A Gradual Noisy Chaotic Neural Network for Solving the Broadcast Scheduling Problem in Packet Radio Networks

Lipo Wang, *Senior Member, IEEE*, and Haixiang Shi

***Abstract*—In this paper, we propose a gradual noisy chaotic neural network (G-NCNN) to solve the NP-complete broadcast scheduling problem (BSP) in packet radio networks. The objective of the BSP is to design an optimal time-division multiple-access (TDMA) frame structure with minimal TDMA frame length and maximal channel utilization. A two-phase optimization is adopted to achieve the two objectives with two different energy functions, so that the G-NCNN not only finds the minimum TDMA frame length but also maximizes the total node transmissions. In the first phase, we propose a G-NCNN which combines the noisy chaotic neural network (NCNN) and the gradual expansion scheme to find a minimal TDMA frame length. In the second phase, the NCNN is used to find maximal node transmissions in the TDMA frame obtained in the first phase. The performance is evaluated through several benchmark examples and 600 randomly generated instances. The results show that the G-NCNN outperforms previous approaches, such as mean field annealing, a hybrid Hopfield network-genetic algorithm, the sequential vertex coloring algorithm, and the gradual neural network.**

***Index Terms*—Broadcast scheduling problem, gradual noisy chaotic neural network, NP-complete, packet radio network.**

I. Introduction

PACKET radio networks provide a good option for high-speed wireless data communications, especially over a broad geographic region [1]. Packet radio networks (PRNs) consist of geographically distributed nodes and provides flexible data communication services for nodes through a shared high-speed radio channel by broadcasting. Each node is equipped with a transmitter and a receiver with limited power. Therefore the transmission range is limited and only the nodes within a certain range can transmit messages to or receive messages from each other. If two nodes are far apart, packets need to be relayed by intermediate nodes and traverse several hops before reaching the destination.

The time-division multiple-access (TDMA) protocol has been adopted in PRNs for nodes to communicate with each other in a single shared radio channel. Since all nodes share one radio channel, conflicts may occur with uncontrolled transmissions, resulting in damaged packets at the destination [7]. These damaged packets increase network delays because they must be retransmitted. Hence effective broadcast scheduling is necessary to avoid any conflict and to use the limited channel bandwidth efficiently. In a TDMA network, time is divided into frames and each TDMA frame is a collection of time slots. A time slot has a unit time length required for a single packet to be communicated between adjacent nodes. When nodes transmit simultaneously, conflicts will occur if the nodes are in a close range. Therefore, adjacent nodes must be scheduled to transmit in different time slots, while nodes some distance away may be arranged to transmit in the same time slot without causing conflict [4]. The goal of the broadcast scheduling problem (BSP) is to find an optimal TDMA frame structure that fulfills the following two objectives. The first is to schedule transmissions of all nodes in a minimal TDMA frame length without any conflict. The second is to maximize channel utilization or total conflict-free transmissions.

The BSP has been proven to be an NP-complete combinatorial optimization problem [2], [4] and has been studied in the literature [3]–[8]. Most of the earlier algorithms for the BSP assume that the frame length is fixed and is known a priori [2], [3]. Recent algorithms [4]–[8] aim at finding both the minimal frame length and the maximum conflict-free transmission, i.e., now there are two objectives in the BSP. Usually, two stages or phases are adopted to tackle the two objectives in a separate fashion. The minimal frame length is achieved in the first stage and conflict-free node transmissions are maximized in the second stage [6]–[8].

Specifically, Ephremides and Truong [2] proposed a distributed greedy algorithm to find a TDMA structure with maximal transmission. They proved that searching for the optimal scheduling of broadcasts in a radio network is NP-complete. Funabiki and Takefuji [3] proposed a parallel algorithm based on an artificial neural network to solve the M-slot problem where the number of time slots is predefined. They used hill-climbing to help the system escape from local minima. Wang and Ansari [4] proposed a mean field annealing (MFA) algorithm to find a TDMA cycle with minimum delay time. In order to find the minimal frame length, they first used MFA to find assignments for all nodes with a lower bound of the frame length. For the unassigned nodes left, they then used another heuristic algorithm, which adds one time slot at each iteration to the node with the highest degree. The heuristic algorithm was run repeatedly until all the nodes left were assigned. After the number of time slots was found, additional feasible assignments to the nodes were attempted. They also proved NP-completeness of the BSP by transforming the BSP to the maximum independent set problem.

Chakraborty and Hirano [5] used genetic algorithms with a modified crossover operator to handle large networks with complex connectivity. Funabiki and Kitamichi [6] proposed a bi-

Manuscript received July 7, 2004; revised March 9, 2005.

The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore (e-mail: ELPWang@ntu.edu.sg).

Digital Object Identifier 10.1109/TNN.2006.875976

nary neural network with a gradual expansion scheme, called a gradual neural network (GNN), to find the minimum frame length and the maximum transmissions through a two-phase process. In phase I, they found a valid TDMA cycle to satisfy the two constraints with the minimum number of time slots. The number of time slots in a TDMA cycle was gradually increased at every P iterations from an initial value during the iterative computation of the neural network, until every node can transmit at least once in the cycle without conflicts, where P is a predefined parameter. In phase II, additional conflict-free transmissions for the TDMA cycle of phase I were found in order to maximize the total number of transmissions. They demonstrated the performance of their method through the three benchmark instances used in [4] and randomly generated geometric graph instances.

Yeo *et al.* [7] proposed a two-phase algorithm based on sequential vertex coloring. They showed that their method can find better solutions compared to the method in [4]. Salcedo-Sanz *et al.* [8] proposed a hybrid algorithm which combines a Hopfield neural network for constrain satisfaction and a genetic algorithm for achieving maximal throughput. They partitioned the BSP into two subproblems, i.e., the problem of maximizing the throughput of the system (P1) and the problem to find a feasible frame with one and only one transmission per radio station (P2). Accordingly they used a hybrid two-stage algorithm in solving these two subproblems, i.e., the Hopfield neural network for P2 and a combination of genetic algorithms and the Hopfield neural network for P1. They compared their results with MFA in [4] in the three benchmark problems and showed that the hybrid algorithm outperformed MFA.

In this paper, we first introduce a novel neural network model with complex neurodynamics, i.e., the noisy chaotic neural network (NCNN). We then apply the NCNN to solve the BSP in two phases. In the first phase, a gradual noisy chaotic neural network (G-NCNN), which combines the NCNN and the gradual expansion scheme, is proposed to obtain the minimal TDMA frame length. In the second phase, the NCNN is used to obtain the maximal number of node transmissions. Numerical results show that our NCNN outperforms the existing algorithms in both the average delay time and the minimal TDMA cycle length. The organization of this paper is as follows. The next section reviews the BSP in packet radio networks, and we then present a formulation of the BSP. In Section III, we first describe the NCNN model. In Section IV, we apply the G-NCNN to the BSP. The performance is evaluated in Section V. Section VI concludes this paper.

II. The Broadcast Scheduling Problem

In this section, we first briefly describe the BSP, as in [3]–[8], and we then present a formulation of the BSP based on the description. A PRN can be represented by a graph $G = (V, E)$, where vertices in $V = \{1, \ldots, N\}$ are network nodes, N being the total number of nodes in the PRN, and E represents the set of transmission links. Two nodes i and j $(i, j \in V)$ are connected by an undirected edge $e_{ij} \in E$ if and only if they can receive each other's transmission [7]. In such a case, the two nodes i and j are *one hop* away. If $e_{ij} \notin E$, but there is an intermediate node k such that $e_{ik} \in E$ and $e_{kj} \in E$, then nodes i and j are *two hops* away. A *primary conflict* occurs when two nodes that are one hop away transmit in the same time slot, as shown in Fig. 1(a). A *secondary conflict* occurs if two nodes are two hops away and transmit in the same time slot, as shown in Fig. 1(b). We summarize the constraints in the BSP in the following two categories.

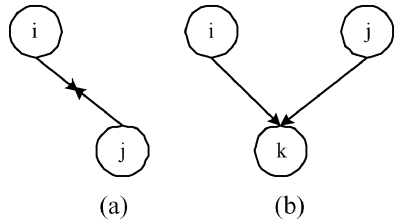


Fig. 1. Situations in which conflicts occur in a packet radio network. (a) Primary conflict. Node i cannot have transmission and reception simultaneously. (b) Secondary conflict. Node k is not allowed to receive two or more transmissions simultaneously.

1) *No-transmission constraint [5]:* Each node should be scheduled to transmit at least once in a TDMA cycle.
2) *No-conflict constraint:* It excludes the primary conflict (a node cannot have transmission and reception simultaneously) and the secondary conflict (a node is not allowed to receive more than one transmission simultaneously).

From the above constraints, we can see that two nodes can transmit in the same time slot without conflicts if and only if they are more than two hops away from each other.

The topology of a TDMA network can be represented by an $N \times N$ symmetric binary matrix $C = \{c_{ij}\}(i, j = 1, \ldots, N)$, called the connectivity matrix

$$c_{ij} = \begin{cases} 1, & \text{if there is a link between node } i \\ & \text{and } j,\ i \neq j \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

An $N \times N$ matrix called the compatibility matrix $\mathrm{D} = \{d_{ij}\}$ [6] is defined as follows:

$$d_{ij} = \begin{cases} 1, & \text{if node } i \text{ and } j \text{ are within} \\ & \text{two-hop distance} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The final optimal solution obtained is a transmission schedule consisting of M time slots. We use an $M \times N$ binary matrix $T = (t_{ij})$ to express a transmission schedule [4], where

$$t_{ij} = \begin{cases} 1, & \text{if time slot } i \text{ in a frame is} \\ & \text{assigned to node } j \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Channel utilization ρ_j for node j is defined as [4]

$$\rho_j = \frac{\text{the number of slots assigned to node } j}{\text{TDMA cycle length}} = \frac{\sum_{i=1}^{M} t_{ij}}{M}.$$

The total channel utilization for the entire network ρ is given by [4]

$$\rho = \frac{1}{N}\sum_{j=1}^{N}\rho_j = \frac{1}{NM}\sum_{j=1}^{N}\sum_{i=1}^{M} t_{ij}. \tag{4}$$

The goal of the BSP is to find a transmission schedule with the shortest TDMA frame length (i.e., smallest M) which satisfies the above constraints, and at the same time, the total transmissions are maximized. Based on the above description of the BSP, we formulate the BSP as follows:

$$\textbf{BSP} \quad \text{minimize } M \text{ and maximize } \rho$$

$$\text{Subject to :} \quad \sum_{i=1}^{M} t_{ij} > 0, \quad \text{for all } j = 1, 2, \ldots, N \tag{5}$$

$$\sum_{i=1}^{M} \sum_{j=1}^{N} \sum_{k=1, k\neq j}^{N} d_{jk} t_{ij} t_{ik} = 0. \tag{6}$$

The no-transmission constraint is formulated in (5), which means each node in the network must transmit at least once in a frame. The no-conflict constraint in (6) indicates that every pair of nodes within one hop or two hops away cannot be scheduled in the same time slot.

A trivial solution satisfying all the above two constraints is an N-slot TDMA cycle where every node transmits in a different time slot. But obviously this solution is not optimal. The BSP is an NP-complete combinatorial optimization problem, and to find a global optimum is not easy. In the next section, we will introduce a novel noisy chaotic neural network and then apply this model to solve the BSP in the subsequent sections.

III. The Noisy Chaotic Neural Network

A. Model Definition

There have been extensive research interests in theory and applications of Hopfield-based type neural networks [24]–[27]. Since the original Hopfield neural network (HNN) [9], [10] can be easily trapped in local minima, stochastic simulated annealing (SSA) [11] has been combined with the HNN [12]. Besides, chaotic neural networks [13]–[18] have also attracted much attention because chaotic neural networks have a richer spectrum of dynamic behaviors, such as stable fixed points, periodic oscillations, and chaos, in comparison with static neural network models. Nozawa demonstrated the search ability of chaotic neural networks [13], [14]. Chen and Aihara [15], [16] proposed chaotic simulated annealing (CSA) by starting with a sufficiently large negative self-coupling in the neurons and then gradually decreasing the self-coupling to stabilize the network. They called this model the transiently chaotic neural network (TCNN). Because the TCNN restricts the random search to a subspace of the chaotic attracting set, which is much smaller than the entire state space, it can search more efficiently [17].

SSA is known to relax to a global minimum with probability one if the annealing takes place sufficiently slowly, i.e., at least inversely proportional to the logarithm of time [19]. In a practical term, this means that SSA is capable of producing good (optimal or near optimal) solutions for many applications if the annealing parameter (temperature) is reduced exponentially with a reasonably small exponent. However, unlike SSA, CSA has completely deterministic dynamics and is not guaranteed to settle down to at a global minimum no matter how slowly the annealing parameter (the self-coupling) is reduced [18]. Practically speaking, this implies that CSA sometimes may not be able to provide a good solution at the conclusion of annealing even after a long time of searching.

By adding decaying stochastic noise into the TCNN, Wang and Tian [20]–[22] proposed a new approach to simulated annealing, i.e., stochastic chaotic simulated annealing (SCSA), using a noisy chaotic neural network (NCNN). Compared with CSA, SCSA performs stochastic searching both before and after chaos disappears and is more likely to find optimal or suboptimal solutions. This novel method has been applied successfully to solving several challenging optimization problems, including the traveling salesman problem (TSP) and the channel assignment problem (CAP) [20]–[22]. In [22], the NCNN performed as well as the TCNN in small-size TSPs, such as ten-city and 21-city TSPs. But when used on larger size TSPs, such as 52-city and 70-city TSPs, the NCNN achieved a better performance compared to the TCNN. In the CAP, the NCNN obtained smaller overall interference (by 2% to 7.3%) compared to the TCNN in all benchmark instances.

The NCNN model is described as follows [20]:

$$x_{jk}(t) = \frac{1}{1+e^{-y_{jk}(t)/\varepsilon}} \tag{7}$$

$$y_{jk}(t+1) = k y_{jk}(t) - z(t)\,[x_{jk}(t) - I_0] + n(t) + \alpha \left\{ \sum_{i=1, i\neq j}^{N} \sum_{l=1, l\neq k}^{M} w_{jkil} x_{jk}(t) + I_{jk} \right\} \tag{8}$$

$$z(t+1) = (1-\beta_1) z(t) \tag{9}$$

$$n(t+1) = (1-\beta_2) n(t) \tag{10}$$

where the notations are:

- x_{jk} output of neuron jk;
- y_{jk} input of neuron jk;
- w_{jkil} connection weight from neuron jk to neuron il, with $w_{jkil} = w_{iljk}$ and $w_{jkjk} = 0$

$$\sum_{i=1, i\neq j}^{N} \sum_{l=1, l\neq k}^{M} w_{jkil} x_{jk} + I_{jk} = -\frac{\partial E}{\partial x_{jk}} \tag{11}$$

- I_{jk} input bias of neuron jk;
- k damping factor of nerve membrane ($0 \leq k \leq 1$);
- α positive scaling parameter for inputs;
- β_1 damping factor for neuronal self-coupling ($0 \leq \beta_1 \leq 1$);
- β_2 damping factor for stochastic noise ($0 \leq \beta_2 \leq 1$);
- $z(t)$ self-feedback connection weight or refractory strength ($z(t) \geq 0$);
- I_0 positive parameter;
- ε steepness parameter of the output function ($\varepsilon > 0$);
- E energy function;

$n(t)$ random noise injected into the neurons, in $[-A, A]$ with a uniform distribution;

$A[n]$ amplitude of noise n.

This NCNN model is a general form of chaotic neural networks with transient chaos and decaying noise. In the absence of noise, i.e., $n(t) = 0$, for all t, the NCNN as proposed in (7)–(10) reduces to the TCNN in [16]. In order to reveal the search ability of the NCNN, we will examine the nonlinear dynamics of the single neuron model in the next section and discuss the selection of model parameters in the model.

B. Noisy Chaotic Dynamics of the Single Neuron Model

The single neuron model for the NCNN is obtained from (7)–(10) by letting the number of neurons be one. The single neuron model for the TCNN can be obtained with the noise term $n(t)$ set to zero in (13)

$$x(t) = \frac{1}{1 + e^{-y(t)/\varepsilon}} \tag{12}$$

$$y(t+1) = ky(t) + \alpha I_i - z(t)\left[x(t) - I_0\right] + n(t) \tag{13}$$

$$z(t+1) = (1 - \beta_1)z(t) \tag{14}$$

$$A[n(t+1)] = (1 - \beta_2)A\left[n(t)\right]. \tag{15}$$

where I_i is the input bias of this neuron. We can transform (12) and (13) to a one-dimensional map from $y(t)$ to $y(t+1)$ by substituting (12) into (13)

$$y(t+1) = ky(t) + \alpha I_i - z(t)\left[\frac{1}{1 + e^{-y(t)/\varepsilon}} - I_0\right] + n(t). \tag{16}$$

There is a set of parameters in this model, i.e., k, α, ε, I_0, $z(0)$, $n(0)$, β_1, β_2. It is obvious that different values of these parameters will produce different neurodynamics. In order to investigate the dynamics of the NCNN model, we will vary the parameters above and plot the neuron dynamics and choose the set of parameter values which can produce richer and more flexible dynamics. Since α, β_1, ϵ, and I_0 were already discussed in detail in [16], we will not discuss these parameters but simply adopt these parameters

$$\alpha = 0.015 \quad \beta_1 = 0.001 \quad \varepsilon = 1/250 \quad I_0 = 0.65. \tag{17}$$

Since β_2 is the damping factor of noise which has a similar effect as the chaos damping factor β_1, we choose $\beta_2 = 0.0001$. We will then vary only the parameters left, i.e., the nerve membrane damping factor k, the initial negative self-interaction $z(0)$, and the initial noise amplitude $n(0)$ to investigate the dynamics while keeping the other parameters fixed.

Fig. 2 shows the dynamics of the single neuron for different parameter k. The x axis of each subfigure is time step t, and the y axis is the output of the neuron $x(t)$. We see from Fig. 2 that the larger the value of k, the more bifurcations they are. Since chaos is important for searching and k is between zero and one, we choose $k = 0.9$, as shown in Fig. 2(d).

Fig. 3 shows that a small value of $z(0)$, e.g., 0.01, cannot produce chaos. Larger values of $z(0)$ lead to more bifurcations, as shown in Fig. 3(c) and (d). But larger $z(0)$ also leads to more

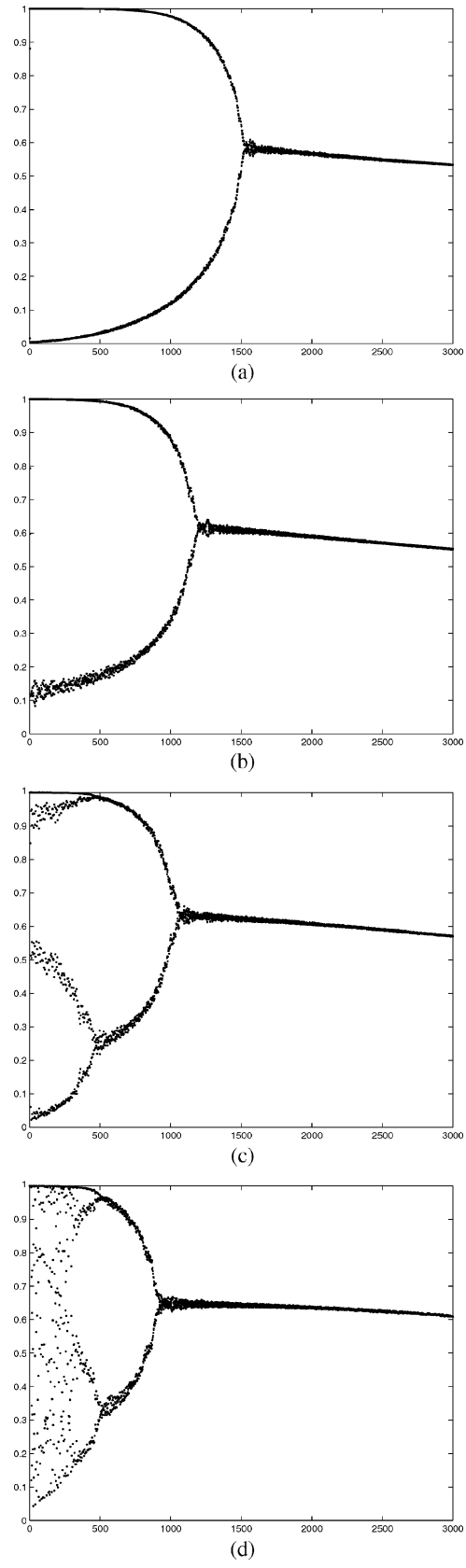


Fig. 2. Neurodynamics in the single neuron model with different parameter k and with $\alpha = 0.015$, $\beta_1 = 0.001$, $\beta_2 = 0.0001$, $\varepsilon = 1/250$, $I_0 = 0.65$, $z(0) = 0.08$, and $n(0) = 0.001$. (a) $k = 0.1$; (b) $k = 0.5$; (c) $k = 0.7$; and (d) $k = 0.9$.

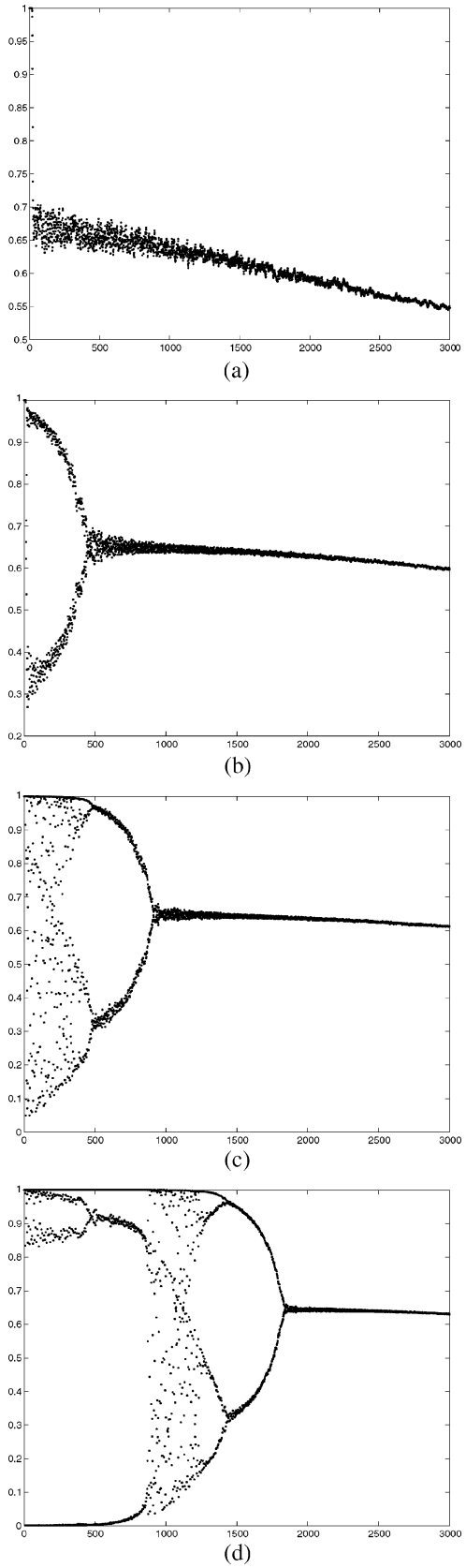


Fig. 3. Neurodynamics in the single neuron model with different parameter $z(0)$ and with $k = 0.9$, $\alpha = 0.015$, $\beta_1 = 0.001$, $\beta_2 = 0.0001$, $\varepsilon = 1/250$, $I_0 = 0.65$, and $n(0) = 0.001$. (a) $z(0) = 0.01$; (b) $z(0) = 0.05$; (c) $z(0) = 0.08$; and (d) $z(0) = 0.02$.

iteration steps until the chaos disappears, and a solution is found which inevitably results in longer computational time. Hence we choose a tradeoff between efficiency and solution quality, i.e., we use the value of $z(0) = 0.08$ in this paper.

Fig. 4 reveals that too much additive noise destroys the bifurcations [Fig. 4(d)]. On the other hand, if the magnitude of the additive noise is too small, it does not take effect on the stochastic search. Thus in this paper, we choose the initial noise amplitude as $n(0) = 0.001$.

Based on the above discussions on the selection of model parameters for the NCNN, we finally choose the set of parameters

$$k = 0.9 \quad \varepsilon = 1/250 \quad I_0 = 0.65 \quad z(0) = 0.08$$
$$n(0) = 0.001 \quad \beta_1 = 0.001 \quad \beta_2 = 0.0001. \tag{18}$$

The difference between the TCNN and the NCNN is the stochastic nature of the NCNN which the TCNN lacks. Chaos disappears after around 950 iterations through the reverse period-doubling bifurcations in the TCNN, whereas in the NCNN shown in Fig. 4(c), after the chaos disappears, the additive noise still remains and decays with time. With both stochastic nature for global searching and chaotic characteristic for efficient searching, the NCNN gradually approaches a dynamic structure similar to that of the Hopfield neural network and converges to a stable fixed point.

IV. Noisy Chaotic Neural Network for the BSP

A two-phase optimization is adopted in this paper like previous work [6]–[8]. In the first phase, in order to obtain a minimal TDMA length M, a gradual expansion scheme (GES) [6] is combined with our NCNN, i.e., a gradual noisy chaotic neural network (G-NCNN) is adopted in this phase. In the second phase, we use the NCNN to obtain a maximal number of conflict-free transmissions based on the results obtained in the first phase.

A. Minimizing the TDMA Frame Length Using a G-NCNN

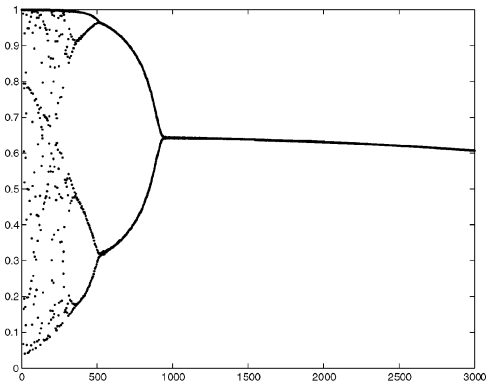
Consider the first objective of the BSP. In order to obtain the minimal number of time slots M, we start to search for solutions with a small M and increase M until a feasible solution is found. The G-NCNN consists of $M \times N$ neurons. M is initially set as its lower bound value L_m. The GES stops when the G-NCNN finds a feasible assignment and the current number of time slots together with its transmission assignments are the optimal results for phase I of the BSP.

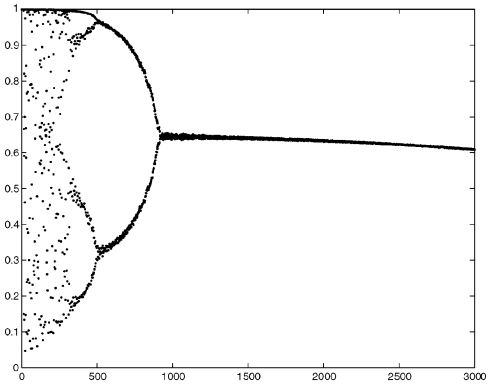
The lower bound L_m can be computed using the following equation [4]:

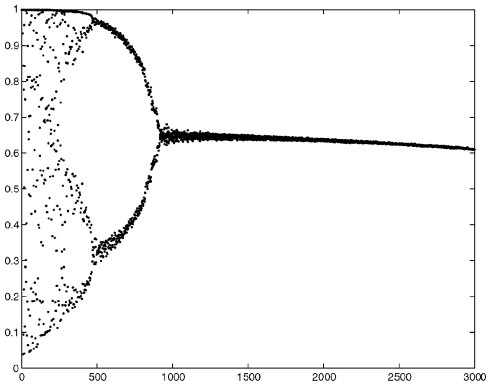
$$L_m = \max_{i \in V} \deg_i + 1 \tag{19}$$

where $\deg_i$ means the degree of node i in the PRN and the degree of a node is defined as the number of edges connected to this node

$$\deg_i = \sum_{k=1, k \neq i}^{N} c_{ik}. \tag{20}$$


(a)


(b)


(c)

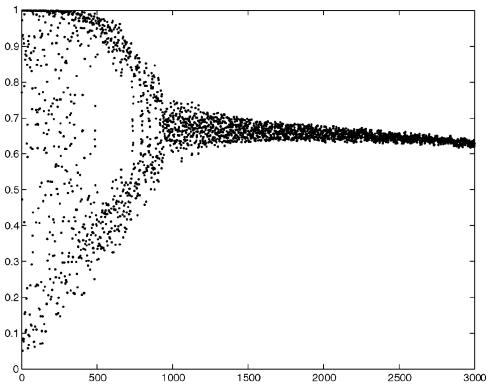

(d)

Fig. 4. Neurodynamics in the single neuron model with different parameter $n(0)$ and with $k = 0.9$, $\alpha = 0.015$, $\beta_1 = 0.001$, $\beta_2 = 0.0001$, $\varepsilon = 1/250$, $I_0 = 0.65$, and $z(0) = 0.08$. (a) $n(0) = 0.0001$; (b) $n(0) = 0.0005$; (c)$n(0) = 0.001$; and (d) $n(0) = 0.005$.

Note that L_m obtained from (19) is not a tight bound. A tighter bound [8] can be easily obtained using graph theory [28]. First the original graph $G = (V, E)$ is transformed into $G' = (V, E')$, where E in G stands for one-hop-away edges and E' in G' stands for one-hop-away and two-hop-away edges. The tight bound L'_m is

$$L'_m = \omega(G') \tag{21}$$

where $\omega(G')$ is the maximal cardinality of a clique in G' [8]. In this paper, we use the tight bound L'_m as in (21).

In this paper, different from the GNN in [6], where the neurons are expanded gradually at every P iterations during the iterative computation of the neural network, we implement the GES based on a convergence index $\delta(t)$ of the network energy, which we defined as

$$\delta(t) = \sum_{q=t-4}^{t} |E(q) - E(q-1)| / E(0) \tag{22}$$

where $E(q)$ is the value of energy function at time step q. If index $\delta(t)$ is less than a very small value, e.g., $\delta(t) < 10^{-4}$ in our simulation, the neural network is considered as having fully converged. If the network has converged but no feasible solutions are found using the current number of time slots, the number of time slots is increased by one, i.e., $M \rightarrow M + 1$, and the G-NCNN restarts to search for optimal solutions with the updated number of neurons.

A partial scheduling [6] is implemented which aims at reducing the computational time. Since any nodes in one-hop away should be assigned with a different time slot, we find a group of nodes $G = \{g(j)\}$, $(j = 1, 2, \ldots, L_m)$, consisting of the node with the maximum degree and its one-hop-away nodes. Each node in G is assigned to a different time slot before the G-NCNN begins to compute. After the partial scheduling, the outputs of the corresponding neurons t_{ij} $(i = 1, 2, \ldots, L_m, j = 1, 2, \ldots, L_m)$ are fixed and further computations on these neurons are skipped.

The energy function E_1 for phase I is given as follows [6]:

$$E_1 = \frac{W_1}{2} \sum_{j=1}^{N} \left(\sum_{k=1}^{M} x_{jk} - 1 \right)^2 + \frac{W_2}{2} \sum_{j=1}^{N} \sum_{i=1}^{M} \sum_{k=1, k \neq i}^{N} d_{jk} x_{ij} x_{ki} \tag{23}$$

where W_1 and W_2 are weighting coefficients. The W_1 term represents the constraint that each of the N nodes in the PRN must transmit exactly once during each TDMA cycle. The W_2 term indicates the constraint that any pair of nodes that is one-hop or two-hop away must not transmit simultaneously during each TDMA cycle.

From (8), (11), and (23), we obtain the dynamics of the G-NCNN as follows:

$$\begin{aligned} y_{jk}(t+1) &= k y_{jk}(t) - z(t)\left[x_{jk}(t) - I_0\right] + n(t) \\ &\quad + \alpha \left\{ -W_1 \left(\sum_{k=1}^{M} x_{jk} - 1 \right) - W_2 \sum_{k=1, k \neq j}^{N} d_{jk} x_{ki} \right\}. \end{aligned} \tag{24}$$

B. Maximizing the Node Transmissions Using the NCNN

After phase I, the minimal TDMA frame length M is found and each node is assigned with one and exactly one time slot. In phase II, we aim at maximizing the channel utilization by adding as many conflict-free transmissions as possible to the TDMA frame. Because in phase I one node is assigned with exactly one slot in order to find a minimal frame length, there are many nodes that can use other time slots without violating the no-conflict constraint. Thus, additional transmissions may be found on some nodes but frame length M and the assigned transmissions in phase I are fixed [6]. We use the energy function as follows [6]:

$$E_2 = \frac{W_3}{2}\sum_{j=1}^{N}\sum_{i=1}^{M}\sum_{k=1,k\neq i}^{N} d_{jk}x_{ij}x_{ki} + \frac{W_4}{2}\sum_{j=1}^{N}\sum_{i=1}^{M}(1-x_{ij})^2 \tag{25}$$

where W_3 and W_4 are weighting coefficients. W_3 represents the constraint term that any pair of nodes that is one-hop or two-hops away must not transmit simultaneously during each TDMA cycle. W_4 is the optimization term which maximizes the total number of firing neurons.

From (8), (11), and (25), we obtain the dynamics of the NCNN for phase II of the BSP as follows:

$$y_{jk}(t+1) = k y_{jk}(t) - z(t)\left[x_{jk}(t) - I_0\right] + n(t) + \alpha\left\{-W_3\sum_{k=1,k\neq j}^{N} d_{jk}x_{ki} + W_4(1-x_{ij})\right\}. \tag{26}$$

The neuron output is continuous between zero and one, we convert the continuous output x_{ij} of neuron ij to discrete neuron output (t_{ij}) as follows [15]:

$$t_{ij} = \begin{cases} 1, & \text{if } x_{ij} > \sum_{k=1}^{N}\sum_{l=1}^{M} x_{kl}(t)/(N\times M) \\ 0, & \text{otherwise} \end{cases}$$

where the output of neuron $T = \{t_{ij}\}$ is the binary matrix mentioned in Section II.

The NCNN is updated cyclically and asynchronously. The new state information of a neuron is immediately available for the other neurons in the next iteration. The iteration is terminated once a feasible transmission schedule is obtained, i.e., the transmissions of all nodes are conflict-free.

We noted that previous methods, such as the MFA [4], the HNN-GA [8], and the GNN [6], had no discussions on the issue of time complexity. And the sequential vertex coloring algorithm (SVC) [7] is a polynomial time algorithm that has $O(N^3)$ computational complexity for the entire search. Reference [29] showed that the worst time complexity in one iteration step is $O(NM^2)$. In phase I, the G-NCNN has the worst time complexity of $O(NM^2)$ for one iteration step. In phase II, the complexity is down to $O(NM)$. Hence, our algorithm has the worst time complexity of $O(NM^2)$ in one iteration step. However, it is difficult to determine the exact number of iterations required for different problem instances with various problem size.

V. Simulation Results

A. Parameter Selection

An issue related to the efficiency of the NCNN model in solving combinatorial optimization problems is how to select appropriate weighting coefficients in the energy function although there are quite a few parameters to be selected in the NCNN (including model parameters and weighting coefficients). Fortunately these parameters are similar to those used in other optimization problems [20]–[22]. The set of model parameters in (23)–(26) was discussed in Section III-B and listed in (18). The selection of weighting coefficients in the energy function is based on the rule that all terms in the energy function should be comparable in magnitude, so that none of them dominates. Thus we choose the coefficients of the two energy functions as follows:

$$W_1 = 1.0 \quad W_2 = 1.0 \quad W_3 = 1.0 \quad W_4 = 1.0. \tag{27}$$

Note that the parameters are chosen experimentally, and tuning of these parameters may be necessary when solving different optimization problems or different instances of the same optimization problem, but from our experience, the tuning is only on a small scale. In this paper, we use the parameters for the benchmark examples as listed in Table VIII.

B. Evaluation Indexes

We use three evaluation indexes to compare with different algorithms. One is the TDMA frame length M. The second index is the channel utilization factor ρ defined in (4). The third is the average time delay η for each node to broadcast packets [6]

$$\eta = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{M}{\sum_{j=1}^{M} t_{ij}}\right) = \frac{M}{N}\sum_{i=1}^{N}\left(\frac{1}{\sum_{j=1}^{M} t_{ij}}\right). \tag{28}$$

Another definition of the average time delay can be found in [4]. To derive the definition of the average time delay, the following assumptions were made [4].

1) Packets have a fixed length, and the length of a time slot is equal to the time required to transmit a packet.
2) The interarrival time for each node i is statistically independent from those of the other nodes, and packets arrive according to a *Poisson* process with a rate of λ_i (packets/slot). The total traffic in node i consists of its own traffic and the incoming traffic from the other nodes. Packets are stored in buffers in each node and the buffer size is infinite.
3) The probability distribution of the service time of node i is deterministic. Let the service rate of node i be μ_i (packets/slot).
4) Packets can be transmitted only at the beginning of each time slot.

Under the above assumptions, the PRN can be modeled as N $M/D/1$ queues. According to Pollaczek–Khinchin formula [23], the average time delay for each node D_i is given as follows:

$$D_i = \frac{1}{\mu_i} + \frac{1}{\lambda_i}\frac{(\lambda_i/\mu_i)^2}{2(1-\lambda_i/\mu_i)} \tag{29}$$

where $\mu_i = \sum_{m=1}^{M} x_{mi}/M$ (packets/slot) is the service rate for node i. The total time delay is given by [4]

$$D = \frac{\sum_{i=1}^{N}\lambda_i D_i}{\sum_{i=1}^{N}\lambda_i}. \tag{30}$$

TABLE I
SPECIFICATIONS OF THE THREE BENCHMARK EXAMPLES GIVEN BY [4] (BM #1, BM #2, AND BM #3) AND THE GEOMETRIC INSTANCES RANDOMLY GENERATED AS IN [6] (CASES 1–20). THE LOWER BOUNDS FOR THE NUMBER OF TIME SLOTS M FOR CASES 1–20 ARE DISPLAYED AS AVERAGE/MAXIMUM/MINIMUM VALUES

Instance	Nodes N	Edges E	Max D	Min d	Lower bound M
BM #1	15	29	7	2	8
BM #2	30	70	8	2	10
BM #3	40	66	7	1	8
case #1	100	150.7	7.3	0.0	8.3/10/7
case #2	300	445.4	8.3	0.0	9.3/11/8
case #3	500	763.3	8.8	0.0	9.8/11/9
case #4	750	1279.8	8.9	2.1	10.0/13/9
case #5	1000	1732.0	9.4	2.0	10.6/14/9
case #6	100	558.0	19.7	1.0	20.7/21/16
case #7	300	1720.5	22.1	2.2	22.7/26/19
case #8	500	2834.5	22.0	2.5	23.0/24/21
case #9	750	4500.2	23.1	2.3	24.5/29/21
case #10	1000	5936.3	25.2	2.0	24.7/28/20
case #11	100	1101.2	33.7	5.9	36.3/43/29
case #12	300	3655.4	40.3	5.4	42.0/48/32
case #13	500	6299.7	42.5	5.7	43.5/50/37
case #14	750	9664.5	43.3	5.9	44.3/51/38
case #15	1000	12993.4	44.2	5.6	45.3/52/40
case #16	100	1755.6	55.4	12.3	57.4/66/46
case #17	300	6168.3	64.3	11.4	66.2/75/59
case #18	500	11201.1	65.3	11.5	68.1/77/62
case #19	750	16621.1	67.0	11.9	68.6/83/62
case #20	1000	22543.2	69.2	13.1	71.6/78/65

TABLE II
COMPARISONS OF AVERAGE DELAY TIME η GIVEN BY (26) AND NUMBER OF TIME SLOTS M OBTAINED BY THE G-NCNN AND OTHER ALGORITHMS FOR THE THREE BENCHMARK PROBLEMS GIVEN BY [4]

BM	G-NCNN η/M	HNN-GA η/M	SVC η/M	GNN η/M	MFA η/M
BM #1	6.8/8	7.0/8	7.2/8	7.1/8	7.2/8
BM #2	9.0/10	9.3/10	10.0/10	9.5/10	10.5/12
BM #3	5.8/8	6.3/8	6.76/8	6.2/8	6.9/9

TABLE III
PAIRED T-TEST OF AVERAGE TIME DELAY η (SECOND) BETWEEN THE HNN-GA AND THE G-NCNN

Instance	Node	HNN-GA	G-NCNN
BM #1	15	6.84	6.84
BM #2	30	9.17	9.00
BM #3	40	6.04	5.81
Case #4	60	15.74	13.40
Case #5	80	16.33	14.48
Case #6	100	17.17	15.16
Case #7	120	17.85	16.02
Case #8	150	20.47	16.37
Case #9	180	20.04	16.38
Case #10	200	20.31	17.22
Case #11	230	20.36	16.58
Case #12	250	20.25	17.17
T-Value = 5.22			
P-Value (one-tail) = 0.0001			
P-Value (two-tail) = 0.0003			

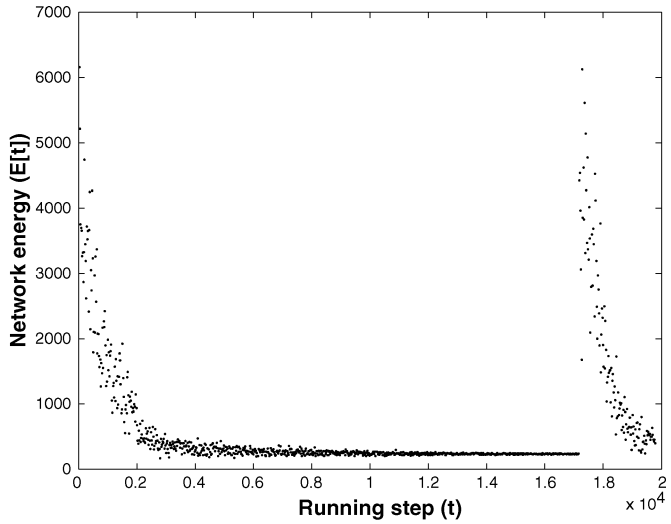


Fig. 5. Noisy chaotic dynamics of the neural network energy for benchmark BM #2.

In this paper, we will use both definitions of the average time delay in (28) and (30) in order to compare with other methods.

C. Benchmark Problems

In order to evaluate the performance of our NCNN-based algorithm, we first compare it with other methods on the three benchmark problems in [14], which have been solved by all the methods mentioned, i.e., MFA [4], the GNN [6], the HNN-GA technique [8], and the SVC [7]. We then compare the performance between the G-NCNN and the GNN [6] on a total of 600 randomly generated geometric graph instances. We use the method in [6] to generate the random instances.

1) Set the number of nodes N for the instance to be generated and the edge generation parameter r to be used below. Following [6], we choose $r = 1/\sqrt{N}$ in cases 1–5, $r = 2/\sqrt{N}$ in cases 6–10, $r = 3/\sqrt{N}$ in cases 11–15, and $r = 4/\sqrt{N}$ in cases 16–20.
2) Generate N two-dimensional coordinates randomly

$$x_i = \text{Random}[0,1] \quad y_i = \text{Random}[0,1], \qquad \text{for } i = 1, \ldots, N \tag{31}$$

where x_i and y_i are the x and y coordinates of node i, respectively, and Random$[z, w]$ generates a random number uniformly distributed between z and w.
3) Assign an edge for a pair of nodes whose distance is less than r

$$\text{if } \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} < r, \text{ then } c_{ij} = 1$$
$$\text{else } c_{ij} = 0 \text{ for } i = 1, \ldots, N \text{ and } j = 1, \ldots, N. \tag{32}$$

Each of the 20 geometric graphs is randomly generated 30 times as shown in Table I. As we use the same methods as proposed in [6], our randomly generated graphs are statistically identical to those in [6].

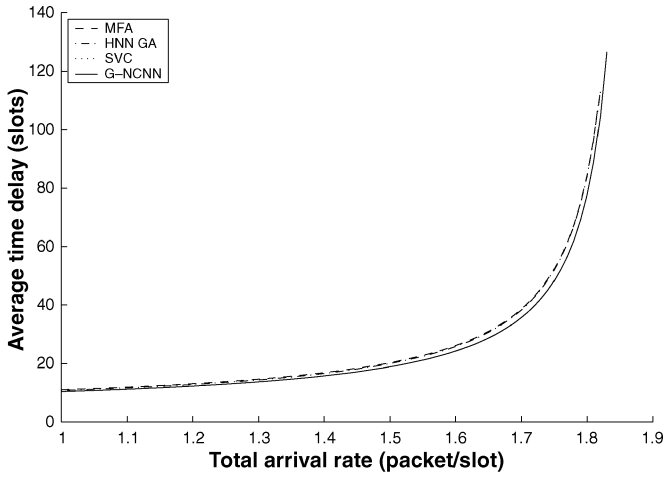


Fig. 6. Comparison of the average time delay as a function of the total arrival rate for the 15-node-29-edge benchmark [according to the Pollaczek–Khinchin formula given by (28)] among different approaches.

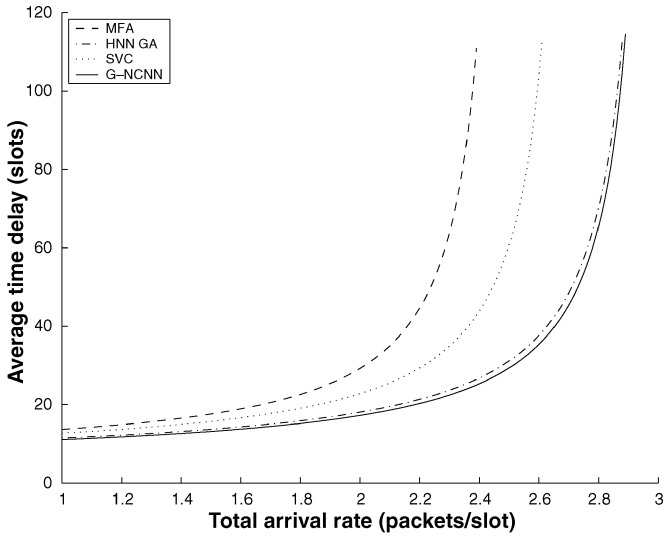


Fig. 7. Same as Fig. 6, for the 30-node-70-edge benchmark.

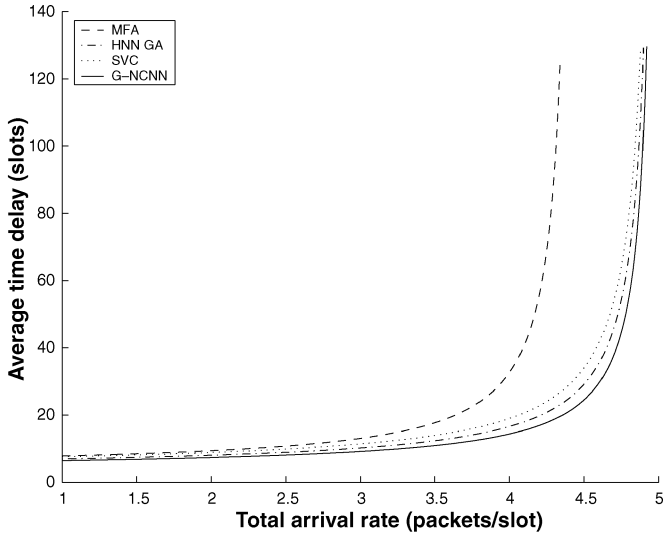


Fig. 8. Same as Fig. 6, for the 40-node-66-edge benchmark.

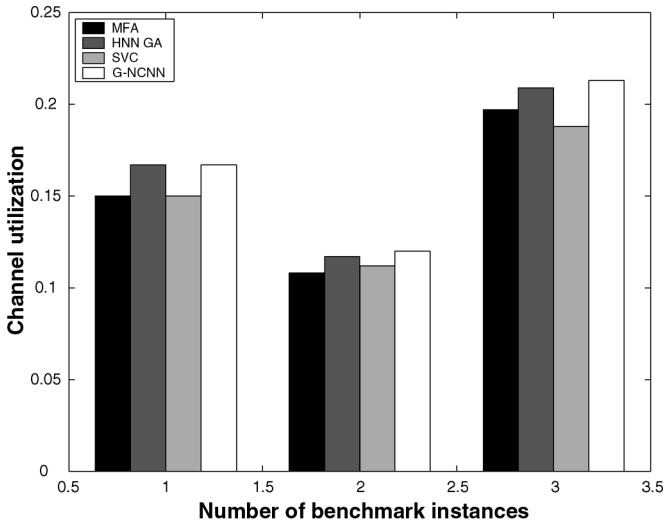


Fig. 9. Comparisons of channel utilization for the three benchmark problems. The numbers 1, 2, and 3 in the horizontal axis stand for benchmark problems BM #1, BM #1, and BM #3 in Table I, respectively.

D. Result Evaluations and Discussions

Three benchmark problems from [4] have been chosen to compare with other algorithms in [4] and [6]–[8]. The three examples are instances with 15-node-29-edge (BM #1), 30-node-70-edge (BM #2), and 40-node-66-edge (BM #3), respectively.

The dynamics of the G-NCNN energy in phase I of BM #2 are plotted in Fig. 5. From Fig. 5, we can see clearly that the energy does not decrease smoothly but fluctuates due to the noisy and chaotic nature of the G-NCNN model. The energy gradually settles down to a stable value. The gradual expansion scheme adds the number of time slots by one when there are no feasible solutions with the current number of time slots and the G-NCNN restarts the search again. Such a re-searching procedure repeats until a feasible assignment is found and the G-NCNN converges.

We labeled our two-phase methods which consist of the G-NCNN in phase I and the NCNN in phase II as G-NCNN for short while comparing with other methods. The average time delay for the three benchmark problems is computed based on the final TDMA schedules. Because no schedules have been published on these three benchmark problems in [6], we only compared our G-NCNN with MFA, HNN-GA, and SVC. Figs. 6–8 show that the time delays obtained by the G-NCNN are much smaller compared to those obtained by the MFA algorithm in all three instances. In the 15-node instance,

TABLE IV
RESULTS OF THE TCNN AND THE G-NCNN IN THE 100-NODE 522-EDGE BENCHMARK INSTANCE USING VARIOUS NOISE LEVELS IN 20 DIFFERENT RUNS, WHERE SD STANDS FOR STANDARD DEVIATION

	Noise Level	Minimum η	Average η	SD	Converge Rate
TCNN	n(0)=0.0	15.38	15.56	0.06	75%
G-NCNN	n(0)=0.001	15.28	15.45	0.16	100%
	n(0)=0.002	15.23	15.57	0.13	100%
	n(0)=0.005	15.16	15.32	0.14	100%
	n(0)=0.009	14.98	15.12	0.10	100%

TABLE V
SAME AS TABLE IV FOR THE 250-NODE 1398-EDGE BENCHMARK INSTANCE

	Noise Level	Minimum η	Average η	SD	Converge Rate
TCNN	n(0)=0.0	17.13	17.15	0.03	90%
G-NCNN	n(0)=0.001	16.73	16.90	0.09	100%
	n(0)=0.002	16.80	16.83	0.05	100%
	n(0)=0.005	16.44	16.52	0.07	100%
	n(0)=0.009	INF	INF	-	0%

TABLE VI
SAME AS TABLE IV FOR THE 750-NODE 4371-EDGE BENCHMARK INSTANCE

	Noise Level	Minimum η	Average η	SD	Converge Rate
TCNN	n(0)=0.0	N/A	N/A	-	0%
G-NCNN	n(0)=0.001	18.16	18.35	0.11	100%
	n(0)=0.002	18.04	18.20	0.10	70%
	n(0)=0.005	INF	INF	-	0%
	n(0)=0.009	INF	INF	-	0%

the MFA, SVC, and HNN-GA obtained the same time delay, but the time delay obtained by our G-NCNN is less than their results, as we can see from Fig. 6. Because BM #1 is a rather small instance with only 15 nodes, the improvement is not much for this case. With increasing problem sizes, improvements made by the NCNN become more evident in BM #2 (Fig. 7) and BM #3 (Fig. 8). Fig. 9 shows that the G-NCNN can find solutions with higher channel utilization compared to MFA, HNN-GA, and SVC. The computational results for the three benchmark problems are also summarized in Table II. It shows that our G-NCNN can find shorter average time delay as given by (28) in all three benchmark problems compared to the other methods.

In order to show the difference between the HNN-GA and the NCNN, a paired t-test is performed between the two methods, as shown in Table III. We compared the two methods in 12 cases with node size from 15 to 250, where BM #1 to BM #3 are benchmark examples and cases 4–12 are randomly generated instance with edge generation parameter $r = 2/\sqrt{N}$. The results show that the P-value is 0.0001 for one-tail test and 0.0003 for two-tail test. We found that the G-NCNN (mean $= 13.7$, standard deviation $= 4.12$) reported having significantly better performance than did the HNN-GA (mean $= 15.9$, standard deviation $= 5.45$) did, with T-value $t(11) = 5.22$ and P-value < 0.05. From Figs. 6–8 and Tables II and III, it can be concluded that the G-NCNN can always find the shortest conflict-free frame schedule while providing the maximum channel utilization compared to existing algorithms.

In order to show the effects of noise in the computation of the G-NCNN model, we compared the NCNN model with and without noise in three random generated instance with node size 100, 250, and 750. Tables IV–VI are the results of time delay η for the three instances. The comparisons are performed between the NCNN with different noise amplitude $(n(0) > 0)$ and the TCNN with $n(0) = 0$. We ran the simulations of each instance 20 different times using the model parameters and coefficient weights in (17), (18), and (27). "INF" in Tables V and VI means that the NCNN algorithms can find feasible solution in the first phase but failed to converge to a feasible one within predefined steps in the second phase, due to the large amplitude of additive noise. "N/A" in Table VI means that the algorithm failed to find a solution even in the first phase of the BSP. From the three tables we can draw the conclusion that both the convergence and the time delay obtained by the NCNN are better than the TCNN in all three instances. The TCNN can find solutions in 100- and 250-node instances, but when applied to large instances like the 750-node instance, the TCNN failed in all 20 runs. Among different noise levels, it shows that noise level with amplitude $n(0) = 0.002$ has the best performance among all noise levels tested.

TABLE VII
COMPARISONS OF AVERAGE DELAY TIME η GIVEN BY (26) AND TIME SLOT M OBTAINED BY THE G-NCNN AND THE GNN ON THE RANDOMLY GENERATED INSTANCES. THE RESULTS OF THE G-NCNN ARE DISPLAYED AS AVERAGE $\pm$ STANDARD DEVIATION MAXIMUM MINIMUM. THE RESULTS OF THE GNN ARE DISPLAYED AS AVERAGE/MAXIMUM/MINIMUM

	G-NCNN		GNN	
	η	M	η	M
#1	5.1±0.4/5.8/4.3	8.4/11/6	6.3/7.5/5.5	8.1/11/7
#2	5.2±0.2/5.6/4.7	9.3/12/8	6.7/7.5/6.0	9.6/12/8
#3	5.2±0.2/5.6/4.7	10.0/12/8	6.8/7.4/6.3	10.0/12/9
#4	5.2±0.1/5.5/5.0	10.1/12/9	6.9/7.5/6.5	10.2/13/9
#5	5.2±0.1/5.5/5.0	9.8/12/9	7.0/7.5/6.6	10.6/12/9
#6	15.1±0.8/16.5/13.8	19.1/23/16	17.4/20.0/15.6	20.2/24/18
#7	16.8±0.5/18.2/16.1	22.5/25/21	19.7/20.8/18.0	23.3/26/20
#8	17.5±0.4/18.2/17.0	23.1/28/21	20.3/22.7/18.8	24.2/30/22
#9	17.7±0.3/18.2/17.2	25.1/28/23	20.9/22.9/19.7	25.4/29/23
#10	17.9±0.2/18.3/17.6	25.2/29/24	21.2/22.2/20.3	25.5/28/24
#11	30.3±1.7/33.4/26.7	34.3/42/29	34.1/38.2/30.0	37.4/46/32
#12	34.9±0.9/37.1/33.3	42.9/48/38	39.1/41.9/36.0	43.2/48/38
#13	36.3±0.6/37.6/35.1	43.3/47/40	41.0/44.8/39.0	45.2/52/42
#14	37.7±0.5/38.7/36.8	46.0/52/43	42.9/45.4/41.2	47.3/53/45
#15	38.3±0.5/39.3/37.5	47.2/52/44	43.7/46.2/42.3	48.1/52/46
#16	49.8±2.7/54.5/45.8	56.3/66/46	54.5/60.0/45.9	57.9/65/48
#17	57.1±1.5/59.7/54.3	67.6/80/61	63.6/71.5/58.2	68.4/79/61
#18	61.0±1.3/63.1/58.4	71.5/81/66	67.4/72.1/63.2	72.2/79/66
#19	63.3±1.0/65.4/61.3	73.7/81/70	70.3/75.4/67.4	75.2/83/71
#20	65.1±1.0/68.5/63.3	76.8/82/72	72.9/75.0/69.7	78.2/82/73

Table VII shows the average, maximum, and minimum values of two indexes η and M of solutions for 600 instances solved by the G-NCNN and the GNN for the 20 randomly generated cases (30 instances for each case). From the results, we can see that the G-NCNN always finds shorter average time delay η than the GNN does in all cases. We use the ANOVA software[1] to obtain the standard deviations and the error bars for the G-NCNN (Fig. 10). However, the standard deviations for the GNN are not available; thus we plotted the best results for the GNN in each case for comparisons. From Fig. 10, we can see that the time delay obtained by our G-NCNN is smaller than the best results from the GNN in most cases.

VI. CONCLUSION

In this paper, we present a noisy chaotic neural network model for solving the broadcast scheduling problem in packet

[1]http://www.physics.csbsju.edu/stats/anova.html

TABLE VIII
PARAMETERS OF THE G-NCNN MODEL FOR THE BENCHMARK EXAMPLES WITH $W_1 = W_2 = W_3 = 1.0, k = 0.9, \epsilon = 1/250, I_0 = 0.65, \alpha = 0.015$

Case	W_4	$z(0)$	$n(0)$	β_1	$\beta 2$
BM #1	0.6	0.08	0.001	0.001	0.0001
BM #2	1.0	0.08	0.001	0.001	0.0001
BM #3	1.0	0.08	0.001	0.001	0.0001
case #1	1.0	0.08	0.002	0.001	0.0001
case #2	1.0	0.08	0.002	0.001	0.0001
case #3	1.0	0.08	0.002	0.001	0.0001
case #4	1.0	0.08	0.002	0.001	0.0001
case #5	1.0	0.08	0.002	0.001	0.0001
case #6	0.8	0.08	0.002	0.001	0.0001
case #7	0.8	0.08	0.002	0.001	0.0001
case #8	0.8	0.08	0.002	0.001	0.0001
case #9	0.8	0.08	0.002	0.001	0.0001
case #10	0.8	0.08	0.002	0.001	0.0001
case #11	0.8	0.08	0.002	0.001	0.0001
case #12	0.8	0.08	0.002	0.001	0.0001
case #13	0.8	0.08	0.002	0.001	0.0001
case #14	0.8	0.08	0.002	0.001	0.0001
case #15	0.8	0.08	0.002	0.001	0.0001
case #16	0.6	0.09	0.003	0.001	0.0005
case #17	0.6	0.09	0.003	0.001	0.0005
case #18	0.6	0.09	0.003	0.001	0.0005
case #19	0.6	0.09	0.003	0.001	0.0005
case #20	0.6	0.10	0.003	0.001	0.0005

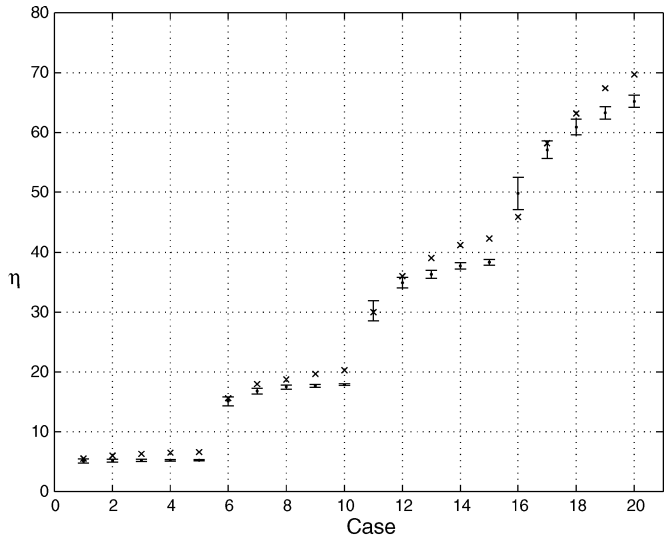


Fig. 10. Error bars for the average delay time η given by (26) obtained using the G-NCNN for 20 randomly generated cases of the BSP. Since the standard deviations for the GNN are not available, only the best (minimum) results for the GNN are plotted here (x-marks).

radio networks. A two-phase optimization is adopted to solve the two objectives of the BSP with two different energy functions. A G-NCNN is proposed to find an optimal transmission schedule with the minimal TDMA frame length in the first phase. In the second phase, additional node transmissions are found using the NCNN based on the results from the first phase. We evaluate our G-NCNN algorithm in three benchmark examples and 600 randomly generated geometric graph instances. We compare our results with existing methods including mean filed annealing, HNN-GA, the sequential vertex coloring algorithm, and the gradually neural network. The results of the benchmark problems show that the G-NCNN always finds the best solutions with minimal average time delays and maximal channel utilization among the existing methods. We shows that our G-NCNN has significant improvements over the HNN-GA through a paired t-test. We also compared the NCNN with the TCNN in three instances and showed that the NCNN has better performance than the TCNN. These results, together with the results for other optimization problems [20]–[22], support the conclusion that the G-NCNN is an efficient approach for solving large-scale combinatorial optimization problems.

ACKNOWLEDGMENT

The authors would like to sincerely thank the Associate Editor and the reviewers for their constructive comments and suggestions that helped to improve the manuscript significantly.

REFERENCES

[1] B. M. Leiner, D. L. Nielson, and F. A. Tobagi, "Issues in packet radio network design," *Proc. IEEE*, vol. 75, no. 1, pp. 6–20, Jan. 1987.
[2] A. Ephremides and T. V. Truong, "Scheduling broadcast in multihop radio networks," *IEEE Trans. Commun.*, vol. 38, no. 6, pp. 456–460, Jun. 1990.
[3] N. Funabiki and Y. Takefuji, "A parallel algorithm for broadcast scheduling problems in packet radio networks," *IEEE Trans. Commun.*, vol. 41, no. 6, pp. 828–831, Jun. 1993.
[4] G. Wang and N. Ansari, "Optimal broadcast scheduling in packet radio networks using mean field annealing," *IEEE J. Sel. Areas Commun.*, vol. 15, no. 2, pp. 250–260, Feb. 1997.
[5] G. Chakraborty and Y. Hirano, "Genetic algorithm for broadcast scheduling in packet radio networks," *IEEE World Congr. Computational Intelligence*, pp. 183–188, 1998.
[6] N. Funabiki and J. Kitamichi, "A gradual neural network algorithm for broadcast scheduling problems in packet radio networks," *IEICE Trans. Fund.*, vol. E82-A, no. 5, pp. 815–824, 1999.
[7] J. Yeo, H. Lee, and S. Kim, "An efficient broadcast scheduling algorithm for TDMA ad-hoc networks," *Comput. Oper. Res.*, no. 29, pp. 1793–1806, 2002.
[8] S. Salcedo-Sanz, C. Bousoño-Calzón, and A. R. Figueiras-Vidal, "A mixed neural-genetic algorithm for the broadcast scheduling problem," *IEEE Trans. Wireless Commun.*, vol. 2, no. 2, pp. 277–283, Mar. 2003.
[9] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci. USA*, vol. 81, pp. 3088–3092, May 1984.
[10] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.
[11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimisation by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.
[12] C. Peterson and J. R. Anderson, "A mean field theory algorithm for neural networks," *Complex Syst.*, vol. 1, pp. 995–1019, 1987.
[13] 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.
[14] ——, "Solution of the optimization problem using the neural network model as a globally coupled map," in *Towards the Harnessing of Chaos*M. Yamaguti, Ed. Amsterdam, The Netherlands: Elsevier Science, 1994, pp. 99–114.
[15] L. Chen and K. Aihara, "Transient chaotic neural networks and chaotic simulated annealing," in *Towards the Harnessing of Chaos*, M. Yamguti, Ed. Amsterdam, The Netherlands: Elsevier Science, 1994, pp. 347–352.
[16] ——, "Chaotic simulated annealing by a neural network model with transient chaos," *Neural Netw.*, vol. 8, no. 6, pp. 915–930, 1995.
[17] ——, "Global searching ability of chaotic neural networks," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 46, no. 8, pp. 974–993, Aug. 1999.
[18] I. Tokuda, K. Aihara, and T. Nagashima, "Adaptive annealing for chaotic optimization," *Phys. Rev. E*, vol. 58, pp. 5157–5160, 1998.

[19] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 6, no. 6, pp. 721–741, Nov. 1984.
[20] L. Wang and F. Tian, "Noisy chaotic neural networks for solving combinatorial optimization problems," in *Proc. Int. Joint Conf. Neural Networks (IJCNN 2000)*, Como, Italy, Jul. 24–27, 2000, vol. 4, pp. 37–40.
[21] S. Li and L. Wang, "Channel assignment for mobile communications using stochastic chaotic simulated annealing," in *Proc. 2001 Int. Work-Conf. Artificial Neural Networks (IWANN2001)*, J Mira and A. Prieto, Eds., Granada, Spain, Jun. 13–15, 2001, vol. 2084, pp. 757–764, Part I, Lecture Notes in Computer Science.
[22] L. Wang, S. Li, F. Tian, and X. Fu, "A noisy chaotic neural network for solving combinatorial optimization problems: Stochastic chaotic simulated annealing," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 5, pp. 2119–2125, Oct. 2004.
[23] D. Bertsekas and R. Gallager, *Data Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1987.
[24] R. L. Wang, Z. Tang, and Q. P. Cao, "A Hopfield network learning method for bipartite subgraph problem," *IEEE Trans. Neural Netw.*, vol. 15, no. 6, pp. 1458–1465, Nov. 2004.
[25] H. Tang, K. C. Tan, and Z. Yi, "A columnar competitive model for solving combinatorial optimization problems," *IEEE Trans. Neural Netw.*, vol. 15, no. 6, pp. 1568–1573, Nov. 2004.
[26] R. S. T. Lee, "A transient-chaotic autoassociative network (TCAN) based on Lee oscillators," *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1228–1243, Sep. 2004.
[27] T. Kwok and K. A. Smith, "A noisy self-organizing neural network with bifurcation dynamics for combinatorial optimization," *IEEE Trans. Neural Netw.*, vol. 15, no. 1, pp. 84–98, Jan. 2004.
[28] D. Jungnickel, *Graphs, Networks and Algorithms*. Berlin, Germany: Springer-Verlag, 1999.
[29] N. Funabiki and S. Nishikawa, "A gradual neural-network appraoch for frequency assignment in satellite communication systems," *IEEE Trans. Neural Netw.*, vol. 8, no. 6, pp. 1359–1370, Nov. 1997.

Lipo Wang (M'97–SM'98) received the B.S. degree from National University of Defense Technology, China, in 1983 and the Ph.D. degree from Louisiana State University, Baton Rouge, in 1988.

In 1989, he worked at Stanford University, Stanford, CA, as a postdoctoral fellow. In 1990, he was a faculty member at the Department of Electrical Engineering, University College, Australian Defence Force Academy (ADFA), University of New South Wales, Canberra, Australia. From 1991 to 1993, he was on the staff of the Laboratory of Adaptive Systems, National Institute of Health, Bethesda, MD. From 1994 to 1997, he was a tenured faculty member in computing at Deakin University, Australia. Since 1998, he has been an Associate Professor at School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, Singapore. He is author or coauthor of more than 60 journal publications, 12 book chapters, and 90 conference presentations. He has received one U.S. patent in neural networks. He is the author of two monographs and has edited 16 books. He was Keynote/Panel Speaker for several international conferences. He has been an Associate Editor of *Knowledge and Information Systems* since 1998 and Area Editor of *Soft Computing* since 2002. He is/was Editorial Board Member of four additional international journals. He has been on the Governing Board of the Asia-Pacific Neural Network Assembly since 1999 and was its President in 2002–2003. He has been General/Program Chair for 11 international conferences and a member of steering/advisory/organizing/program committees of more than 100 international conferences. His research interests include computational intelligence with applications to data mining, bioinformatics, and optimization.

Dr. Wang has been an Associate Editor of 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 is Vice President—Technical Activities of the IEEE Computational Intelligence Society and he was Chair of the Emergent Technologies Technical Committee. He was Founding Chair of both the IEEE Engineering in Medicine and Biology Chapter Singapore and IEEE Computational Intelligence Chapter Singapore.

Haixiang Shi received the B.S. degree in electronic engineering from Xidian University, Xian, China, in 1998. He is currently working toward the Ph.D. degree from Nanyang Technological University, Singapore, Singapore.

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.