings of noise levels $A[n(0)]$ and the weighting constants μ_1 and μ_4. Results are shown in the Tables 3, 4, and 5. The *Local minima rate* shows the percentage of simulations in which the NN finds solutions within the delay constraint but the solutions are not global optima. The *Infeasible rate* means the percentage of the simulations in which the solutions do not satisfy the delay constraint or the algorithm does not converge. In the previous tables and figures in this paper, the weighting coefficients are properly set, and the "Infeasible rates" are zero.

As the noise level goes high, the computation time increases, the chance to reach the global optima is also enhanced, and there are more infeasible solutions if the

TABLE 2
Results for the HNN, TCNN, and NCNN for Instances 1-12

Instance No.	Cost mean±sd			Time mean±sd (s)		
	HNN	TCNN	NCNN	HNN	TCNN	NCNN
1	12.19±0.53	8.72±0.31	8.10±0.44	0.18±0.22	0.68±0.29	0.71±0.45
2	32.06±3.23	23.20±0.28	23.03±0.79	2.17±0.64	6.49±0.70	6.95±0.97
3	24.35±1.31	22.67±1.95	20.29±1.41	23.32±3.71	46.86±4.18	53.64±5.07
4	29.07±1.02	25.13±0.98	23.27±0.73	537.3±79.2	1150±127	1284±133
5	31.25±0.18	31.20±0.21	31.05±0.22	1788±253	2505±289	2641±350
6	20.85±1.42	16.02±0.28	16.02±0.27	205.2±42.8	292.3±37.2	323.5±46.7
7	31.62±1.73	29.01±0.89	26.66±0.75	622.8±82.1	1038±63.3	1152±78.4
8	39.35±0.76	37.69±0.69	36.55±0.52	1582±134	2030±210	2161±183
9	62.28±3.40	50.35±0.87	48.65±0.93	2034±384	3238±265	3482±210
10	71.03±2.58	56.35±1.80	53.95±0.21	2980±291	3870±380	4170±414
11	26.86±1.55	25.74±0.95	24.81±0.39	58.4 ±8.7	95.1 ±12.8	134.6±19.5
12	41.2±0.25	39.0±0.56	38.6±0.54	4590±284	5011 ±327	5749±382

"sd" stands for "standard deviation."

TABLE 3
Performance Comparison of the NCNN Models with Different Noise Levels for the 30-Node Network

Noise level $A[n(0)]$	0.02	0.05	0.1	0.2	0.4
Cost mean±sd	21.65±2.49	20.92±1.76	20.29±1.41	20.21±1.34	20.14±0.97
Time mean±sd (s)	42.26±9.29	45.19±5.68	41.13±4.82	48.31±11.7	52.67±13.8
Global minima rate %	82.1	84.2	89.2	91.6	92.2
Local minima rate %	17.9	15.8	10.8	7.7	3.6
Infeasible rate %	0	0	0	0.7	4.2

$\mu_1 = 150$ *and* $\mu_4 = 200$. *"sd" stands for "standard deviation."*

TABLE 4
Performance Comparison of the NCNN Models with Different Weighting Constants μ_1 for the 30-Node Network

μ_1	125	150	175	200
Cost mean±sd	23.79±3.41	21.65±2.49	20.58±1.60	24.59±1.39
Time mean±sd (s)	43.94±4.52	42.26±9.29	44.01±18.4	44.30±19.7
Global minima rate %	80.5	82.1	81.9	53.5
Local minima rate %	13.3	17.9	5.9	24.8
Infeasible rate %	6.2	0	12.2	21.7

$\mu_4 = 200$ *and* $A[n(0)] = 0.02$. *"sd" stands for "standard deviation."*

additive noise is too much. The tuning of weighting coefficients in the energy function is an important issue related to the efficiency when solving optimization problems with NNs. The NN is more capable to reach the global optima when the weighting coefficient for the cost term, i.e., μ_1, is larger. However, if μ_1 is too large, the solution may not satisfy the delay constraint, or the NN does not even converge.

To compare the chaotic NN with the BSMA, we run both of them on instances 1-10 in Table 1. Since the BSMA is a deterministic heuristic algorithm, it will reach the same solution every time. We compare this result

TABLE 5
Performance Comparison of the NCNN Models with Different Weighting Constants μ_4 for the 30-Node Network

μ_4	150	200	250	300	350	400
Cost mean±sd	21.70±2.59	21.65±2.49	21.61±2.54	21.56±2.49	21.69±2.34	21.73±2.28
Time mean±sd (s)	41.41±8.74	42.26±9.29	42.29±4.36	42.45±10.1	42.96±10.9	42.08±4.94
Global minima rate %	81.3	82.1	82.4	82.4	83.2	81.7
Local minima rate %	18.7	17.9	17.1	14.9	14.6	16.1
Infeasible rate %	0	0	0.5	2.7	2.2	2.2

$\mu_1 = 150$ *and* $A[n(0)] = 0.02$. *"sd" stands for "standard deviation."*

with the optimal route cost achieved by TCNNs and NCNNs. The comparison is shown in the histogram in Fig. 4. Though the chaotic NN cannot promise to reach the global optima in every run, it indeed can find a better solution (with a smaller route cost) compared with the BSMA.

The COPT algorithm [19], which is implemented by the branch-and-bound technique, can be applied to only small communication networks because of the long computation time. The optimal results from the NCNN and TCNN on instances 1-3 are the same with those obtained by the COPT algorithm.

5 Conclusion

In this paper, we review previous work on the DCMR problem and its formulation as an energy minimization problem that is solvable by NNs. Equations governing the dynamics of NNs have been discussed. The DCMR problem in communications desires 1) satisfaction of the delay constraint on the paths from the source to each destination and 2) minimum cost of the multicast tree. In the cases that we have studied, the NCNN finds global optima more frequently compared to the TCNN and the original HNN model. The cost of the multicast tree found by the chaotic NNs is equal to if not less than the one obtained by the BSMA. Results from the COPT algorithm showed that chaotic NNs, i.e., TCNNs and NCNNs, are able to find the global optima.

In the future, we will focus on the improvement of the NCNN and the application on many-to-many multicast routing problems or dynamic routing. To utilize the parallel processing ability of NNs, we will also explore the hardware implementation of the NCNN. The NN model is also applicable to the communication networks with asymmetric link characteristics or with variable delay constraints for different destinations. We will also look into these issues.

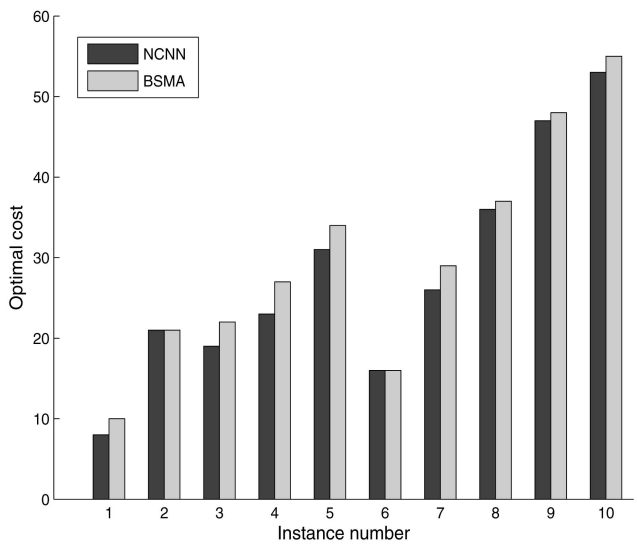


Fig. 4. Comparisons of the optimal route costs found by the chaotic NNs (NCNNs and TCNNs) and the BSMA for 10 instances. The numbers 1-10 in the horizontal axis stand for instances 1-10 in Table 1, respectively.

References

[1] R.-S. Chang and C.-D. Wang, "Improved WWW Multimedia Transmission Performance in HTTP/TCP over ATM Networks," *IEEE Trans. Multimedia,* vol. 1, no. 3, pp. 278-290, Sept. 1999.
[2] T. Korkmaz and M. Krunz, "Routing Multimedia Traffic with QoS Guarantees," *IEEE Trans. Multimedia,* vol. 5, no. 3, pp. 429-443, Sept. 2003.
[3] A. Ganjam and H. Zhang, "Internet Multicast Video Delivery," *Proc. IEEE,* vol. 93, no. 1, pp. 159-170, 2005.
[4] K.M.F. Elsayed, "A Framework for End-to-End Deterministic-Delay Service Provisioning in Multiservice Packet Networks," *IEEE Trans. Multimedia,* vol. 7, no. 3, pp. 563-571, 2005.
[5] B. Wang and C.J. Hou, "A Survey on Multicast Routing and Its QoS Extension: Problems, Algorithms, and Protocols," *IEEE Network Magazine,* vol. 14, no. 1, pp. 22-36, 2000.
[6] M. Charikar, J.S. Naor, and B. Schieber, "Resource Optimization in QoS Multicast Routing of Real-Time Multimedia," *IEEE Trans. Networking,* vol. 12, no. 2, pp. 340-348, Apr. 2004.
[7] V. Sarangan, D. Ghosh, and R. Acharya, "Capacity-Aware State Aggregation for Interdomain QoS Routing," *IEEE Trans. Multimedia,* vol. 8, no. 4, pp. 792-808, Aug. 2006.
[8] D. Ghosh, V. Sarangan, and R. Acharya, "Quality-of-Service Routing in IP Networks," *IEEE Trans. Multimedia,* vol. 3, no. 2, pp. 200-208, June 2001.
[9] C.P. Low and X.Y. Song, "On Finding Feasible Solutions for the Delay Constrained Group Multicast Routing Problem," *IEEE Trans. Computers,* vol. 51, no. 5, pp. 581-588, May 2002.
[10] H.Y. Tyan, J.C. Hou, and B. Wang, "Many to Many Multicast Routing with Temporal Quality of Service Guarantees," *IEEE Trans. Computers,* vol. 52, no. 6, pp. 826-832, June 2003.
[11] D. Chakraborty, G. Chakraborty, and N. Shiratori, "A Dynamic Multicast Routing Satisfying Multiple QoS Constraints," *Int'l J. Network Management,* vol. 13, no. 5, pp. 321-335, 2003.
[12] V.P. Kompella, J.C. Pasquale, and G.C. Polyzos, "Multicast Routing for Multimedia Communication," *IEEE/ACM Trans. Networking,* vol. 1, no. 3, pp. 286-292, June 1993.
[13] H.F. Salama, D.S. Reeves, and Y. Viniotis, "Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks," *IEEE J. Selected Areas in Comm.,* vol. 15, no. 3, pp. 332-345, 1997.

[14] R. Widyono, "The Design and Evaluation of Routing Algorithms for Real-Time Channels," Technical Report 94-024, Univ. of California, Berkeley, and Int'l Computer Science Inst., June 1994.
[15] Q. Sun and H. Langendoerfer, "Efficient Multicast Routing for Delay-Sensitive Applications," *Proc. Second Workshop Protocols Multimedia Systems (PROMS '95),* pp. 452-458, 1995.
[16] Q. Zhu, M. Parsa, and J.J. Garcia-Luna-Aceves, "A Source-Based Algorithm for Near-Optimum Delay-Constrained Multicasting," *Proc. IEEE INFOCOM '95,* pp. 377-385, 1995.
[17] M. Parsa, Q. Zhu, and J.J. Garcia-Luna-Aceves, "An Iterative Algorithm for Delay-Constrained Minimum-Cost Multicasting," *IEEE Trans. Networking,* vol. 6, no. 4, pp. 461-474, 1998.
[18] J.Y. Yen, "Finding the K Shortest Loopless Paths in a Network," *Management Science,* vol. 17, no. 11, pp. 712-716, 1971.
[19] P. Manyem, "Routing Problems in Multicast Network," PhD dissertation, North Carolina State Univ., Raleigh, chapter 3, pp. 28-39, 1996.
[20] H.E. Rauch and T. Winarske, "Neural Networks for Routing Communication Traffic," *IEEE Control Systems Magazine,* vol. 8, no. 2, pp. 26-31, 1988.
[21] J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proc. Nat'l Academy of Sciences,* vol. 79, pp. 2554-2558, 1982.
[22] A.H. Gee and R.W. Prager, "Limitations of Neural Networks for Solving Traveling Salesman Problems," *IEEE Trans. Neural Networks,* vol. 6, no. 1, pp. 280-282, 1995.
[23] H. Nozawa, "A Neural-Network Model as a Globally Coupled Map and Applications Based on Chaos," *Chaos,* vol. 2, no. 3, pp. 377-386, 1992.
[24] L.N. Chen and K. Aihara, "Chaotic Simulated Annealing by a Neural Network Model with Transient Chaos," *Neural Networks,* vol. 8, no. 6, pp. 915-930, 1995.
[25] Z. Ding, H. Leung, and Z. Zhu, "A Study of the Transiently Chaotic Neural Network for Combinatorial Optimization," *Math. and Computer Modelling,* vol. 36, no. 9, pp. 1007-1020, 2002.
[26] X. Xu, Z. Tang, and J. Wang, "A Method to Improve the Transiently Chaotic Neural Network," *Neurocomputing,* vol. 67, pp. 456-463, Mar. 2005.
[27] L.P. Wang and F.Y. Tian, "Noisy Chaotic Neural Networks for Solving Combinatorial Optimization Problems," *Proc. Int'l Joint Conf. Neural Networks (IJCNN '00),* vol. 4, pp. 37-40, July 2000.
[28] L.P. Wang, S. Li, F.Y. Tian, and X.J. Fu, "A Noisy Chaotic Neural Network for Solving Combinatorial Optimization Problems: Stochastic Chaotic Simulated Annealing," *IEEE Trans. Systems, Man, and Cybernetics—Part B: Cybernetics,* vol. 34, no. 5, pp. 2119-2125, 2004.
[29] M.K.M. Ali and F. Kamoun, "Neural Networks for Shortest Path Computation and Routing in Computer Networks," *IEEE Trans. Neural Networks,* vol. 4, no. 6, pp. 941-954, 1993.
[30] D.C. Park and S.E. Choi, "A Neural Network Based Multi-Destination Routing Algorithm for Communication Network," *Proc. IEEE Int'l Joint Conf. Neural Networks (IJCNN '98),* pp. 1673-1678, 1998.
[31] C.W. Ahn, R.S. Ramakrishna, C.G. Kang, and I.C. Choi, "Shortest Path Routing Algorithm Using Hopfield Neural Network," *Electronics Letters,* vol. 37, no. 19, pp. 1176-1178, 2001.
[32] C. Pornavalai, G. Chakraborty, and N. Shiratori, "A Neural Network Approach to Multicast Routing in Real-Time Communication Networks," *Proc. Int'l Conf. Network Protocols (ICNP '95),* pp. 332-339, 1995.
[33] K. Aihara, "Chaos Engineering and Its Application to Parallel Distributed Processing with Chaotic Neural Networks," *Proc. IEEE,* vol. 90, no. 5, pp. 919-930, 2002.
[34] M. Delgado-Restituto and A. Rodriguez-Vazquez, "Integrated Chaos Generators," *Proc. IEEE,* vol. 90, no. 5, pp. 747-767, 2002.
[35] M. Bucolo, R. Caponetto, L. Fortuna, M. Frasca, and A. Rizzo, "Does Chaos Work Better Than Noise?" *IEEE Circuits and Systems Magazine,* vol. 2, no. 3, pp. 4-19, 2002.
[36] V. Pavlovic, D. Schonfeld, and G. Friedman, "Stochastic Noise Process Enhancement of Hopfield Neural Networks," *IEEE Trans. Circuits and Systems II,* vol. 52, no. 4, pp. 213-217, 2005.
[37] P. Venkataram, S. Ghosal, and B.P.V. Kumar, "Neural Network Based Optimal Routing Algorithm for Communication Networks," *Neural Networks,* vol. 15, no. 10, pp. 1289-1298, 2002.
[38] B. Waxman, "Routing of Multipoint Connections," *IEEE J. Selected Areas in Comm.,* vol. 6, no. 9, pp. 1617-1622, 1988.

Lipo Wang received the BS degree from the National University of Defense Technology, China, in 1983 and the PhD degree from Louisiana State University, Baton Rouge, in 1988. He is currently with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interest is computational intelligence, with applications to data mining, bioinformatics, and optimization. He has published more than 170 papers in journals and conference proceedings. He holds a US patent in neural networks. He has authored two monographs and edited 16 books. He was/will be a keynote/plenary/panel speaker for several international conferences. He has been an associate editor for the *IEEE Transactions on Neural Networks* since 2002, *IEEE Transactions on Evolutionary Computation* since 2003, and *IEEE Transactions on Knowledge and Data Engineering* since 2005. He has been an area editor of the *Soft Computing* journal since 2002. He serves on the editorial board of six additional international journals and was on the editorial board of three other journals. He was the vice president for technical activities of the IEEE Computational Intelligence Society (2006-2007) and served as the chair of the Emergent Technologies Technical Committee (2004-2005). He has been on the governing board of the Asia-Pacific Neural Network Assembly since 1999 and served as its president in 2002/2003. He was the founding chair of both the IEEE Engineering in Medicine and Biology Chapter Singapore and IEEE Computational Intelligence Chapter Singapore. He serves/served as the General/Program Chair for 11 international conferences and as a member of the steering/advisory/organizing/program committees of more than 120 international conferences. He is a senior member of the IEEE.

Wen Liu received the BS degree in electronic engineering from Peking University, Beijing, in 2004. She is currently working toward the PhD degree in the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. Her current research interests include the cellular neural network and combinatorial optimization using neural networks. She is a student member of the IEEE.

Haixiang Shi received the BS degree in electronic engineering from Xidian University, Xian, China, in 1998 and the PhD degree from Nanyang Technological University, Singapore, in 2007. He is currently with the School of Electrical and Electronic Engineering, Nanyang Technological University. His research interests are in combinatorial optimization using computational intelligence, with applications to routing, assignment, and scheduling problems in wireless ad hoc and sensor networks.