Solving optimization problems in communications using neural networks

Liu, Wen

2008

Liu, W. (2008). Solving optimization problems in communications using neural networks.
Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/18828>

<https://doi.org/10.32657/10356/18828>

Solving Optimization Problems in Communications Using Neural Networks

LIU Wen

School of Electrical & Electronic Engineering

A Thesis Submitted to the Nanyang Technological University
in Fulfillment of the Requirement for the Degree of
Doctorate of Philosophy

2008

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research done by me and has not been submitted for a higher degree to any other University or Institute.

2008.08.30

Date

刘文

LIU Wen

Acknowledgments

I am extremely grateful to my supervisor, Dr. WANG Lipo , without whom I may quit the Ph.D. studies in my first year, and without whom you will never see this thesis. The perfect balance that he achieved, between allowing me the freedom to pursue my own ideas and guiding me when needed, made my research work an enjoyable and interesting learning experience.

A thousand words of appreciation go to Dr. SHI Haixiang who is ever-willing to teach me the useful knowledge about his professional areas with an ever-ready smile.

I wish to thank my fellow students in Nanyang Technological University and friends at any corner of the world, who helped me in various ways.

I thank my parents, for their love.

Abbreviations

CNN	Cellular Neural Network
COP	Combinatorial Optimization Problems
CSA	Chaotic Simulated Annealing
CST	Constrained Steiner Tree
DCMR	Delay Constrained Multicast Routing
FAP	Frequency Assignment Problem
HNN	Hopfield Neural Network
NCNN	Noisy Chaotic Neural Network
NCNN-VT	Noisy Chaotic Neural Network with Variable Threshold
NHNN	Noisy Hopfield Neural Network
QoS	Quality of Service
SCSA	Stochastic Chaotic Simulated Annealing
SP	Shortest Path
TCNN	Transiently Chaotic Neural Network
TC-CNN	Transiently Chaotic Cellular Neural Network
TSP	Traveling Salesman Problem

Table of contents

Acknowledgements	ii
Abbreviations	iii
Abstract	viii
List of Figures	x
List of Tables	xiv
1 Introduction	1
1.1 Combinatorial optimization problems	1
1.1.1 QoS constrained multicast routing	2
1.1.2 Frequency assignment problems in satellite communications	7
1.2 Neural networks	8
1.2.1 Hopfield neural networks	10
1.2.2 Chaotic neural networks	11
1.2.3 Cellular neural networks	13
1.3 Chaos and Noise	14

1.3.1	Deterministic annealing and stochastic annealing	16
1.4	Motivation and contributions of the thesis	16
1.5	Organization of the thesis	18
2	A Noisy Hopfield Neural Network for the Shortest Path Routing	
	Problem	20
2.1	Introduction	20
2.2	The shortest path routing problem	23
2.2.1	Problem formulation	23
2.2.2	Energy function	24
2.3	A noisy Hopfield neural network	26
2.4	Simulation results	27
2.5	Conclusions	32
3	Noisy Chaotic Neural Networks for Delay Constrained Multicast	
	Routing	35
3.1	Introduction	35
3.2	Problem formulation	38
3.3	The NCNN for the CST problem	41
3.3.1	Noisy chaotic neural networks	41
3.3.2	Energy function	43
3.3.3	The structure of the neural network	47
3.4	Simulation results	48
3.5	Conclusions	59

4	Noisy Chaotic Neural Networks with Variable Thresholds for the Frequency Assignment Problem in Satellite Communications	63
4.1	Introduction	63
4.2	Satellite frequency assignments using the TCNN-VT	67
4.2.1	Transiently chaotic neural networks with variable thresholds .	68
4.2.2	Problem formulation	69
4.2.3	Simulation results and discussions	73
4.3	Noisy chaotic neural networks with variable thresholds	77
4.3.1	Objective mapping scheme	77
4.3.2	Application of the NCNN-VT to the FAP	83
4.3.3	Simulation results and discussions	86
4.4	Conclusions	99
5	Transiently Chaotic Cellular Neural Networks	102
5.1	Introduction	102
5.2	Cellular neural networks	106
5.3	Transiently chaotic cellular neural networks	109
5.4	Stability of the TC-CNN	114
5.5	Simulation results	116
5.5.1	Chaotic cellular neural network	116
5.5.2	Thresholds analysis of the C-CNN	117
5.5.3	Transiently chaotic dynamics of the single neuron model . . .	121
5.6	Conclusions	125

6	Conclusions	130
6.1	Outcomes of the thesis	130
6.2	Suggestions for future research	131
	Author’s Publications	134
	Bibliography	137
	Appendices	159

Summary

The development of communications engineering (Internet, satellites, mobiles) changes our daily life. The optimization problems in communications have motivated research in computational intelligence techniques these years.

To improve neural network algorithms for the shortest path routing problem (SPRP), we propose a solution approach using a noisy Hopfield neural network (NHNN) by adding decaying stochastic noise to the continuous Hopfield neural network (HNN). We also improve the energy function for the SPRP. Simulation results show that our approach offers further improvements on route optimality rate compared to other algorithms that employ the HNN.

We present algorithms to solve the delay constrained multicast routing problem by employing chaotic neural networks. Experimental results show that the noisy chaotic neural network (NCNN) offers further improvements, i.e., has a higher probability in reaching the global optimum, compared with the transiently chaotic neural network (TCNN) and the conventional HNN. Compared with the existing algorithm, i.e., the bounded shortest multicast algorithm (BSMA), NCNN is able to find delay constrained multicast trees with lower cost.

We explore the noisy chaotic neural network with variable thresholds for the frequency assignment problem (FAP). To solve the FAP in satellite communications, we extend the constant threshold in chaotic neural networks to variable thresholds. We let a neuron of the chaotic neural network to have a larger bias (the negative

of the neuronal threshold) when the neuron presents a frequency assignment with smaller interference, so that the neuron is more likely to be selected for a frequency assignment. With this mapping scheme, optimization objectives of the problem are achieved by variable thresholds of the neural network, while the energy function is in charge of only constraints. As a result, tuning of the weighting coefficients in the energy function becomes easier. The performance of the chaotic neural networks with variable thresholds is demonstrated through solving a set of benchmark problems and randomly generated testing instances. Simulation results show that the new model obtains better solutions with low computational cost compared to the previous methods.

A new model of cellular neural networks with transient chaos (TC-CNN) is proposed by adding negative self-feedbacks into cellular neural networks (CNNs) after transforming the dynamic equation to discrete time via Euler's method. The simulation on a single neuron model shows complex dynamics, i.e., stable fix points, bifurcation, and chaos. Hence this new CNN model has richer and more flexible dynamics, and therefore may possess powerful capabilities of solving various problems, compared to the conventional CNN with only stable dynamics. The TC-CNN may have a promising future in solving combinatorial optimization problems, as this model has complex dynamics so as to possess higher ability to search for optimal solutions. And furthermore, the local interconnection feature of TC-CNN makes it especially suited for large scale analog implementation.

List of Figures

1.1 Approaches for multicast routing: (a) The source-based approach;
(b) The core based approach. 3

3.1 A 16-node communication network for the delay constrained multicast
routing problem. 39

3.2 A comparison of dynamics of (a) the NCNN and (b) the TCNN. . . . 44

3.3 (a) A 50-node network used in our simulations, with an average degree
(the number of links for a node) 4; (b) The delay constrained optimal
multicast tree found by the NCNN for the 50-node network with one
source node (circle) and 5 destinations (black dots). 50

3.4 Network topology of the optimal delay constrained multicast route
for an eight-node communication network when the delay constraint
is 15. (a), (b), and (c), presents the solutions 1, 2, and 3 in Table 3.2,
respectively; (d) the output matrix for solution 2. 52

3.5 The optimal delay constrained multicast route when $\Delta = 20$ (a) The
optimal route; (b) the output matrix. 53

3.6 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 1000 runs. 55

3.7 Comparisons of the optimal route costs found by the chaotic neural network (NCNNs and TCNNs) and the BSMA for ten instances. The numbers 1-10 in the horizontal axis stand for the instances #1-#10 in Table 3.1, respectively. 56

4.1 The dynamics of the single-neuron NCNN with different values of thresholds I_0 . The X-axis is the time t and the Y-axis is the output of neuron $x(t)$ 79

4.2 Computation of cost matrix D from interference matrix E_I , extended from [104], which only showed figure (b). * denotes infinity. (a) The interference matrix and computation method. For carriers with carrier length larger than 1, the maximum value of the crossed elements on the red line represents the corresponding value in the cost matrix. (b) The cost matrix obtained from (a). 81

4.3 The neural network formulation for the FAP (extended from Figure. 4 of [104], which only consists of (a) and (b)). (a) The 24 neurons for the 4-carrier-6-segment FAP. (b) The convergence state of the neural network. (c) The full assignment of the carriers. (Carrier length $c_2 = c_4 = 2$) (d) The assignment of segments. 84

4.4 Computation time of the HopSA and NCNN-VT for the 20 instances. 95

5.1 (a)A 2-dimensional 4-by-4 cellular neural network: the nine shaded squares are the r-neighborhood cells of the black cell $C(i, j)$ when $r = 1$; (b) The piecewise-linear activation function of CNNs. 103

5.2 Block diagram of a chaotic cellular neural network. 112

5.3 Same template values can produce various dynamics with different self-feedback for C-CNN (a) stable dynamic with $z=0.7$; (b) period-2 oscillations with $z=2$; (c) period-3 oscillations with $z=2.15$; (d) chaotic dynamics with $z=5$ 117

5.4 Lyapunov exponents for different thresholds settings. For $I_0 > 0.8$, $\lambda = -0.0089$ (fixed points). For $I_0 < 0.8$, λ is mostly positive (chaos); however, there exist periodic windows with negative λ , for example, $I_0 = 0.27$ (period-3). 120

5.5 Neuron output y as a function of time t in the single-neuron chaotic cellular neural network, for different choices of I_0 (but fixed against time t). (a) $I_0 = 0.18$, chaotic output; (b) $I_0 = 0.27$, period-3 oscillations between 1, 1, and -1 ; (c) $I_0 = 0.8$, fixed point. 122

5.6 The internal state and output of a single-neuron chaotic cellular neural network with an exponential decaying I_0 (a) the internal state of neuron x ; (b) the output of neuron y ; (c) exponential decaying I_0 . . . 123

5.7 The one-dimensional mapping function from $x(t)$ to $x(t + 1)$ for the TC-CNN: (a) zoom out (b) zoom in. 124

5.8 Dynamics of the single neuron model (a) the output of the neuron;
(b) the self-feedback in network. 125

5.9 Dynamics of the single neuron model: the relationship between the
Lyapunove exponent and the self-feedback. 126

5.10 Dynamics of a multi-neuron CNN(a two-dimensional 3×3 network). 126

List of Tables

2.1	Performance comparison of different noise levels.	29
2.2	Performance comparison of different weighting constants.	29
2.3	Performance comparison of different μ_2 with μ_1 fixed at 200.	30
2.4	Performance comparison between Park and Choi, Ahn and our proposed approach.	30
2.5	Performance (Route opt %) comparison of different noise amplitude decreasing laws and rates for uniform noise.	31
2.6	Performance (Route opt %) comparison of different kinds Gaussian additive noise.	31
2.7	Performance comparison of discarding 0/1 term.	32
3.1	Specifications of the randomly generated geometric instances.	49
3.2	Comparison of three solutions when the delay constraint $\Delta = 15$. . .	53
3.3	Results for the HNN, TCNN, and NCNN for instances #1-#10. “sd” stands for “standard deviation”.	54
3.4	Performance comparison of the NCNN models with different noise levels for the networks of 16-node and 30-node. $\mu_1 = 150, \mu_4 = 200$. .	57

3.5 Performance comparison of the NCNN models with different weight-
ing constants μ_1 the networks of 16-node ($\mu_4 = 200, A[n(0)] = 0.1$)
and 30-node ($\mu_4 = 200, A[n(0)] = 0.02$). 58

3.6 Performance comparison of the NCNN models with different weight-
ing constants μ_4 for the networks of 16-node ($\mu_1 = 250, A[n(0)] = 0.1$)
and 30-node ($\mu_1 = 150, A[n(0)] = 0.02$). 58

3.7 Performance comparison of the NCNN models with different β_1 on the
8-node network with $\Delta = 20, \beta_2 = 0.0001$ and the 16-node network
with $\Delta = 20, \beta_2 = 0.0001$ 59

3.8 Performance comparison of the NCNN models with different β_2 on the
8-node network with $\Delta = 20, \beta_1 = 0.001$ and the 16-node network
with $\Delta = 20, \beta_1 = 0.001$ 60

4.1 Specifications of the FAP instances used in the simulation. 74

4.2 Weight coefficients $W_i, i = 1, \dots, 4$, of the energy function for all the
8 instances. W_1 is fixed at 1.0. 75

4.3 The performance of the TCNN-VT on 8 instances. # denotes the
instance number. I_L is the largest interference and I_T is the total
interference. The interference is shown as the best and average values
(Best/ Ave). “Opt rate” stands for the rate that the TCNN-VT
reached the optimum in the 1000 runs. T is the average number of
iteration steps. η is the convergence rate. “SD” stands for “standard
deviation”. 75

4.4 Comparison of simulation results (largest interference and total interference) obtained by the TCNN-VT, GNN and HopSA for instances 1 to 8. 77

4.5 Comparison between the HopSA and the TCNN-VT on computation time (Sec.). Results are displayed as mean±SD (standard deviation). 78

4.6 Specifications of the FAP instances used in the simulation. 88

4.7 Parameter setting of the NCNN-VT approach for different instances, with $k = 0.9$, $\epsilon = 1/250$, $\alpha = 0.015$, $z(0) = 0.08$, $\beta_1 = 0.001$, and $W_1 = 1.0$ 89

4.8 The performance of the NCNN-VT on 20 instances. “Opt rate” stands for the percentage at which the NCNN-VT reached the best known result in the 1000 runs. “SD” stands for “standard deviation”. 90

4.9 Comparison on the effect of the 0/1 term in the energy function for the FAP. 91

4.10 Comparison of largest interference and total interference obtained by the NCNN-VT, TCNN-VT, GNN and HopSA for benchmarks 1 to 8. Results are shown as the best, average values and standard deviation (Best/ Ave/ SD). 92

4.11 Comparisons of the largest interference and total interference on randomly generated instances between the HopSA [107] and the NCNN-VT. Results are displayed as mean±SD (standard deviation). 94

4.12 Comparison between the HopSA and the NCNN-VT on computation time in seconds and convergence rate. Results are displayed as mean $\pm$ SD (standard deviation). η stands for the convergence rate. . . 96

4.13 Comparison of the TCNN, NCNN, TCNN-VT and NCNN-VT on benchmarks BM 1-8 in 1000 runs. # denotes the instance number. η is the convergence rate. T is the average number of iteration steps. I_L is the largest interference and I_T is the total interference. The interference is shown as the best and average values (Best/ Ave). . . 98

Chapter 1

Introduction

1.1 Combinatorial optimization problems

Combinatorial optimization problems (COP) [1] are a branch of optimization problems where feasible solutions are discrete. Many problems in science and technology are related to the COP, which has an energy or cost function under some constraints with an equation or inequality. The objective is to develop efficient techniques for construction of possible optimal solutions which realize the minimum or maximum value of this function and satisfy the set of constraints with discrete variables [2–4].

COPs are very regular in computer science and artificial intelligence engineering, to name a few, the traveling salesman problem (TSP), the N-queen problem, the frequency assignment problem, and various routing problems. These problems are hard to deal with as they are NP-hard (nondeterministic polynomial). Two basic strategies are “divide-and-conquer” and iterative improvement. With inherent parallel computational architecture and fault tolerance, the development of neural networks have offered attractive techniques for COPs and received considerable popularity as a tool for COPs. A well known fact is that trapping at local minima of

the cost function is the main challenge for efficient operation of COPs, especially when the problem size is large.

Many researchers developed neural network techniques to jump out of the local minima and search for global optima. In this thesis, we studied the effectiveness of stochastic dynamics, chaotic dynamics in neural networks and their application in COPs within communications domain.

1.1.1 QoS constrained multicast routing

The multicast routing problem is also called Steiner tree problem [5], which aims to minimize the total cost of a multicast tree, and is known to be NP-complete (non-deterministic polynomial time complete). If the multicast group includes all nodes in the communication network, the Steiner tree problem reduces to the minimum spanning tree problem. Wang and Hou [6] gave a deliberate review on the problem formulation of multicast routing together with the multicast routing algorithms and protocols.

Multicast service is a network-layer function which has been increasingly used by various continuous media applications. In a communication network, data generated by the source flows through the multicast tree, traversing each tree edge exactly once. As a result, multicast is more resource-efficient than conventional unicast routing or the broadcast routing protocol, and is well suited to applications such as video distribution and teleconferencing [6]. The multicast problem is particularly important for the wireless network which suffers from resource (bandwidth) scarcity

Figure 1.1: Approaches for multicast routing: (a) The source-based approach; (b) The core based approach.

and high bit error rate [7].

The approaches for constructing multicast trees as shown in Figure 1.1 can be classified into two categories [8]: (1) the source-based multicast tree approach; (2) the core-based multicast tree approach. Source-based approach build a tree beginning at the source first, then add edges and nodes to the tree at each iteration. Topology information is gathered and processed at the source. In the core-based approach one node is selected as the core for each group. A tree rooted at the core is then constructed to span to all the group members. A node forwards packets to its parent and children except the one where data packets came from. However, core-based tree methods may lead to side effects, e.g., network congestion on some links that are near to the “core” as data for different receivers will have to traverse these edges. Another problem of using core-based algorithms is the large delay variations.

Real-time communication networks are designed mainly to support multimedia applications, especially those interactive ones, which require a guarantee of quality

of service (QoS) [9]. Moreover, multicast routing is required as there are usually more than two peers who communicate together using multimedia applications. As for the routing, multicast is a critical issue in packet-switched networks due to its significant impact on the network performance [10]. Network has to find an optimal multicast route, that has enough resources to provide guarantee for the required QoS. This problem is called QoS constrained multicast routing and was proved to be a NP-complete problem.

For example, the objective of delay constrained multicast routing (DCMR), which we are going to solve in Chapter 3, is to find a tree rooted at the source S and spanning to all the members of destination group D such that: (1) The total cost on the links of the tree is minimum; (2) The delay from source to each destination is not greater than the required delay constraint.

Constraints of the routing problem in communication networks can be classified into two categories [6]:

1. Link constraints: restrictions on the use of links for route selection;
2. Tree constraints:
 - Bounds on the combined value of a performance metric along each individual path from the source to a destination, e.g. the end-to-end bound.
 - Bounds on the difference of the combined value of a performance metric along the paths from the same source to any two different destinations.

In recent years, the integrated services, differentiated services, and multi-protocol

label switching improved the Internet to support QoS requirements [11]. The integrated services model is characterized by resource reservation. For real-time applications, before data are transmitted, the applications must first set up paths and reserve resources. Resource reservation setup protocol is a signaling protocol for setting up paths and reserving resources [12]. The efficient management of the network resources has many merits, such as the reduction of the cost for the network service and permission for more applications to operate simultaneously [13]. The integrated services model proposes two service classes in addition to best effort service. They are: 1) *Guaranteed Service* for applications requiring fixed delay bound; 2) *Controlled Load Service* for applications requiring reliable and enhanced best effort service.

In differentiated services, packets are marked differently to create several packet classes. Packets in different classes receive different services. Constraint based routing extends QoS routing by considering other constraints of the network such as policy. The goals of constraints based routing are to select routes that can meet certain QoS requirements and to increase the utilization of the communication network [12].

Constraints based routing considers not only topology of the network, but also the requirement of the flow, the resource availability of the links, and possibly other policies specified by the network administrators.

Common route metrics include cost, hop-count, bandwidth, reliability, delay and jitter. These can be divided into three classes. Let $x(i, j)$ be a metric for link (i, j) .

For any path $P = (i, j, k, \dots, l, m)$, metric x is considered:

$$\text{Additive if} \quad x(P) = x(i, j) + x(j, k) + \dots + x(l, m)$$

$$\text{Multiplicative if} \quad x(P) = x(i, j) * x(j, k) * \dots * x(l, m)$$

$$\text{Concave if} \quad x(P) = \min\{x(i, j), x(j, k), \dots, x(l, m)\}$$

According to this definition, metrics delay, jitter, cost and hop-count are additive, reliability is multiplicative, and bandwidth is concave [12].

Computing optimal routes while satisfy constraints of two or more additive and/or multiplicative metrics is NP-complete [14]. The proof is based on the assumptions that all the metrics are independent and the delay and jitter of a link are known a priori. However, individual QoS parameter may be conflicting and interdependent, thus making the problem even more challenging [7]. If we intend to optimize a multicast path with respect to a single QoS parameter, say bandwidth, then the problem can be solved in polynomial time even with uncertain network resources [15], by mapping it to a shortest path finding problem. For multicast routing, it is more difficult to solve when each destination has different QoS requirements [9]. Furthermore, routers should be made QoS aware as the conventional IP routing is based on hop counts and not suitable for multimedia applications [16].

In [17] and [18], some neural network applications in high speed communication networks are dealt with for optimal routing. Venkataram, Ghosal and Kumar [10] summarized some requirements when solving QoS problems:

1. The computation should be carried out in real time;

2. The algorithm should adapt to the changes in the link costs;
3. The overhead should be as low as possible;
4. The algorithm should adapt to the changes in the network topology.

Some heuristics have been developed to construct low-cost trees with a bound on the end-to-end delay, and this is what we are going to focus on in this thesis. Salama et al [19] carried out a survey with extensive simulation study of several existing multicast algorithms. An evaluation of the performances of those algorithms on a high-speed system is presented.

1.1.2 Frequency assignment problems in satellite communications

Frequency assignment problems (FAP) [20–23] appear in various areas, including mobile telecommunications, broadcasting, military operations, and satellite communications. In our work, we focus on minimization of system interference for fixed frequency assignments in satellite communications. Interference in satellite communications depends on transmitter power, channel loss, receiver sensitivity, and antenna gains.

Frequency rearrangements are an effective complement alongside with the reduction of interference itself. Hence in order to deal with interference, rearrangements of frequency assignments are considered as an effective way. Early efforts have focused on various analytical methods for evaluations of cochannel interference [24, 25] and

very few systematic methods have been adopted to optimize frequency assignments to reduce cochannel interference. The later work of Muzuike and Ito [26] revealed the importance of mathematical models for reduction of cochannel interference. They proposed a basic mathematical model to formulate the cochannel interference reduction problem as an “assignment problem”. The assignment problem aims to minimize the largest and accumulated interference among carriers.

In the FAP, carrier frequencies for one set of carriers are to be rearranged while keeping the other set fixed, i.e., frequencies for carriers in system 2 are rearranged while frequencies for system 1 are fixed. The communications are assumed to operate between F_a and F_b , where F_a and F_b are frequency bands. The cochannel interference can be evaluated by considering each pair of carriers under the same frequency, which varies with carrier pairs. The objective of this assignment problem is to find the optimal assignment of frequencies in system 2 in order to reduce the cochannel interference.

In Chapter 4, we solve the FAP in satellite communications through the TCNN-VT and NCNN-VT, and simulation results show that the performance of these neural network methods is comparative with existing heuristics.

1.2 Neural networks

Since we are using neural networks, this section will focus on the characteristic and development of the neural network. In the development history of the modern computation technology, bio-inspired neural networks [27], due to their intrinsic op-

eration functions, have played a very important role. Derived from some aspects of neurobiology and adapted to integrated circuits, neural networks have been used widely in various fields such as optimization, linear and nonlinear programming, associative memory, pattern recognition, feature extraction [28–30] and so on. Regarded as one prominent information processing technique, neural networks have tremendous impact on many disciplines that require challenging information processing techniques.

Neural networks are famous with following merits:

- fast processing through massive parallelism;
- learning and adapting ability;
- robustness.

Kohonen [31] defined an artificial neural network as “*a massively parallel interconnected networks of simple (usually adaptive) elements and their hierarchical organizations which are intended to interact with the objects of the real world in the same way as biological nervous systems do*”. In general, there are three fundamentally different classes of network architectures: single-layer feedforward networks, multi-layer feedforward networks, and recurrent neural networks. By defining proper processing functions for each node and defining associated weights for each interconnection, the neural network is capable of solving optimization problems relatively rapidly [32]. The adjustable weight connections correspond to synapses in biological neural system.

1.2.1 Hopfield neural networks

The Hopfield neural network (HNN) [33] proposed in 1982 consists of a set of neurons and a corresponding set of unit delays, forming a multiple-loop feedback system. The neurodynamics of a continuous mode HNN based on the additive model is defined in the form:

$$C_j \frac{du_j(t)}{dt} = -\frac{u_j(t)}{R_j} + \sum_{i=1}^N w_{ji} \phi_i[u_i(t)] + I_j \quad (1.1)$$

$$v_i = \phi_i(u_i) = \frac{1}{1 + e^{-u_i/\epsilon}} \quad (1.2)$$

where

u_j = Internal state of neuron j ,

v_j = Output of neuron j ,

w_{ji} = Synaptic weights from neuron j to neuron i ,

I_j = An independent voltage source,

C_j = A linear capacitor,

R_j = A linear resistor,

ϵ = A parameter which controls the shape of activation function.

The HNN operating in a discrete mode is based on the McCulloch-Pitts model [34].

The HNN has three major forms of parallel organization found in neural systems: parallel input channels, parallel output channels, and a large amount of inter-

connectivity between the neural processing elements [10].

1.2.2 Chaotic neural networks

Since Hopfield and Tank's seminal work on solving the TSP with a HNN [35], the HNN has been recognized as a powerful tool for optimization problem. However, these attempts at solving various optimization problems with the HNN suffered from the fact that a HNN often becomes trapped at a local minimum as discussed before. The employment of HNN thus has been hampered by problems with solution quality and slow development of suitable hardware to solve the large sized problems.

Stochastic simulated annealing (SSA) [36], which allows for temporary energy increases, has been incorporated into the HNN, in order to improve the solution quality. Nozawa [37,38] modified the Hopfield network model with a negative self-feedback connection and obtained a neural network model as the globally coupled map (GCM). The solution of the TSP showed that chaotic search by the GCM model is efficient in approaching the global optimum or sub-optima. Chen and Aihara [2] proposed chaotic simulate annealing (CSA) by adding a negative self-feedback to a Hopfield-type neural network and gradually removing this negative self-feedback. This network leads to remarkable improvements over the HNN in terms of solution quality of optimization problems. Using chaotic dynamics to solve COPs has been studied widely after the work of Aihara et al [39], some found that chaotic dynamics is more effective for solving optimization problems than stochastic dynamics [40]. However, the study still continues for further progress or alternative

technique to address the issue of solution quality: particularly improvements which can be implemented easily in hardware [41].

Many reports have proved that the chaotic dynamics can be promising techniques for optimization problems. Chen and Aihara obtained the chaotic neural network with richer and far-from equilibrium dynamics [39, 42]:

$$u_i(t+1) = ku_i(t) + \alpha \left[\sum_{j=1}^N w_{ji} v_j(t) + I_i \right] - z_i(t) v_i(t) \quad (1.3)$$

Chen and Aihara [2] then solved the convergence problem of chaotic neural networks by removing the negative self-feedback gradually, thereby obtained remarkable improvement over other neural-network-based methods, in terms of the frequency of finding the optimal or near-optimal solutions. They proposed a transiently chaotic neural network (TCNN) by introducing a decaying negative self-feedback. The dynamics of the new model are characterized as transient chaos. Numerical experiments on the TSP and the maintenance scheduling problem showed that the TCNN has high efficiency to converge to globally optimal solutions.

The TCNN, which is also known as chaotic simulated annealing, is not a problem-specific but a powerful general method for COPs [43]. With a decaying self-feedback connection, TCNNs are more effective in solving COPs compared to the HNN. He *et al.* [40] used the TCNN combined with a multistage self-organizing algorithm to solve the cellular channel assignment problem where the TCNN is in charge of searching for the optimal assignment.

Wang and Smith [41] suggested an alternative approach to chaotic simulated an-

nealing by reducing the time-step rather than the self-feedback and generalized the stability theorems to less restrictive and more compact forms. Kwok and Smith [44] presented a unifying framework which encompasses three main model types, and simulated the new framework on the N-queen problem to show the effect of the chaotic neural dynamics. Wang et al [45] combined the SSA and CSA, and proposed a new approach, i.e., stochastic chaotic simulated annealing (SCSA) for solving optimization problems. The simulation results on a TSP and a channel assignment problem demonstrated the effectiveness of SCSA. These advances have made modified HNN competitive with the best heuristics for solving combinatorial optimization problems.

When simulated in serial computers, such as a desk-top personal computer, neural networks are known to be slow and rather ineffective in solving optimization problems in comparisons with classical methods, e.g., branch-and-bound technique and tabu search. The advantages of neural networks can be truly realized only when the parallel processing capability of neural networks is used, e.g., when neural networks are implemented in hardware [46].

1.2.3 Cellular neural networks

The cellular neural network (CNN) proposed by L. Chua [47, 48] is well known because of its continuous time feature and local interconnection feature. Compared with general neural networks, CNNs are much more amenable to very large scale integration (VLSI) implementation according to its neighbor interactive property.

Applications of the CNN in signal processing and pattern recognition have prominent success. Some researchers [49, 50] presented the bifurcation and chaotic dynamics in discrete-time CNNs using locally coupled cells and logistic equations as activation functions. It has been approved that the chaotic dynamics is good for information processing and optimization. In order to take advantage of both efficient search with chaos and the mature technology of VLSI implementation of CNNs, we proposed a transiently chaotic cellular neural network. We will introduce this part in Chapter 5.

1.3 Chaos and Noise

Both chaotic dynamics (a result of deterministic simulated annealing) and stochastic noise (a result of stochastic simulated annealing) have gained extra attention from scientists of different areas. Aihara [51] reviewed chaos engineering from various aspects, including system models, applications, and hardware implementations of chaotic neural networks. Delgado-Restituto and Rodriguez-Vazquez [52] surveyed design techniques of integrated chaos generators both at system and circuit levels.

Chaotic dynamics [53] are complex behaviors which can be generated by a finite set of deterministic nonlinear equations with a simple system. Chaos is globally stable and locally unstable [54]. A lot of research these years is focus on developing techniques to harness chaos when it is undesirable or to generate chaos so that the useful function of a chaotic system can be utilized. Chua proposed a nonlinear electronic circuit which has become a universal paradigm for chaos [55]. Hayakawa [56] emphasized the effects of chaos in neural network dynamic through a simple model

for the TSP. Aihara [51] discussed chaotic dynamics from an engineering point of view. The paper recounted the development and prospect of chaos engineering. He [43] added decaying chaotic noise which was generated by the logistic map to the discrete-time continuous-output HNN. The simulation on the TSP showed improvement in terms of searching ability and iteration steps.

Chaos has been widely investigated by not only mathematicians and physicists, but also engineers, economists, and scientists from various discipline. Chaotic dynamics have several special characteristics, such as:

1. a sensitivity to initial conditions,
2. determinism as the system function is well defined,
3. long term unpredictability.

Bucolo et al [57] investigated the effect of chaos on helping order to arise from disorder and brought forward the question *“does chaos work better than noise?”* In their work, effects of chaos and random signals 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 [58] employed stochastic noise to enhance the HNN. The experiment results on the image classification problem confirmed the expected improvement. The noisy chaotic neural network (NCNN) [45,59] model has both decaying noisy and chaotic dynamics, which helps to improve the ability of the network model to reach global optima. Hence we are going to further explore the effectiveness of the NCNN on COPs within communications confine.

1.3.1 Deterministic annealing and stochastic annealing

Simulated annealing is a kind of optimization techniques, whose convergence has been proved. Stochastic simulated annealing (SSA) [36] is developed by Kirkpatrick et al with Monte Carlo technique [60]. To avoid getting stuck in local minima, the SSA emulates the metal annealing process by first heating the solid to its melting point and then cooling at a natural rate related to the heat transmission speed of the metal. The SSA has been widely applied to various optimization problems. However, it suffers from the following problems: 1) subtle adjustment of parameters in the annealing schedule due to the stochastic mechanics; and 2) time consuming due to Monte Carlo scheme.

On the other hand, the deterministic simulated annealing (DSA) approaches have also been proposed and applied to neural networks to conquer the local minimum problem. For example, gradient method, gain sharpening of the Hopfield networks [61,62] and mean field approximation annealing [63].

1.4 Motivation and contributions of the thesis

In this thesis, we are concerned with evaluating the potential of neural networks for solving combinatorial optimization problems.

We analyze the effect of stochastic noise in the HNN for the shortest path routing problem (SPRP) and solve delay constrained multicast routing problem with the NCNN. Then we extend the neural network by varying the thresholds with the optimization objective of the problem. The simulation of variable thresholds neural

networks on the frequency assignment problem in satellite communications is successful. We also propose a transiently chaotic cellular neural network (TC-CNN), which shares the best feature of both world: it has complex dynamics so possesses higher ability to search for global optimal solutions for optimization problems; And furthermore, the local interconnection feature of cellular neural networks (CNNs) made it especially suited for large scale analog implementation.

While details of any hardware implementation are beyond the scope of this thesis, our aim is to provide an indication via simulation of the solution quality which could be expected from neural networks.

In summary, the main contributions of the thesis are:

- Evaluate the effect of stochastic noise in the HNN system;
- Solve the shortest path problem with the noisy Hopfield neural network (NHNN) and the delay constrained optimal multicast routing problem with the NCNN;
- Explore the design of thresholds in the chaotic neural networks to separate optimization term from the energy function thus improving solution quality;
- Demonstrate how the delay constrained multicast routing problem and the frequency assignment problem are mapped from standard form onto chaotic neural networks;
- Proposal of a new transiently chaotic cellular neural network together with convergence and stability properties analysis through a dynamical systematic perspective.

1.5 Organization of the thesis

The broad outline of this thesis is as follows:

Chapter 2 we use a NHNN to improve HNN's ability to reach the global optimal solution, while retaining the merits of Ali and Kamoun's model, i.e., model simplicity and the ability to adapt to changes in the communication network topology, the source or the destination node, and link costs. We also evaluate the effect of the stochastic noise in the HNN model and investigate the sensitivity of the model to the value of weighting constants. Adding the decaying noise enhances the route optimal rate for the SPRP, compared with other heuristics based on the HNN.

Chapter 3 By the TCNN and the NCNN, we solve the delay constrained optimal multicast routing problem in communications network. The NCNN has higher probability in finding the global optimum compared to the TCNN and the original HNN model. The cost of the multicast tree found by the chaotic neural network is equal to if not less than the one obtained by existing algorithms. Results from the COPT algorithm prove that chaotic neural networks, i.e., TCNNs and NCNNs, are able to find the global optima.

Chapter 4 We further explore chaotic neural networks with variable thresholds, which separate the optimization term from the constraints terms in the cost function by assigning different neurons with variable thresholds. The neural networks with variable thresholds offer further improvements, i.e., achieve

smaller interference with much lower computation cost compared to existing algorithms.

Chapter 5 We add negative self-feedbacks into the Euler approximation of the continuous CNN model and demonstrate there are a variety of dynamic behaviors, including fixed points, periodic oscillations, and chaos, thereby creating a new *chaotic cellular neural network* (C-CNN). Then we extend C-CNN to a *transiently chaotic cellular neural network* (TC-CNN) by decaying the self-feedback. Compared to existing chaotic CNNs, our model uses conventional neurons and is relatively easy to analyze mathematically when the network is large.

Chapter 6 closes the thesis by presenting our main conclusions, and outlining some suggestions for future research.

Chapter 2

A Noisy Hopfield Neural Network for the Shortest Path Routing Problem

2.1 Introduction

The optimization problem of finding the shortest path (SP) from a single source to a single destination in a graph arises as a subproblem to many problems, including routing problems in computer networks. Stochastic search is an efficient method for such optimization problems. Although stochastic search algorithm does not guarantee the best solution, it tries to find a good solution. Starting from a random point, stochastic search algorithm gradually improves the solution by making changes. To list a few, there are Tabu search, ant colony, swam algorithm, simulated annealing and etc. There are some well-known polynomial algorithms for the SP problem, namely Bellman-Ford's [64, 65] or Dijkstra's [66]. The Bellman-Ford algorithm is able to deal with negative edge costs but has a longer running time compared with

Dijkstra's. Dijkstra's algorithm can only process non-negative edge cost. It outputs a shortest path tree as it finds the route with lowest cost between the source node and every other nodes. Dijkstra's is used for the SP problem by stopping the algorithm once the shortest path to the destination node has been determined.

Genetic algorithm [67, 68] is based on basic principles of evolution. Firstly a number of chromosomes will be generated randomly, then GA will decode each chromosome to evaluate its fitness and perform *selection, cross-over and mutation* operations. Ant algorithm is proposed since studies showed that ants are capable to find the shortest path from their nest to the food source. Lu and Liu [69] have formulated communication networks with the theory of ant algorithm. Another popular tool for optimization problem is simulated annealing [36]. Different with evolutionary algorithm, simulated annealing is physics oriented and requires less memory.

The use of neural networks to find the shortest path between a given source-destination pair was first introduced by Rauch and Winglike [32]. An adaptive algorithm with many good features (eg. scaling properties and an ability to operate in real time) for the network structure is introduced in [70] with topological information. They formulated the problem onto a HNN and simulated the algorithm on a 5-node network with satisfactory results. But the algorithm often fails to converge when the number of nodes in the graph increases.

The SP problem can extend to unicast routing problem. In general, routing involves two entities: routing protocols and routing algorithms. Routing protocols

capture the network state information such as available resources and disseminate it throughout the network, while routing algorithms use this information to compute appropriate paths. While the best effort routing simply performs these tasks based on a single, relatively static measure, quality of service (QoS) routing takes into account both the application's requirements and the availability of network resources [71].

For the shortest path routing problem (SPRP) [72–74], Park and Choi [72] introduced a new term into the energy function proposed by Ali and Kamoun [70] so that the flow is one-directional to the destination, but at the same time this term made connection weights of the neural network dependent on the topology of the communication network. Araujo et al [73] proposed a new formulation and extended the HNN to a two-layer architecture, in which the number of neurons is equal to the number of arcs in the graph, so the computation time is reduced. By this method, they enhanced reliability but impaired the ability to reach optima. Ahn et al [74] added two new terms into Park-Choi energy function to avoid possible loops. Although the simulation showed improvements over other approaches, its energy function has seven terms and it is hard to tune the associated seven parameters.

A number of authors have explored adding noise into the recurrent neural network in order to improve network performance. Jim et al [75] presented a detailed investigation on the effects of injecting synaptic noise to recurrent neural networks and analyzed several injection methods. They added different kinds of noise onto synaptic connection weights of the neural network and showed several benefits, such

as improvements on generalization and convergence. Das and Olurotimi [76,77] analyzed noisy recurrent neural networks in both continuous and discrete cases. They showed that knowledge about the behavior of neural networks in the presence of noise would be helpful in design and parameter selection. The noisy chaotic neural network [45] which we are going to introduce in Chapter 3 also achieved encouraging simulation results in the traveling salesman problem and the channel assignment problem.

In this chapter we use a noisy Hopfield neural network (NHNN) to improve HNN's ability to reach the global optimal solution, while retaining the merits of Ali and Kamoun's model, i.e., model simplicity and the ability to adapt to changes in the communication network topology, the source or the destination node, and link costs. This analysis is necessary as thermal noise is inherent in analog implementations, especially in very large scale integration (VLSI) implementations.

2.2 The shortest path routing problem

2.2.1 Problem formulation

The SPRP aims to find a path in a communication network from source node s to destination node d such that the total cost on the path is minimum. Here, we use the formulation proposed in [70].

Considering a communication network with n nodes, the neural network is arranged on an $n \times n$ matrix, with all diagonal elements removed and every element

of the matrix is treated as a neuron. The neuron of row x and column i describes the link from node x to node i in the communication network. P_{xi} characterizes the connection topology of the communication network: if the arc from node x to node i does not exist, $P_{xi} = 1$; otherwise, $P_{xi} = 0$.

The routing solution is described by the final output V_{xi} of neuron (x, i) as follows:

$$V_{xi} = \begin{cases} 1, & \text{the arc from node } x \text{ to node } i \text{ is on the final path;} \\ 0, & \text{otherwise.} \end{cases}$$

The cost of an arc from node x to node i is denoted by C_{xi} , which is a real non-negative number. For non-existing arcs, $C_{xi} = 0$.

2.2.2 Energy function

We modify the energy function based on the one proposed by [70] and [72].

$$\begin{aligned} E = & \frac{\mu_1}{2} \sum_{x=1}^n \sum_{i=1, i \neq x}^n C_{xi} V_{xi} + \frac{\mu_2}{2} \left[\sum_{x=1}^n \left(\sum_{i=1, i \neq x}^n V_{xi} - \sum_{i=1, i \neq x}^n V_{ix} - \gamma_x \right)^2 \right. \\ & \left. + \sum_{x=1}^n \sum_{i=1, i \neq x}^n P_{xi} V_{xi} \right] \end{aligned} \quad (2.1)$$

where

$$\gamma_x = \begin{cases} 1, & \text{if } x = s; \\ -1, & \text{if } x = d; \\ 0, & \text{if } x \neq s, d. \end{cases}$$

Hence γ_x equals to 1 when $x = s$, equals to -1 when $x = d$, equals to 0 if others. Here $\{\mu_i, i = 1, 2\}$ are the weighting constants. The μ_1 term aims at minimizing the cost on the route. The μ_2 term is the constraints with two parts. The first part is used to satisfy the requirement that for each single node except the source and the destination, the number of incoming links is equal to the number of outgoing links. The second part penalizes neurons that represent non-existing links in the communication network. In our energy function formulation, we treat various constraints as soft, i.e, weight each constraint according to its priority and absolute value. We use the soft penalty term to suppress the non-existing links and to guarantee that the result is feasible. Compared to hard constraints technique, which impose a global requirement on the dynamic system, the use of soft constraints is easier and much more popular. There are only two parameters need to be balanced in this model, which makes tuning of parameters more efficient.

We have discarded the fourth term of the energy function used in [70] which forces the outputs of the neural network to approach either 0 or 1. This term is not necessary as other constraints terms can achieve the same objective. Although this 0 or 1 term can help the neural network converge, it also increases the difficulty to tune the weighting coefficients. The third and fifth terms in [70] are combined here as in [72], and furthermore, we have combined all the constraint terms so that there are only two parameters in the energy function to tune.

2.3 A noisy Hopfield neural network

A noisy Hopfield neural network (NHNN) is obtained by adding decaying stochastic noise into the continuous HNN [33]:

$$\frac{dU_{xi}(t)}{dt} = -\frac{1}{\tau}U_{xi}(t) + \sum_{y=1}^N \sum_{j=1, j \neq y}^N w_{(yj \rightarrow xi)} V_{yj}(t) + I_{xi} + n(t) \quad (2.2)$$

$$V_{xi} = f_{xi}(U_{xi}) = \frac{1}{1 + e^{-U_{xi}/\epsilon_{xi}}} \quad (2.3)$$

where U_{xi} is internal state of neuron xi . V_{xi} is output of neuron xi as described in section 2.2.1. $w_{(yj \rightarrow xi)}$ is the weight on the connection from neuron yj to neuron xi . τ is a circuit time constant. I_{xi} is the input bias of neuron (x, i) . $f_{xi}()$ is the activity function of the neuron. ϵ_{xi} ($\epsilon \geq 0$) is the steepness parameter of the neuronal output function and $n(t)$ is the random noise.

We consider random noise $n(t)$ with uniform and Gaussian distributions. For uniform distribution, $n(t)$ is uniformly chosen from the range $(-A[n], A[n])$, where $A[n]$ is the noise amplitude. To investigate the effect of different decaying schemes of the noise amplitude, we will use exponential decaying and linear decaying. For exponential decaying, $A[n(t+1)] = (1 - \beta)A[n(t)]$, β ($0 \leq \beta \leq 1$) is the damping factor. For linear decaying, $A[n(t)] = A[n(0)] - \kappa t$, where κ is the slope of linear decaying. As for Gaussian distribution, random noise $n(t)$ follows Gaussian distribution with mean 0 and initial variance $\sigma^2(0) = 1$. We reduce the variance by $\sigma^2(t+1) = (1 - \beta_g)\sigma^2(t)$. Hence as time goes on, the chance to generate a large

noise value decreases.

To assure that the neural network is convergent, i.e., the energy function will decrease monotonously as neurons updating after the disappear of noise, the connection weights $w_{(yj \rightarrow xi)}$ and bias terms I_{xi} are derived from the equation bellow [33]:

$$-\frac{\partial E}{\partial V_{xi}} = \sum_{y=1}^N \sum_{j=1, j \neq y}^N w_{(yj \rightarrow xi)} V_{yj}(t) + I_{xi} \quad (2.4)$$

Substitute the energy function in Eqn. 2.4 with Eqn. 2.1:

$$w_{yj,xi} = -\mu_2[\delta_{xy} - \delta_{xj} + \delta_{ij} - \delta_{iy}] \quad (2.5)$$

$$I_{xi} = -\frac{\mu_1}{2}C_{xi}(1 - \delta_{xm}\delta_{is}) + \frac{\mu_2}{2}\delta_{xm}\delta_{is} - \frac{\mu_2}{2}P_{xi}(1 - \delta_{xm}\delta_{is}) \quad (2.6)$$

where δ_{ij} is the Kronecker delta. No terms in the connection matrix depend on the cost of links or the topology of the communication network. Thus the connection matrix is independent of any changes in the communication network. The cost term is mapped into the bias. The benefit is immense because there is no need to adjust internal parameters of the neural network to adapt to changes in the environment, therefore the hardware implementation of the neural network is a general one [70].

2.4 Simulation results

Firstly, we run the algorithm 10000 times on 20-node networks which are randomly generated with a Linux cluster (dual Xeon 3.06 GHz, Intel IA32). The cost of an arc from node x to node i , i.e., C_{xi} , is also randomly generated and normalized.

As solving the SP problems using HNNs in [72], the parameters in our model are chosen as follows: $\epsilon_{xi} = \epsilon = 1$, independent of neuron location (x, i) . The circuit constant $\tau = 1$ and the time step $\Delta t = 10^{-4}$. Initial inputs of the neural network $U_{xi}(0)$ are randomly generated between $[-0.001, 0.001]$. At the end of each iteration, we set each neuron on or off according to its output value. If $V_{xi} \geq 0.5$, the neuron is on, i.e., $V_{xi} = 1$, which means the link from x to i is chosen in the final optimal tree, and vice versa.

We compare the performance of NHNNs with different initial noise amplitudes (from 0.0 to 5.0) in Table 2.1. The damping factor β is set to 0.001. According to [70, 74] and our experience on the routing problem, the weighting constants are initially set to $\mu_1 = 300$ and $\mu_2 = 2500$. The setting of those weighting coefficients is problem-specific and is carried out by trial-and-error. However, there is still a guideline. Here, the route optimal rate denotes the ratio at which the algorithm reached the optimal solution in 10000 runs, in comparison with Dijkstra's results. Adding noise helps the neural network reaching the global optima: without noise, the optimal rate is 71.46%; when the initial noise amplitude $A[n(0)] = 2.0$, the optimal rate is 92.73%, with more than 20% improvement. However, if the noise amplitude is too high (large than 3.0), the route optimal rate starts to decrease. The algorithm fails to find a valid solution if the algorithm does not converge to the optimal solution, as we are using soft constraints, where penalty terms are added to the objective function directly.

We also investigate the sensitivity of the model to weighting coefficients in Ta-

Table 2.1: Performance comparison of different noise levels.

Noise level		0.0	0.5	1.0	2.0	3.0	5.0
Time (s)	mean	1.57	2.72	2.87	3.05	3.14	3.25
	SD	0.81	0.83	0.82	0.77	0.72	0.69
Route opt %		71.46	82.64	89.42	92.73	90.89	84.25

Table 2.2: Performance comparison of different weighting constants.

μ_1		100	150	200	250	300
Time (s)	mean	2.77	2.83	2.92	3.02	3.05
	SD	0.70	0.75	0.76	0.75	0.77
Route opt %		88.02	94.49	95.32	93.68	92.73

ble 2.2 while setting the initial noise amplitude level at 2.0. According to [72], we vary μ_1 from 100 to 300 while fixing μ_2 at 2500. The setting of those weighting coefficients is problem-specific and is carried out by trial-and-error. We found that when μ_2 is fixed at 2500 the model performs the best at $\mu_1 = 200$ and the route optimal rate reaches 95.32%. As there are only two weighting constants, it is not difficult to find the proper weighting constants by trial-and-error.

Similarly, we also analyze the effect of weighting coefficient μ_2 in Table 2.3 by varying μ_2 from 1500 to 3500 while fixing μ_1 at 200. From Tables 2.2 and 2.3, we can see that the performance of the NHNN is not sensitive to the setting of connection weights. Therefore the tuning of these two parameters for a good performance is not difficult.

We further investigate NHNNs for the SPRP on networks with 30, 50, and 80 nodes. The parameters are set as follows: $A[n(0)] = 2.0$, $\mu_1 = 200$, $\mu_2 = 3000$. The

Table 2.3: Performance comparison of different μ_2 with μ_1 fixed at 200.

μ_2	1500	2000	2500	3000	3500
Route opt %	89.12	92.81	95.32	96.18	95.04

Table 2.4: Performance comparison between Park and Choi, Ahn and our proposed approach.

Network size	20	30	50	80
Park and Choi [72]	72.14	67.35	50.18	41.12
Ahn [74]	78.72	71.16	62.40	50.77
NHNN	96.18	88.37	79.91	62.90

Performance of the proposed NHNN and those of Park and Choi [72] and Ahn [74] are compared in Table 2.4. The NHNN reaches global optima more effectively.

To know more about the effect of the additive noise in the HNN, for the uniformly distributed noise, we have done a comparison of performance brought by noise with different decaying laws (exponential and linear) and different decaying rates. Results are shown in Table 2.5. μ_1 and μ_2 are set to 200 and 3000, respectively. It is evident that exponential decay works better than linear decay in terms of the probability to reach optima and parameter sensitivity.

Also, we simulate NHNNs using random noise with Gaussian distribution. When using Gaussian noise, we gradually decrease the variance of the distribution, which is different from the case with uniform noise distribution. Hence the probability to have a small value noise increases as iteration goes on. As shown in Table 2.6 and Table 2.5, Gaussian noise is comparable to uniform noise in performances.

To justify our decision on discarding the 0/1 term, we simulated the NCNN with the 0/1 term $\frac{\mu_c}{2} \sum_{x=1}^n \sum_{i=1}^n V_{xi}(1 - V_{xi})$ in the energy function and compared results

Table 2.5: Performance (Route opt %) comparison of different noise amplitude decreasing laws and rates for uniform noise.

Network size			20	30	50	80
uniform	exponential	$\beta = 0.01$	91.17	83.87	75.42	54.76
		$\beta = 0.001$	96.18	88.37	79.91	62.90
		$\beta = 0.0001$	71.73	63.90	54.93	48.55
	linear	$\kappa = 0.0001$	69.08	58.74	43.79	36.77
		$\kappa = 0.0002$	86.35	79.70	62.41	51.35
		$\kappa = 0.0005$	72.27	66.38	51.30	42.76

Table 2.6: Performance (Route opt %) comparison of different kinds Gaussian additive noise.

Network size	20	30	50	80
$\beta_g = 0.1$	82.08	72.75	57.30	51.26
$\beta_g = 0.01$	88.16	83.52	64.85	59.47
$\beta_g = 0.001$	90.08	87.12	67.10	62.80
$\beta_g = 0.0001$	62.52	52.06	34.18	30.25

Table 2.7: Performance comparison of discarding 0/1 term.

		time mean	time SD	Route opt %
10-node network	Without 0/1 term	0.0033	0.057	81.5
	With 0/1 term	0.0019	0.043	23.6
20-node network	Without 0/1 term	0.054	0.427	75.89
	With 0/1 term	0.034	0.30	21.61
30-node network	Without 0/1 term	0.67	3.06	67.44
	With 0/1 term	0.28	1.73	20.80

in Table 2.7. We can see that the 0/1 term helps the chaotic neural network to converge, as simulations with the 0/1 term need less time to reach valid solutions compared with simulations without the 0/1 term. However, this 0/1 term will affect the network’s capability to reach global optima. Take the 10-node network for example, if we set the weighting coefficient of the 0/1 term, μ_e , equals to the weighting coefficient of the cost term μ_1 , the Route opt is 17.43%; and when $\mu_e = \frac{1}{2}\mu_1$, the Route opt is 23.6%.

2.5 Conclusions

We proposed a solution using *noisy Hopfield neural networks* for the shortest path routing problem (SPRP). We simplify the energy function for the problem so there are only two weighting constants to tune. We also evaluated effects of different noise distributions and decaying rates, as well as the sensitivity of the neural network model to the values of weighting constants. Adding the decaying noise enhances the route optimal rate for the SPRP, compared with other heuristics based on the HNN.

We should note that in general, routing consists of two parts of work: 1) Acquire the network state information such as group member, network topology, link

cost, delay and available bandwidth resource; 2) Construct a feasible optimal tree with those information. In our work, we focus on the second task and assume the completely true state of the network is available to the source node.

While using uniform distributed stochastic noise, we investigated the effectiveness of both exponential decaying and linear decaying in the noise amplitude. For either kind of decaying schemes, a proper decaying rate is essential for a good performance of the NHNN. From this point of view, Gaussian distribution is more practicable, as the performance of the neural network is less sensitive to the damping factor when using Gaussian distribution compared to using noise with uniform distribution. About the noise level setting for problems with different sizes, one guide is that to maintain the balance in Eqn. 2.2. From Table 2.1, we can see that while the initial noise level varying from 0.5 to 5.0, the movement of "route opt" is within 10%. Hence choosing a proper noise value is not difficult. On another hand, we have only two weighting coefficients in the energy function. One is a penalty term and another one is an optimization term. Hence the tuning of the parameters is simple and convenient.

The NHNN introduces stochastic noisy dynamics into the HNN. In the cases that we have studied, the NHNN is more capable of achieving the optimal path compared with the HNN. But on another hand, the computation time increases because of the decaying noise. With the NHNN, the network takes more iteration steps to converge if the damping factor is small. Hence a client may choose the NHNN when he does not care computation time very much, but needs a network which is more capable

to reach the optimum.

Chapter 3

Noisy Chaotic Neural Networks for Delay Constrained Multicast Routing

3.1 Introduction

Multimedia communications and data delivery [78–81] have been motivating research in multicast routing, resource reservation, and architecture development. Along with progress in audio, video, and data storage technologies, multicast routing is very 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) [6].

For most real-time applications, the routing algorithm needs to find an optimal multicast route that has sufficient resources to guarantee the required quality of service (QoS) [82]. This problem is called QoS constrained multicast routing and is also proved to be a NP-complete problem [6]. Furthermore, QoS requirements may

be inter-dependent or have effects on one another, which makes the problem more complicated.

Research in QoS constrained multicast routing includes routing with the bandwidth requirement [83], many-to-many multicast routing [84], routing with imprecise information [85], and dynamic routing (the node may join or leave the network at any instance of time) [86]. We focus on delay constrained multicast routing (DCMR), which aims to find a minimum cost 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.

Kompella et al [87] presented the first heuristic for the CST problem. It is a source-based heuristic, which assumes that through the routing protocol the source node can obtain accurate information about the connection status of the communication network. Widyono [88] proposed four CST heuristics based on the constrained Bellman-Ford (CBF) algorithm and a merge algorithm. Constrained adaptive ordering (CAO) heuristic is the best among them. Sun and Langendoerfer [89] proposed another heuristic for the CST problem. It constructs a CST by merging a minimum cost tree with a minimum delay tree, where the minimum trees are found by Dijkstra's algorithm. Parsa et al proposed the bounded shortest multicast algorithm (BSMA) [90,91]. This method finds the minimum-delay multicast tree also through Dijkstra's algorithm in the first place, then reduces the cost of so called *super-edges* to each destination monotonously through a *k*th-shortest path algorithm [92], while satisfying the delay constraint until no further optimization in the total cost is pos-

sible. Salama et al [19] evaluated all these heuristic algorithms (Kompella's, CAO, 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) presented by Manyem [93], BSMA's final costs are less than 7% more expensive than optima found by the COPT, i.e., the multicast tree obtained by the BSMA is not the global optimal one. The COPT was implemented by a branch and bound technique and therefore needs a very long execution time, as a result, it is not applicable to the large scale network.

By defining proper energy (cost) functions and deriving associated weights between neurons, one can use neural networks technique to solve routing problems. After Hopfield presented the HNN, there have been many researchers exploring and improving the performance of HNNs on versatile applications. Nozawa [37] proposed a chaotic neural network by adding negative self-feedbacks into HNNs to solve the local minimum problem. The chaotic neural network has been further developed in recent years, such as introducing some time-dependent parameters for richer dynamics [94–96]. The stochastic chaotic simulated annealing (SCSA) method, which we are going to use for DCMR problem, is proposed by an introduction of a decaying stochastic noise into the TCNN. Compared with TCNNs, NCNNs can solve the traveling salesman problem (TSP) more efficiently [45]. Wang and Shi [97] proposed a gradual noisy chaotic neural network (G-NCNN), which is a combination of the NCNN and the gradual expansion scheme [98]. They applied the G-NCNN on the broadcast scheduling problem in packet radio networks. The algorithm obtained

good performance not only in average delay time but also minimal time-division multiple-access cycle length.

Pornavalai et al [9] applied the HNN on the CST problem of an 8-node communication network. They first modified the energy function of the unconstrained unicast routing problem to be suitable for the delay constrained unicast routing problem, then changed this energy function to be able to solve the delay constrained multicast routing problem. These two steps were realized through f-type neurons and LP-type neurons, which we will deliberate in detail in Section 3.3.2.

The rest of the chapter is organized as follows. In Section 3.2, we review the formulation of the CST problem. In Section 3.3, we review the chaotic neural network model and the energy function for the CST problem, followed by the presentation of neural network dynamics. Simulation results and performance comparison are given in Section 3.4. Finally, we conclude this chapter in Section 3.5.

3.2 Problem formulation

The delay constrained multicast routing (DCMR) problem [9], 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;
2. The delay from source to each destination is not greater than a required delay constraint.

Figure 3.1: A 16-node communication network for the delay constrained multicast routing problem.

This kind of routing algorithm is desirable for applications like distance education and video-conferencing. For example, Fig. 3.1 shows a 16-node network. The source node is shown as in grey and the destination nodes are shown as dark dots with circles.

We now review the formulation proposed by Pornavalai et al [9]. An n -node communication network with D destinations is arranged on $D \times n \times n$ matrices. Matrix m is used to compute the constrained unicast route from source node s to destination $m, (m = 1, \dots, D)$. Each element in a matrix is treated as a neuron and neuron mx_i (element xi in matrix m) describes the link from node x to node i for destination m in the communication network.

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

$$V_{xi}^{(m)} = \begin{cases} 1, & \text{if the arc from node } x \text{ to node } i \text{ is on the final tree for} \\ & \text{destination } m; \\ 0, & \text{otherwise.} \end{cases} \quad (3.1)$$

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

$$P_{xi} = \begin{cases} 1, & \text{if the arc from node } x \text{ to node } i \text{ does not exist;} \\ 0, & \text{otherwise.} \end{cases} \quad (3.2)$$

The cost of a link from node x to node i is denoted by $C_{xi} \geq 0$. For non-existent arcs, $C_{xi} = 0$. L_{xi} denotes a finite real positive delay on the link from node x to node i . We assume that link cost and link delay are independent on each other. For example, cost could be a measure of the amount of buffer space or channel bandwidth used, and the link delay 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.

3.3 The NCNN for the CST problem

3.3.1 Noisy chaotic neural networks

Wang *et al.* [45,59] proposed the NCNN by adding decaying stochastic noise into the TCNN [2]. Besides the chaotic nature of the TCNN, the NCNN is also stochastic, hence alternatively known as stochastic chaotic simulated annealing (SCSA). The NCNN performs stochastic searching both before and after chaos disappears, and is more likely to find optimal or sub-optimal solutions compared to both the TCNN and simulated annealing [45].

The dynamic equations for the NCNN are described as follows:

$$\begin{aligned}
 U_{xi}^{(m)}(t+1) = & kU_{xi}^{(m)}(t) + \alpha \left(\sum_{y=1}^N \sum_{j=1}^N w_{yj,xi} V_{yj}^{(m)}(t) + I_{xi} \right) \\
 & - z_{xi}(t) [V_{xi}^{(m)}(t) - I_0] + n(t)
 \end{aligned} \tag{3.3}$$

$$V_{xi}^{(m)} = g_{xi}^{(m)}(U_{xi}^{(m)}) = \frac{1}{1 + e^{-U_{xi}^{(m)}/\epsilon_{xi}^{(m)}}} \tag{3.4}$$

where the variables are

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

Furthermore, α is the positive scaling parameter for inputs. $w_{yj,xi}$ is the connection weight from neuron yj to neuron xi , which satisfies the following conditions:

$$w_{yj,xi} = w_{xi,yj}; \quad w_{yjyj} = 0; \quad (3.5)$$

The connection weights between neurons are derived from the energy function

(Equation (3.6)) to assure that the energy function will decrease monotonically as neurons update after both the noise and chaos disappear.

$$\sum_{y=1}^N \sum_{j=1}^N w_{yj,xi} V_{yj}^{(m)}(t) + I_{xi} = -\frac{\partial E}{\partial V_{xi}^{(m)}} \quad (3.6)$$

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 problem. For example, in our work, the final neuron outputs should stand for an optimal delay constraint multicast tree. In the absence of noise, i.e., $n(t) = 0$, the NCNN reduces to the TCNN of Chen and Aihara [2].

Figure.3.2 shows the dynamics of the TCNN and the NCNN, respectively. Compared with the TCNN, the NCNN explores 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.

3.3.2 Energy function

Our energy function is based on the work of Pornavalai et al [9]. In their model, the final outputs of neurons are firstly pushed towards 0 or 1 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 0(1) 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 status, i.e., all outputs above the average value are set to 1, otherwise 0.

Figure 3.2: A comparison of dynamics of (a) the NCNN and (b) the TCNN.

The total energy function E is the sum of energy functions for the delay constrained unicast routing to each destination [9]:

$$E = \sum_{m \in D} E^{(m)} \quad (3.7)$$

$$E^{(m)} = \mu_1 E_1^{(m)} + \mu_2 E_2^{(m)} + \mu_3 E_3^{(m)} + \mu_4 E_{4,LP}^{(m)} \quad (3.8)$$

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. Selection of these weighting coefficients is discussed later in Simulation section.

E_1^m is the total cost of the unicast route:

$$E_1^{(m)} = \sum_{x=1}^n \sum_{i=1, i \neq x}^n C_{xi} f_{xi}^{(m)}(V) V_{xi}^{(m)} \quad (3.9)$$

$$f_{xi}^{(m)}(V) = \frac{1}{1 + \sum_{j=1, j \neq m, j \in D}^n V_{xi}^{(j)}} \quad (3.10)$$

The connections among the neurons in different matrices are not directly from the outputs but through some special neurons, i.e., f-type neurons [9] which have an input-output function specified in Equation (3.10). These f-type neurons help to reduce the cost term $E_1^{(m)}$ in proportion to the number of the unicast routes who 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.

$E_2^{(m)}$ 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.

$$E_2^{(m)} = (1 - V_{ms}^{(m)}) + \sum_{x=1}^n \left\{ \sum_{i=1, i \neq x}^n V_{xi}^{(m)} - \sum_{i=1, i \neq x}^n V_{ix}^{(m)} \right\}^2 \quad (3.11)$$

$E_3^{(m)}$ penalizes neurons that represent non-existing links of the network:

$$E_3^{(m)} = \sum_{x=1}^n \sum_{i=1, i \neq x}^n P_{xi} V_{xi}^{(m)} \quad (3.12)$$

$E_4^{(m)}$ is used to satisfy the delay constraint, which is an in-equality constraint:

$$\sum_{x=1}^n \sum_{i=1, i \neq x}^n L_{xi} V_{xi}^{(m)} \leq \Delta \quad V_{xi}^{(m)} \in \{0, 1\} \quad (3.13)$$

$$E_{4,LP}^{(m)} = \int h(z) dz$$

$$h(z) = \begin{cases} 0, & \text{if } z \leq 0; \\ z, & \text{otherwise.} \end{cases}$$

where

$$z = \sum_{x=1}^n \sum_{i=1, i \neq x}^n L_{xi} V_{xi}^{(m)} - \Delta \quad (3.14)$$

Here Δ is the delay bound. An LP (linear programming) type neuron [9] 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 another word, minimization of the energy function will force this term to approach 0, i.e., the delay of the path must not be greater than the delay constraint.

3.3.3 The structure of the neural network

Substituting the energy function in (3.3) with (3.8), we obtain the network dynamics, i.e., the difference equation of the neural network as follows:

$$\begin{aligned}
 U_{xi}^{(m)}(t+1) = & kU_{xi}^{(m)}(t) - \mu_1 C_{xi} f_{xi}^{(m)}(V)(1 - \delta_{xm} \delta_{is}) \\
 & - \mu_2 [-\delta_{xm} \delta_{is} + \sum_{y=1, y \neq x}^n (V_{xy}^{(m)} - V_{yx}^{(m)}) + \sum_{y=1, y \neq i}^n (V_{iy}^{(m)} - V_{yi}^{(m)})] \\
 & - \mu_3 P_{xi}(1 - \delta_{xm} \delta_{is}) - \mu_4 L_{xi}(1 - \delta_{xm} \delta_{is}) h(z) \\
 & - z_i(t)[V_{xi}^{(m)}(t) - I_0] + n(t)
 \end{aligned} \tag{3.15}$$

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 com-

parison of Equations (3.3) and (3.15):

$$w_{yj,xi} = -\mu_2\delta_{xy} + \mu_2\delta_{xj} - \mu_2\delta_{ij} + \mu_2\delta_{iy} \quad (3.16)$$

$$\begin{aligned} I_{xi} = & -\mu_1 C_{xi} f_{xi}^{(m)}(V)(1 - \delta_{xm}\delta_{is}) + \mu_2\delta_{xm}\delta_{is} \\ & - \mu_3 P_{xi}(1 - \delta_{xm}\delta_{is}) - \mu_4 L_{xi}(1 - \delta_{xm}\delta_{is})h(z) \end{aligned} \quad (3.17)$$

Equation (3.17) can also be written concisely as:

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

No terms in the connection matrix depend on the cost of links and the network topology. The costs are mapped into the biases [10], 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 neural network to adapt to the changes in the environment. Thus the hardware implementation of the neural network is a general one, independent of the topology or link costs of the communications network.

3.4 Simulation results

The communication networks used in the simulation are constructed randomly using a graph generator (based on the method proposed by Waxman [99]). A communications network with n nodes is randomly placed on a Cartesian coordinate. The

Table 3.1: 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	154	25
Case #5	100	5	252	25
Case #6	50	5	171	20
Case #7	50	15		
Case #8	50	25		
Case #9	50	35		
Case #10	50	45		

cost and delay of an arc from node x to node i , i.e., C_{xi} and L_{xi} , are randomly generated and normalized. Fig.3.3(a) presents an example of randomly generated 50-node network. Fig.3.3(b) shows the delay constrained optimal multicast tree found by the NCNN for the 50-node network. The source node is shown as a circle and the destination nodes are shown as dark circles.

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 follows [45]: $\epsilon_{xi} = \epsilon = 0.004$, 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 though the prior definition in Equation (3.1). We assume that the average value of the neuron outputs is V_{Avg} , if

(a)

(b)

Figure 3.3: (a) A 50-node network used in our simulations, with an average degree (the number of links for a node) 4; (b) The delay constrained optimal multicast tree found by the NCNN for the 50-node network with one source node (circle) and 5 destinations (black dots).

$V_{xi} \geq V_{Avg}$, then $V_{xi} = 1$, i.e., the neuron is on, which means 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 $z_{xi}(0)$ and the initial amplitude of random noise $A[n(0)]$ are set properly to keep balance of every term in Equation (3.3). β_1 determines the length of transient chaos [2] and β_2 determines the cooling schedules of noise [45]. 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 [9] as follows:

$$\mu_1 = 150; \quad \mu_2 = 5000; \quad \mu_3 = 5000; \quad \mu_4 = 200$$

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 reach a better solution, we need to slightly reduce μ_1 if the final tree cost is larger, and we need to enlarge μ_4 if the delay bound is stricter. 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 5000 later in this section. The constants μ_2 and μ_3 are instance-independent, since the μ_2 term promotes completeness of the route and the μ_3 term prevents usages of non-existing links.

For an 8-node communication network shown in Fig. 3.4, the source node s is labeled 1, and destination nodes are labeled 4, 5, 7 and 8, respectively. The algorithm

Figure 3.4: Network topology of the optimal delay constrained multicast route for an eight-node communication network when the delay constraint is 15. (a), (b), and (c), presents the solutions 1, 2, and 3 in Table 3.2, respectively; (d) the output matrix for solution 2.

finds three solutions when the delay constraint (Δ) is 15. The final optimal route is presented in Fig.3.4. The output matrix of the neural network is also showed in Fig.3.4(d). We compare these three solutions in Table 3.2 (the solutions 1, 2, and 3 corresponds to (a), (b), and (c) in Fig.3.4, respectively).

When the delay constraint (Δ) is 20, the NCNN finds the optimal tree shown in Fig. 3.5, with total cost 8. The proposed algorithm is run 1000 times using randomly generated topologies and randomly assigned link parameters for each in-

Table 3.2: Comparison of three solutions when the delay constraint $\Delta = 15$.

	solution 1		solution 2		solution 3	
Destination No.	cost	delay	cost	delay	cost	delay
4	5	5	5	11	5	5
5	1	5	1	5	1	5
7	3	13	3	13	6	11
8	4	6	4	6	4	6
total	10	13	10	13	10	11

Figure 3.5: The optimal delay constrained multicast route when $\Delta = 20$ (a) The optimal route; (b) the output matrix.

Table 3.3: Results for the HNN, TCNN, and NCNN for instances #1–#10. “sd” stands for “standard deviation”.

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

stance (#1–#10). In all these cases we have studied, the results have converged to stable states within 4000–6000 iterations. The statistical results are shown in Table 3.3. From those numbers, we can see that NCNNs and TCNNs are more capable to reach solutions with lower cost compared with HNNs. But the trade-off is computation time. NCNNs and TCNNs need longer time than HNNs to converge. In Table 3.3, you can see that the NCNN may require up to thousands of seconds for the DCMR problem on large scale network. This is because that we simulated the neural network on a sequential processor. However, the fast respondent computation time of the neural network is based on its parallel architecture. We will discuss this later in Conclusion section.

The performances of the NCNN is compared with the TCNN and HNN with results as shown in Figure 3.6. From the HNN to the TCNN, and then from the TCNN to the NCNN, the network dynamics become more and more complex, 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

Figure 3.6: 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 1000 runs.

well as the complexity of the dynamic depending on the preference of the client.

To compare the chaotic neural network with the BSMA [90, 91], we coded the BSMA algorithm and ran both of them on 10 instances in Table 3.1. Since the BSMA is a deterministic heuristic algorithm, it will reach the same solution every time. We compare this result with the optimal route cost achieved by TCNNs and NCNNs. The comparison is shown in histogram in Figure 3.7. Though the chaotic neural network can not promise to reach the global optimal in every run, it indeed can find a solution with a smaller route cost compared with the BSMA. Since we simulate the neural network on the sequential computer, the advantage of its parallel processing capability has not been performed. We have also discussed this point at the end of this Chapter. BSMA runs in $O(\kappa|V|^3 \log |V|)$ time. V stands for the

Figure 3.7: Comparisons of the optimal route costs found by the chaotic neural network (NCNNs and TCNNs) and the BSMA for ten instances. The numbers 1-10 in the horizontal axis stand for the instances #1-#10 in Table 3.1, respectively.

number of nodes, and κ is a factor which relates to network density and scale [19].

The COPT algorithm [93] which is implemented by the branch and bound technique can only be applied to small size communication networks because of its large computation time. The optimal results from the NCNN and TCNN on instances #1-#3 are the same with those obtained by COPT algorithm.

We investigate the performance of NCNNs on cases #2 and #3 with different settings of noise levels $A[n(0)]$ and the weighting constants μ_1 and μ_4 . Results are shown in the Table 3.4, 3.5, and 3.6. The *Local minima rate* shows the percentage of simulations in which the neural network find solutions within the delay constraint but are not the global optima. The *Infeasible rate* means the percentage of the simulations in which the solutions do not satisfy the delay constraint or the algorithm

Table 3.4: Performance comparison of the NCNN models with different noise levels for the networks of 16-node and 30-node. $\mu_1 = 150, \mu_4 = 200$.

16-node network					
Noise level $A[n(0)]$	0.02	0.05	0.1	0.2	0.4
Cost mean $\pm$ sd	25.6 $\pm$ 2.85	25.7 $\pm$ 2.85	25.5 $\pm$ 2.65	25.6 $\pm$ 2.65	25.7 $\pm$ 2.78
Time mean $\pm$ sd (s)	15.8 $\pm$ 2.26	15.7 $\pm$ 2.27	15.7 $\pm$ 2.26	15.7 $\pm$ 2.34	15.7 $\pm$ 2.3
Global minima rate	29.2	29.8	30.1	31.6	27.4
Local minima rate	67	66.7	66.4	63.4	69.2
Infeasible rate	3.8	3.5	3.5	5	3.4
30-node network					
Noise level $A[n(0)]$	0.02	0.05	0.1	0.2	0.4
Cost mean $\pm$ sd	21.65 $\pm$ 2.49	20.92 $\pm$ 1.76	20.29 $\pm$ 1.41	20.21 $\pm$ 1.34	20.14 $\pm$ 0.97
Time mean $\pm$ sd (s)	42.26 $\pm$ 9.29	45.19 $\pm$ 5.68	41.13 $\pm$ 4.82	48.31 $\pm$ 11.7	52.67 $\pm$ 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

does not converge. In the previous Tables and Figures in this Chapter, the weighting coefficients are properly set and the “Infeasible rates” are 0.

As the noise level goes high, the computation time increases, the chance of reaching the global optima is also enhanced, and there are more infeasible solutions if the 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 neural networks. The neural network is more capable to reach the global optima when the weighting coefficient for the cost term, i.e., μ_1 , is larger. But the increase of the cost term will have impact on the other terms, e.g. if μ_1 is too large, the solution may not satisfy the delay constraint, or the neural network does not even converge.

The effect of the damping factor β_1 and β_2 is analyzed and results are shown

Table 3.5: Performance comparison of the NCNN models with different weighting constants μ_1 the networks of 16-node ($\mu_4 = 200, A[n(0)] = 0.1$) and 30-node ($\mu_4 = 200, A[n(0)] = 0.02$).

16-node network				
μ_1	100	150	200	250
Cost mean $\pm$ sd	28.0 $\pm$ 3.82	25.5 $\pm$ 2.65	23.9 $\pm$ 2.15	23.3 $\pm$ 1.53
Time mean $\pm$ sd (s)	16.5 $\pm$ 2.43	15.7 $\pm$ 2.26	16.1 $\pm$ 2.09	16.6 $\pm$ 1.89
Global minima rate	8.5	30.1	56.4	75.8
Local minima rate	88.5	66.4	33.4	10
Infeasible rate	3	3.5	10.2	14.2
30-node network				
μ_1	125	150	175	200
Cost mean $\pm$ sd	23.79 $\pm$ 3.41	21.65 $\pm$ 2.49	20.58 $\pm$ 1.60	24.59 $\pm$ 1.39
Time mean $\pm$ sd (s)	43.94 $\pm$ 4.52	42.26 $\pm$ 9.29	44.01 $\pm$ 18.4	44.30 $\pm$ 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

Table 3.6: Performance comparison of the NCNN models with different weighting constants μ_4 for the networks of 16-node ($\mu_1 = 250, A[n(0)] = 0.1$) and 30-node ($\mu_1 = 150, A[n(0)] = 0.02$).

16-node network						
μ_4	100		200		300	
Cost mean $\pm$ sd	24.8 $\pm$ 1.9		23.3 $\pm$ 1.53		23.1 $\pm$ 1.33	
Time mean $\pm$ sd (s)	18.6 $\pm$ 2.6		16.6 $\pm$ 1.89		15.9 $\pm$ 1.23	
Global minima rate	39.6		75.8		82.8	
Local minima rate	7.2		10		14	
Infeasible rate	53.2		14.2		3.2	
30-node network						
μ_4	150	200	250	300	350	400
Cost mean $\pm$ sd	21.70 $\pm$ 2.59	21.65 $\pm$ 2.49	21.61 $\pm$ 2.54	21.56 $\pm$ 2.49	21.69 $\pm$ 2.34	21.73 $\pm$ 2.28
Time mean $\pm$ sd (s)	41.41 $\pm$ 8.74	42.26 $\pm$ 9.29	42.29 $\pm$ 4.36	42.45 $\pm$ 10.1	42.96 $\pm$ 10.9	42.08 $\pm$ 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

Table 3.7: Performance comparison of the NCNN models with different β_1 on the 8-node network with $\Delta = 20$, $\beta_2 = 0.0001$ and the 16-node network with $\Delta = 20$, $\beta_2 = 0.0001$.

8-node network					
β_1		0.01	0.005	0.001	0.0005
Time (s)	mean	1.10	1.27	1.55	3.35
	SD	0.30	0.37	0.50	0.48
Route opt %		93.02	95.30	96.27	96.85
16-node network					
β_1		0.01	0.005	0.001	0.0005
Time(s)	mean	2.04	2.83	10.4	18.4
	SD	0.81	0.81	2.0	3.28
Route opt %		50	82.4	79.4	76

in Table 3.7 and 3.8. In applications, we can balance the “route opt” ratio and the execution time through parameters β_1 and β_2 to control the decay of chaotic dynamics and noise. The larger β_1 and β_2 , the faster the NCNN converges, while the smaller the parameters, the more probable the NCNN can reach the global optimal.

3.5 Conclusions

By the NCNN and the TCNN, we solve the delay constrained optimal multicast routing problem in communications which desire: i) delay constraint on the paths from the source to each destination; and ii) minimum cost of the multicast tree. Simulation results showed that the NCNN has higher probability in finding global optima compared to the TCNN and the original HNN model. The cost of the multicast tree found by the chaotic neural network is equal to if not less than the one obtained by the BSMA. Results from the COPT algorithm showed that chaotic

Table 3.8: Performance comparison of the NCNN models with different β_2 on the 8-node network with $\Delta = 20$, $\beta_1 = 0.001$ and the 16-node network with $\Delta = 20$, $\beta_1 = 0.001$

8-node network					
β_2		0.01	0.005	0.001	0.0005
Time (s)	mean	0.89	1.01	1.55	1.85
	SD	0.45	0.45	0.50	0.47
Route opt %		94.04	94.70	96.27	97.23
16-node network					
β_2		0.01	0.005	0.001	0.0005
Time(s)	mean	5.2	9.9	10.1	10.3
	SD	1.12	2.13	1.87	1.98
Route opt %		78.5	80.2	82	82

neural networks, i.e., TCNNs and NCNNs, are able to find global optima. The chaotic neural network models are also applicable to the communication network with asymmetric link characteristics or with variable delay constraints for different destinations.

The BSMA algorithm is a deterministic searching algorithm. In the cases that we have investigated, the results from chaotic neural networks are comparable to if not better than the BSMA. There are also some evolutionary computation methods on the multicast routing problem [100,101]. A. Roy and S. K. Das [68] proposed an algorithm based on the multi-objective genetic algorithm. The new model took care of not only the delay on the path and the cost of the tree, but also the bandwidth utilization. Number of iterations increases linearly along with the network size. Chu et al [102] present an ant colony based heuristic to solve the QoS constrained multi-cast routing problems in an 8 node network. The transmission quality of the data in

the network is effectively improved. A full scale exploration and analysis on various methods for the CST problem would be significant to research on multicast routing. For example, evolutionary computation, neural networks, simulated annealing algorithm and other searching heuristics.

In this chapter, the equation governing dynamics of neural networks and the formulation of the delay constrained multicast routing problem have been reviewed. The neural techniques, which operate in continuous solution space, will consume considerably more computational resources when simulated on a sequential digital computer than the discrete heuristics. If using parallel processing system, neural networks will perform much better for sure in computation-time wise.

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. Military communications also pay a lot of attentions on the research of multicast routing. Except a requirement similar to commercial systems, there is a need of the potential to route high-priority packets under radically changing environmental conditions. In large-size military communication systems, the amount of traffic and the number of nodes and links may be so large that the implementation of routing algorithms in a sequential general-purpose computer may not be effective. On the other hand, implementation in a parallel neural network computer has the potential for much greater speed [32].

To utilize the parallel processing ability of neural networks, we need to explore the hardware implementation of the NCNN. There are mainly two parts: i) the

damping of the self-feedback; $z_{xi}(t + 1) = (1 - \beta_1)z_{xi}(t)$; ii) the addition of the stochastic noise $n(t)$. For the HNN, synaptic weights are implemented by resistors of resistances in the analog circuits design. As one of the synaptic weights, the damping of the self-feedback may be achieved by adjusting the parameters of the resistors. At the same time, the random noise source can be used to add noise into the neural network system. With parallel computing networks, we expect encouraging results in significant runtime savings.

Chapter 4

Noisy Chaotic Neural Networks with Variable Thresholds for the Frequency Assignment Problem in Satellite Communications

4.1 Introduction

Wireless communication has received a lot of attention these years due to its various applications. One important research direction in wireless communication is interference minimization, so as to guarantee a desired level of quality of service. The frequency rearrangement is an effective complement alongside with the technique to reduce the interference itself. Among diverse formulations and objectives of frequency assignment problems (FAP) [22, 23], we focus on frequency assignments in satellite communications in our work.

Satellites communications are a billion-dollar technology, with applications rang-

ing from weather forecasting to mobile telecommunications [103]. Nowadays, there is an increasing number of satellites in geostationary orbits. In order to accommodate crowded satellites in the same orbit, optimal frequency assignments are necessary to provide high quality transmissions. In satellite communication systems, cochannel interference is the greatest problem which seriously affects the design and operation of the system [26]. Minimization of cochannel interference has arisen as a major issue to deal with in satellite communications.

The calculation of interference involves nonlinear terms. To avoid the treatment of nonlinearity, Mizuike and Ito [26] proposed a segmentation method which divides the commonly shared frequency band into a number of segments. Through segmentation, nonlinear terms can be evaluated in a linear manner. The carrier is uniformly divided to a collection of consecutive unit segments whose width can be set arbitrarily. The FAP is then reduced to placing carriers into an integral multiple of unit segments. They then presented a method based on the branch-and-bound approach for the FAP in satellite communications. The application of their method on both the intersystem problem and intrasystem problem showed that the method was effective in the interference minimization. However, the branch-and-bound algorithm may fail when applied to large instances [104].

For both intersystem interference and intrasystem interference [26], frequency rearrangements take advantage of carrier interleaving and are effective in practical situations. Early efforts have focused on various analytical methods for evaluations of cochannel interference [24, 25], rather than systematic methods to optimize fre-

quency assignments and to reduce cochannel interference. The later work of Mizuike and Ito [26] revealed the importance of a mathematical model for reduction of interference. They formulated the cochannel interference reduction problem as a FAP, which minimizes the largest and the total interference among carriers.

The FAP, where frequencies of one set of carriers are to be rearranged while keeping the other set fixed, exists in many areas, such as mobile communications, broadcast, and satellite communications [105, 106], which require optimal assignments of limited frequency resources to a number of users. Cochannel interference can be evaluated by considering each pair of carriers which use the same frequency.

The FAP was proven to be an NP-complete COP [26]. Due to the NP-complete nature of this assignment problem, heuristic methods, especially neural networks, are commonly adopted. The objectives of the FAP [104] in satellite communications are:

- 1) minimize the largest interference of elements selected for the assignment;
- 2) minimize the sum of interferences of all the selected elements.

Funabiki and Nishikawa [104] presented a gradual neural network (GNN) which consists of $N \times M$ binary neurons for an N -carrier- M -segment system with a gradual expansion scheme of activated neurons. They sorted the $N \times M$ neurons in an ascending order of the cost which firing of the neuron will cause and divided the neurons into P groups $g_1, g_2, \dots, g_P$, where g_1 is the group of neurons with the smallest interference and g_P is the group of neurons with the largest interference. The GNN starts with group g_1 and searches for a solution in several phases, with each

phase adding one next group of neurons. The algorithm stops when a feasible solution is found or all neurons have been added. The objective of the FAP is achieved by searching from the neurons starting with the smallest interference. Hence, optimization is obtained by gradually increasing the number of activated neurons in the order of cost. Here, the cost optimization is achieved by a gradual expansion scheme and a binary neural network is in charge of the satisfaction of constraints. However, the value of parameter P is difficult to determine when the problem size is large and different values for P needs to be tried out for different cases. Furthermore, the multi-phase searching inevitably leads to heavy computation especially for large problems.

Salcedo-Sanz *et al.* combined the Hopfield network with simulated annealing [107] and the genetic algorithm [105] to solve the FAP in satellite communications. The HNN manages constraints and simulated annealing (genetic algorithm) improves the solution quality. The Hopfield network with simulated annealing (HopSA) [107] and the neural-genetic algorithm (NG) [105] are claimed to be more scalable than previous algorithms. However, as kinds of hybrid algorithms, there is an increase in the computational cost of the HopSA and the NG compared with the GNN to find optimal or near-optimal solutions [104, 107].

In this Chapter, we continue using the FAP formulation given by [26] and used in [104, 105, 107]. As the simulation on several COPs showed that chaotic search is efficient in approaching the global optimum or sub-optima [37, 38], we solve the FAP through chaotic neural networks.

4.2 Satellite frequency assignments using the TCNN-VT

The objective of the satellite FAP is to minimize the cochannel interference between two satellite systems by rearranging the frequency assignments. This NP-complete problem is difficult to solve, especially for large-size problems, and is yet growing in importance, since we increasingly depend on satellites to fulfill our communications needs.

In this part, we further develop the TCNN by proposing a transiently chaotic neural network with variable thresholds (TCNN-VT) by letting the thresholds of neurons vary with the interference of the assignment which the neuron represents. The thresholds are designed to minimize the largest interference after frequency rearrangements. We apply the TCNN-VT on the FAP in satellite communications and the simulation results show that the TCNN-VT is efficient.

4.2.1 Transiently chaotic neural networks with variable thresholds

The TCNN [2] model is described as follows:

$$x_{ij}(t) = \frac{1}{1 + e^{-y_{ij}(t)/\varepsilon}} \quad (4.1)$$

$$y_{ij}(t+1) = ky_{ij}(t) + \alpha \left[\sum_{p=1, p \neq i}^N \sum_{q=1, q \neq j}^M w_{ijpq} x_{pq}(t) + I_{ij} \right] - z(t) [x_{ij}(t) - I_0] \quad (4.2)$$

$$z(t+1) = (1 - \beta)z(t) \quad (4.3)$$

where x_{ij} and y_{ij} are output and internal state of neuron ij , respectively. w_{ijpq} is the connection weight from neuron ij to neuron pq , which is confined to the following conditions [108]:

$$\sum_{p=1, p \neq i}^N \sum_{q=1, q \neq j}^M w_{ijpq} x_{pq}(t) + I_{ij} = -\partial E / \partial x_{ij} \quad (4.4)$$

Furthermore, ε denotes the steepness parameter of the neuron activity function ($\varepsilon > 0$). k is the damping factor of the nerve membrane ($0 \leq k \leq 1$). α is the positive scaling parameter for inputs. I_{ij} is an input bias of neuron ij . $z(t)$ is the self-feedback neuronal connection weight ($z(t) \geq 0$). β is the damping factor for the time-dependent neuronal self coupling ($0 \leq \beta \leq 1$). E denotes the energy function, whose minimization corresponds to optimal solutions of the combinatorial optimization problem. The connection weights between neurons are derived from

the energy function by Equation (4.4) so that the energy function will decrease monotonically as neurons update after the self-feedback interaction vanishes ($z = 0$).

Wang and Ross [109] studied the influence of a variable neuronal threshold on fixed points and convergence rates of an associative neural network in the presence of noise. In the original TCNN [2], I_0 is a constant positive bias in the self-feedback term. We propose the TCNN-VT by varying the threshold with the interference of the assignment which firing of the neuron represents and denote it as $I_{ij}^{(0)}$ ($0 \leq I_{ij}^{(0)} < 1$).

$$I_{ij}^{(0)} = 1 - \frac{d_{ij}}{d_{i,max}} \quad (4.5)$$

where d_{ij} is the element on row i column j of the cost matrix D , and $d_{i,max}$ is the maximum value in row i of the matrix D . Cost matrix $D = (d_{ij}, i = 1, \dots, N; j = 1, \dots, M)$ is obtained from the interference matrix $E^{(I)} = (e_{ij}, i = 1, \dots, M; j = 1, \dots, M)$ [104]. Note that the maximum value of the cost matrix does not include infinity.

Hence, the new TCNN-VT model is described as:

$$\begin{aligned} y_{ij}(t+1) = & ky_{ij}(t) + \alpha \left[\sum_{p=1, p \neq i}^N \sum_{q=1, q \neq j}^M w_{ijpq} x_{pq}(t) + I_{ij} \right] \\ & - z(t) [x_{ij}(t) - I_{ij}^{(0)}] \end{aligned} \quad (4.6)$$

4.2.2 Problem formulation

Detail descriptions and several illustrations on cochannel interference in satellite communications can be found in [26, 104, 107]. The objective of the FAP includes

two parts, i.e., minimization of the largest interference after reassignment and minimization of the total accumulated interference between two systems.

We continue using the neural network formulation given by Funabiki and Nishikawa [104] as their model needs less computation resource compared with the one proposed by Kurokawa and Kozuka [110]. An N -carrier- M -segment FAP between two systems is formulated on an $N \times M$ neural network. If the neuron output $x_{ij} = 1$ at the end of the neuron update, then carrier i is assigned to segment j , and no assignments are made if $x_{ij} = 0$. The novel aspect of the TCNN-VT is that the threshold in the self-feedback term of every neuron is dependent on the interference of the frequency assignment which the neuron represents.

There are two constraints to be formulated into the energy function for the FAP:

- 1) Every segment in system 2 must be assigned to one and at most one segment in system 1.
- 2) All segments of one carrier in system 2 should be assigned to consecutive segments in system 1 in the same order.

According to [26, 32, 104], the energy function for the TCNN-VT of the FAP is defined as:

$$E_1 = \sum_{i=1}^N \left(\sum_{j=1}^M x_{ij} - 1 \right)^2 \quad (4.7)$$

$$E_2 = \sum_{i=1}^N \sum_{j=1}^M \sum_{\substack{p=1 \\ p \neq i}}^N \sum_{q=j-c_p+1}^{j+c_i-1} x_{ij} x_{pq} . \quad (4.8)$$

If carrier i is assigned to segment j , any other carrier must not be assigned to segments from j to $(j + c_i - 1)$. The first segment of carrier p ($p \neq i$) should be assigned to the segment before $(j - c_p + 1)$ or after $(j + c_i - 1)$. As $(j - c_p + 1)$ may be negative and $(j + c_i - 1)$ may exceed the total number of the segments, i.e., M , the formulation in (4.8) has errors and produces program bugs during simulations. We revised the second term of the energy function as follows:

$$E'_2 = \sum_{i=1}^N \sum_{j=1}^M \sum_{\substack{p=1 \\ p \neq i}}^N \sum_{q=\max(j-c_p+1,1)}^{\min(j+c_i-1,M)} x_{ij} x_{pq} . \quad (4.9)$$

where $\max(x, y)$ is the larger value between (x, y) and $\min(x, y)$ is the smaller value between (x, y) .

We add the following convergence term into the energy function to force the neuron outputs approach to 0 or 1 [32]:

$$E_3 = \sum_{i=1}^N \sum_{j=1}^M x_{ij} (1 - x_{ij}) . \quad (4.10)$$

We have discarded this term when applied chaotic neural networks in Chapter 2 and 3. This is because for routing problems, other terms in the energy function can also perform this duty, i.e., to force the output to be 0 or 1. However for the FAP, the energy function has different formulation. For the FAP in satellite communications, an extra convergence term will not only save computation time, but also improve the capability of the neural network in searching for optima. We will discuss this later in the simulation analysis section.

The W_4 term is needed to fulfill the optimization of the total interference after the frequency rearrangement.

$$E_4 = \frac{W_4}{2} \sum_{i=1}^N \sum_{j=1}^M d_{ij} x_{ij} \quad (4.11)$$

The total energy function is given by the summation of four parts E_1 , E'_2 , E_3 , and E_4 :

$$\begin{aligned} E = & \frac{W_1}{2} \sum_{i=1}^N \left(\sum_{j=1}^M x_{ij} - 1 \right)^2 + \frac{W_2}{2} \sum_{i=1}^N \sum_{j=1}^M \sum_{\substack{p=1 \\ p \neq i}}^N \sum_{q=\max(j-c_p+1, 1)}^{\min(j+c_i-1, M)} x_{ij} x_{pq} \\ & + \frac{W_3}{2} \sum_{i=1}^N \sum_{j=1}^M x_{ij} (1 - x_{ij}) + \frac{W_4}{2} \sum_{i=1}^N \sum_{j=1}^M d_{ij} x_{ij} . \end{aligned} \quad (4.12)$$

where W_1 , W_2 , W_3 and W_4 are weighting coefficients. The choices of W_1 , W_2 , W_3 and W_4 are based on the rule that all terms in the energy function should be comparable in magnitude, so that none of them dominates [45]. The balance of each term in the energy function is crucial to parameter selection.

From equations (4.4), (4.17), and (4.19), the dynamic equation for the TCNN-VT can be obtained:

$$\begin{aligned} y_{ij}(t+1) = & ky_{ij}(t) + \alpha \left[-W_1 \left(\sum_{j=1}^M x_{ij} - 1 \right) - W_2 \left(\sum_{\substack{p=1 \\ p \neq i}}^N \sum_{q=\max(j-c_p+1, 1)}^{\min(j+c_i-1, M)} x_{pq} \right) \right. \\ & \left. - \frac{W_3}{2} (1 - 2x_{ij}) - \frac{W_4}{2} d_{ij} \right] - z(t) \left[x_{ij}(t) - I_{ij}^{(0)} \right] \end{aligned} \quad (4.13)$$

Different with Salcedo-Sanz *et al.* [105, 107], who use binary Hopfield networks

with neurons outputting only 0 or 1, we use a continuous neural network, i.e., the neuron output is continuous between 0 and 1. We convert the continuous output x_{ij} of neuron (i, j) to discrete neuron output x_{ij}^d as follows [111]:

$$x_{ij}^d = \begin{cases} 1, & \text{if } x_{ij} > \frac{1}{NM} \sum_{p=1}^N \sum_{q=1}^M x_{pq}(t); \\ 0, & \text{otherwise.} \end{cases} \quad (4.14)$$

4.2.3 Simulation results and discussions

Table 4.1 shows specifications of the 8 instances for the TCNN-VT. Instances 1 to 5 are from [104], where these are called instances 1-5, respectively. Using the instance generation algorithm in Appendix I, we generated instances 6 to 8 to show the performance of the TCNN-VT on large-size problems. The interference matrices $E^{(I)}$ for instances 6 to 8 are randomly generated from $[1, 100]$. An iteration is terminated once a feasible assignment is obtained or the number of iteration steps exceeds a predefined maximum number (15000 in our simulation).

Parameters for the neural network are chosen as follows [2]: $\varepsilon = 0.004$, $k = 0.99$, $\alpha = 0.015$, $\beta = 0.001$, and $z(0) = 0.1$. Initial inputs of neural networks $y_{ij}(0)$, $(i = 1, \dots, N, j = 1, \dots, M)$ are randomly generated from $[-1, 1]$.

The weight coefficients of the energy function W_i , $i = 1, \dots, 4$, for all the 8 instances are listed in Table 4.2. The tuning of these weight coefficients is necessary to obtain better performance. From our experience, along with the growing problem size, the total aggregate interference, i.e., the W_4 term increases, so does the

Table 4.1: Specifications of the FAP instances used in the simulation.

Instance	Number of carriers N	Number of segments M	Range of carrier length	Range of interference
1	4	6	1 - 2	5 - 55
2	4	6	1 - 2	1 - 9
3	10	32	1 - 8	1 - 10
4	10	32	1 - 8	1 - 100
5	10	32	1 - 8	1 - 1000
6	30	100	1 - 10	1 - 100
7	50	200	1 - 10	1 - 100
8	80	200	1 - 5	1 - 100

differences between numerical values of the W_1 term and W_2, W_3 terms. Hence, we slightly decrease W_2 and W_3 , but increase W_4 , as the problem size grows.

We run the TCNN-VT on each instance 1000 times with different randomly generated initial neuron states. Table 4.3 shows results for every instance, including the best largest interference I_L , the rate to reach the optimum (Opt rate), the average error from the optimal result, the convergence rate η (the ratio at which the neural network finds a feasible solution in 1000 runs), and the total interference I_T when the optimum of the largest interference is found. The average iteration steps T and standard deviations are also shown in this Table. The convergence rate denotes the ratio that the neural network finds a feasible solution at the end of iterations. The results show that the TCNN-VT is effective in reducing the largest interference and total interference by rearranging the frequency assignment.

Table 4.4 shows results obtained by the TCNN-VT and the comparison with the GNN [104] and the HopSA [107]. Results of the GNN are from references [104,107],

Table 4.2: Weight coefficients $W_i, i = 1, \dots, 4$, of the energy function for all the 8 instances. W_1 is fixed at 1.0.

#	W_2	W_3	W_4
1	1.0	0.7	0.00015
2	1.0	0.7	0.00015
3	0.4	0.7	0.00015
4	0.4	0.7	0.0002
5	0.35	0.7	0.0002
6	0.2	0.6	0.0002
7	0.2	0.6	0.0002
8	0.2	0.6	0.0002

Table 4.3: The performance of the TCNN-VT on 8 instances. # denotes the instance number. I_L is the largest interference and I_T is the total interference. The interference is shown as the best and average values (Best/ Ave). “Opt rate” stands for the rate that the TCNN-VT reached the optimum in the 1000 runs. T is the average number of iteration steps. η is the convergence rate. “SD” stands for “standard deviation”.

#	I_L			I_T (Best/ Ave)	T mean±SD	η (%)
	(Best/ Ave)	Opt rate %	Average error			
1	30/ 31.5	64.0	1.5	100/ 105	491 ± 84	100
2	4/ 4.8	59.4	0.81	13/ 15.4	799 ± 163	100
3	7/ 8.4	29.3	1.46	96/ 130.6	2351 ± 96.5	94.8
4	70/ 94.1	15.8	24.7	828/ 1145	2177 ± 252	89.3
5	661/ 849	23.7	188	6910/ 9527	3075 ± 268	86.6
6	78/ 83.9	23.1	5.9	3296/ 3826	2528 ± 329	77.1
7	97/ 98.1	28.1	1.1	5839/ 5722	3834 ± 397	78.4
8	93/ 96.9	21.5	3.9	6008/ 6340	4319 ± 430	73.3

hence we only have the GNN results on instances 1-5. Results of the HopSA on instances 1-5 are also from the reference [107]. As authors in [107] did not publish the average value of the largest and total interference which they found, only the best value is included in Table 4.4. For instances 6-8, we rerun the HopSA algorithm from the authors of [107].

We show the results from the GNN, the HopSA and the TCNN-VT in Table 4.4. Compared with the GNN, although the best largest interference found by the TCNN-VT is larger, the best total interference found by TCNN-VT is better. Compared with the HopSA, the TCNN-VT is more efficient, especially on large-size problems. For instances 4-7, the TCNN-VT is able to achieve solutions with smaller largest interference. For instance 7 which is a 50-carrier-200-segment FAP, the best largest interference and total interference obtained by the TCNN-VT are 117 and 5839, respectively, whereas for the HopSA, they are 124 and 6533, respectively. The HopSA fails to obtain a solution for instance 8 due to the excessive computation time. "N/A" represents the situation that the algorithm cannot find solutions in one month. In comparison, our proposed TCNN-VT finds a feasible solution in less than one minute of CPU time on all the instances.

We listed the computation times of the TCNN-VT and the HopSA [107] on all 8 instances in Table 4.5. The computation time is measured in seconds (Sec.). Also the computation time is calculated only based on convergence runs. As shown in the Table, the TCNN-VT is much more computational efficient compared with the HopSA. Also, the advantage of the TCNN-VT is more and more distinctive as the

Table 4.4: Comparison of simulation results (largest interference and total interference) obtained by the TCNN-VT, GNN and HopSA for instances 1 to 8.

#	GNN [104]		HopSA [107]		TCNN-VT	
	Largest (Best/Ave)	Total (Best/Ave)	Largest	Total	Largest (Best/Ave)	Total (Best/Ave)
1	30/ 31.5	100/ 100.8	30	100	30 / 31.5	100 / 105
2	4/ 4.9	13/ 15.4	4	13	4 / 4.8	13 / 15.4
3	7/ 8.1	85/ 99.4	7	85	7 / 8.4	96 / 130.6
4	64/ 77.1	880/ 982.0	84	888	70 / 94.1	828 / 1145
5	640/ 766.8	8693/ 9413.9	817	6910	661 / 849	6910 / 9527
6	N/A	N/A	98	3396	78 / 83.9	3296 / 3826
7	N/A	N/A	100	6533	97 / 98.1	5839 / 5722
8	N/A	N/A	N/A	N/A	93 / 96.9	6008 / 6340

problem size grows.

4.3 Noisy chaotic neural networks with variable thresholds

4.3.1 Objective mapping scheme

We have reviewed the NCNN model in Section 3.3.1. The dynamics of a single-neuron NCNN model (with only one neuron described in equations (3.3) and (3.4) for the first 3000 iterations with $I_0 = 0.1, 0.3, 0.6,$ and 0.9 are shown in Figure 4.1. Other parameters are set as the same as in [45], i.e., $\epsilon = 0.004, k = 0.9, \alpha = 0.015, z(1) = 0.1, A[n(1)] = 0.02,$ and $\beta_1 = \beta_2 = 0.001$. When $I_0 = 0.3$, the output of the neuron shows reversed period-doubling bifurcations to a fixed point. The bifurcation

Table 4.5: Comparison between the HopSA and the TCNN-VT on computation time (Sec.). Results are displayed as mean±SD (standard deviation).

#	HopSA	TCNN-VT
1	1.0±0.0	0.026±0.1
2	1.0±0.0	0.038±0.1
3	33.0±0.0	0.53±0.49
4	33.8±0.8	0.46±0.45
5	33.0±0.5	0.47±0.42
6	2845±528	16.2±2.6
7	8916±1529	49.0±3.4
8	N/A	65.9±4.1

point from period-2 oscillations to fixed points is near $x(t) = 0.3$. When I_0 is set to 0.1, 0.6, and 0.9, this bifurcation point is around 0.1, 0.6, and 0.9, respectively.

All previous methods, including the HNN [108], the TCNN [111] and the NCNN [45], combine the constraint satisfaction and the objective optimization in one energy function. A noisy chaotic neural networks with variable thresholds (NCNN-VT) is proposed [112,113], which handles only the constraint with the energy function and maps the objective to variable thresholds (biases) of the neurons.

Equation (3.3) shows that when $t = +\infty$, $z = 0$ and the NCNN output is independent of I_0 . But the output at the bifurcation point depends on I_0 , and as shown in Figure 4.1, the network dynamics has not yet fully settled at $t = 3000$. As shown in our subsequent discussions, when we use the NCNN-VT to solve the FAP, we do not wait for the network to fully converge. Rather, we continuously check the

Figure 4.1: The dynamics of the single-neuron NCNN with different values of thresholds I_0 . The X-axis is the time t and the Y-axis is the output of neuron $x(t)$.

validity of the solution and stop iterations as soon as a valid solution is found. As discussed later in the Chapter in Section 4.3.2, neurons with output greater than the average output of the entire neural network are considered firing and selected for frequency assignments. Hence the larger the value of I_0 , the more likely the neuron will be considered firing and selected for a frequency assignment.

In the NCNN-VT, the bias I_0 is designed to vary for different neurons according to the interference of the frequency assignment that the neuron represents. I_0 is then denoted as $I_{ij}^{(0)}$. The threshold affects the output of the neuron, and therefore the likelihood that the neuron's output is above the average output of all neurons and the neuron is thus selected in frequency assignment. The value of $I_{ij}^{(0)}$ is calculated according to the objective of the optimization problem using a mapping function:

$$I_{ij}^{(0)} = f(d_{ij}), \quad i = 1, 2, \dots, N; j = 1, 2, \dots, M \quad (4.15)$$

where d_{ij} is element ij in cost matrix D and represents the interference resulted by assigning carrier i to segment j .

The cost matrix $D = (d_{ij})$ is computed from the interference matrix E_I . The interference between the two systems (each with M segments) is described by an $M \times M$ interference matrix $E_I = (e_{ij})$. The ij -th element e_{ij} represents the cochannel interference when segment i in system 2 uses a common frequency with segment j in system 1. Cost d_{ij} for neuron ij is given by the largest element in interference values $e_{lj}, e_{l+1,j+1}, \dots, e_{l+c_i-1,j+c_i-1}$, where l is the first segment number of carrier i in the interference matrix and c_i is the length of carrier i [104]. Figure 4.2 shows the

(a)

(b)

Figure 4.2: Computation of cost matrix D from interference matrix E_I , extended from [104], which only showed figure (b). * denotes infinity. (a) The interference matrix and computation method. For carriers with carrier length larger than 1, the maximum value of the crossed elements on the red line represents the corresponding value in the cost matrix. (b) The cost matrix obtained from (a).

way to compute the cost matrix from interference matrix E_I . If the carrier length for carrier i is 1, i.e., $c_i = 1$, then line i for carrier i in the cost matrix is the same as in the interference matrix for carrier i . If $c_i > 1$, then we choose the largest value in the diagonal line for each j , as shown in Figure 4.2 [104]. C_{ij} , ($i = 1, 2; j = 1, \dots, 4$) denotes the j th carrier in system i .

Objectives of the FAP are to minimize the largest element in the interference matrix selected for a frequency assignment and at the same time minimize the sum

of all selected elements of the interference matrix. Since neurons with smaller firing thresholds or greater biases are more likely to be selected in the final solution, we choose the mapping function $I_{ij}^{(0)}$ for the FAP as follows:

$$I_{ij}^{(0)} = 1 - \frac{d_{ij} - d_{i,min}}{d_{i,max} - d_{i,min}} = \frac{d_{i,max} - d_{ij}}{d_{i,max} - d_{i,min}} \quad (4.16)$$

where $d_{i,max}$ and $d_{i,min}$ are the maximum and minimum values in row i of matrix D , respectively. Note that the maximum value of the cost matrix does not include infinity. Actually the neuron corresponding to assignment with infinite interference will never fire due to its prohibitive cost.

Through the objective mapping scheme as described in (4.16) which maps the cost matrix to the thresholds in the neural network model, the NCNN-VT achieves optimization objectives of the FAP. Hence the energy function needs to be concerned with only constraint terms. The separation of the objective term from the energy function will make the tuning of weighting coefficients in the energy function easier, i.e., there is no need to balance the optimization term with constraint terms. Moreover, it will improve the convergence speed compared with the NCNN as shown in the *Simulation results and discussion* part in section 4.3.3.

Thus the dynamic equation of the NCNN-VT is:

$$\begin{aligned} y_{ij}(t+1) = & ky_{ij}(t) + \alpha \left[\sum_{p=1, p \neq i}^N \sum_{q=1, q \neq j}^M w_{ijpq} x_{pq}(t) + I_{ij} \right] \\ & - z(t) [x_{ij}(t) - I_{ij}^{(0)}] + n(t) \end{aligned} \quad (4.17)$$

4.3.2 Application of the NCNN-VT to the FAP

As indicated in [26], each carrier in a satellite communications system can be divided into one or more consecutive unit segments. In order to reduce the cochannel interference between the two systems, the frequency assignments in system 2 are rearranged while the frequencies used in system 1 are fixed. In this way, the FAP is equivalent to assigning the carriers in system 2 to the segments in system 1.

In our work, we use the same two-dimensional neural network formulation as in [104], which consists of $N \times M$ neurons for the FAP with N carriers and M segments. Figure 4.3, which is extended from Figure 4 in [104], shows an example of a 4-carrier-6-segment problem. The output of each neuron x_{ij} will be converted into binary values x_{ij}^d which means the following:

$$x_{ij}^d = \begin{cases} 1, & \text{Carrier } i \text{ is assigned to segments } j \text{ to } (j + c_i - 1); \\ 0, & \text{otherwise.} \end{cases} \quad (4.18)$$

Fig. 4.3(b) shows the convergence state, with the black squares standing for the neurons with output $x_{ij}^d = 1$. We use the first segment in the carrier to denote the assignment. Carrier length $c_1 = c_3 = 1$, $c_2 = c_4 = 2$. The subsequent segments in this carrier are assigned to the consecutive segments. Fig. 4.3(c) shows the full assignment for each carrier by adding the consecutive assignments of subsequent segments in the same carrier. And Figure 4.3(d) shows the assignment of segments. Take carrier 4 in system 2 for example: it is assigned to segment 2 in system 1, as shown in Fig. 4.3(b). Since $c_4 = 2$, carrier 4 actually occupies two segments, as

Figure 4.3: The neural network formulation for the FAP (extended from Figure. 4 of [104], which only consists of (a) and (b)). (a) The 24 neurons for the 4-carrier-6-segment FAP. (b) The convergence state of the neural network. (c) The full assignment of the carriers. (Carrier length $c_2 = c_4 = 2$) (d) The assignment of segments.

shown in Fig. 4.3(c). Carrier 4 is noted as segments 5 and 6 in system 2. Hence segments 5 and 6 are assigned to segments 2 and 3 in system 1, respectively, as shown in Fig. 4.3(d). We can similarly expand from the convergence state Fig. 4.3(b) to Fig. 4.3(c) and Fig. 4.3(d) as the carrier length for each carrier is given. We only provide the solution format in Fig. 4.3(b).

As discussed in section 4.2.2, the energy function corresponding to above constraints is:

$$\begin{aligned}
 E = & \frac{W_1}{2} \sum_{i=1}^N \left(\sum_{j=1}^M x_{ij} - 1 \right)^2 + \frac{W_2}{2} \sum_{i=1}^N \sum_{j=1}^M \sum_{\substack{p=1 \\ p \neq i}}^N \sum_{q=\max(j-c_p+1, 1)}^{\min(j+c_i-1, M)} x_{ij} x_{pq} \\
 & + \frac{W_3}{2} \sum_{i=1}^N \sum_{j=1}^M x_{ij} (1 - x_{ij})
 \end{aligned} \tag{4.19}$$

where $W_i, i = 1, \dots, 3$, are weighting coefficients. Function $\max(x, y)$ returns the larger one between (x, y) and $\min(x, y)$ finds the smaller value between (x, y) .

The W_1 term forces that every segment in system 2 is assigned to one and at most one segment in system 1. The W_2 term guarantees that all the segments of one carrier in system 2 are assigned to consecutive segments in system 1 in the same order [104]. The W_3 term is used to force neuron outputs to approach a corner of the hypercube, i.e., (0 or 1) [32].

From Equations (4.2), (4.4) and (4.19), the dynamic equation is:

$$\begin{aligned}
 y_{ij}(t+1) = & ky_{ij}(t) + \alpha \left[-W_1 \left(\sum_{j=1}^M x_{ij} - 1 \right) - W_2 \left(\sum_{\substack{p=1 \\ p \neq i}}^N \sum_{q=\max(j-c_p+1,1)}^{\min(j+c_i-1,M)} x_{pq} \right) - \frac{W_3}{2} (1 - 2x_{ij}) \right] \\
 & - z(t) \left[x_{ij}(t) - I_{ij}^{(0)} \right] + n(t)
 \end{aligned} \tag{4.20}$$

The neuron output will be converted from continuous to discrete as follows in Section 4.2.2. The NCNN-VT 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 assignment is obtained or the computation step exceeds the predefined maximum number of iteration steps (15000 in our simulations).

4.3.3 Simulation results and discussions

We implement the NCNN-VT algorithm in C language and simulate on a 16-node Dual Xeon 3.06 GHz (Intel IA32) Linux cluster with a parallel C/C++ compiler and a toolkit NPACI ROCKS v3.0.

The specifications of the 20 instances are listed in Table 4.6. Benchmarks BM 1 to BM 5 are from [104], where these were called instances 1-5, respectively. Benchmarks BM 6 to BM 8 are from [107], where these were called problems 3-5, respectively.

Cases 9-20 are newly randomly generated to evaluate the performance of the NCNN-VT in large-size FAP problems and are generated in the following steps. Firstly, we choose the number of carriers N and the number of segments M for the

instance. Next, we select the values of the range of carrier length and the range of interference between the two systems. In order to test the scalability of the NCNN-VT, we gradually increased these two parameters. And finally, we generate a set of carrier lengths c_i , ($i = 1, \dots, N$) and interference matrix E_I ($M \times M$) using uniformly distributed random values in the range that we have defined. We generate three groups of random instances as shown in Table 4.6. Group 1 (cases 9-11) is generated to observe the influence of the magnitude of the interference, by varying the interference while the number of carriers, the number of segments, and the range of carrier length are fixed. Group 2 (cases 12-17) is designed to show the effects of the carrier length. And group 3 (cases 18-20) is used to show the ability of the proposed NCNN-VT to deal with a large number of carriers.

The NCNN-VT approach has two types of parameters to tune: parameters for the NCNN-VT model and weighting coefficients in the energy function. We choose parameters for the NCNN-VT model such that the neural network produces rich and flexible neurodynamics. The choices of these parameters are similar to those used in other optimization problems [41, 59].

The selection of weighting coefficients is based on the rule that all terms in the energy function should be comparable in magnitude. We list the parameters used for each problem in this Chapter in Table 4.7. $I_{ij}^{(0)}$ is computed from (4.16).

Parameters listed here are empirical, and tuning these parameters is necessary for different problems. Our experiences show that the neural network model parameters do not vary much with problems, whereas the weighting coefficients (W_1, W_2, W_3)

Table 4.6: Specifications of the FAP instances used in the simulation.

Instance	Number of carriers N	Number of segments M	Range of carrier length	Range of interference
BM 1	4	6	1 - 2	5 - 55
BM 2	4	6	1 - 2	1 - 9
BM 3	10	32	1 - 8	1 - 10
BM 4	10	32	1 - 8	1 - 100
BM 5	10	32	1 - 8	1 - 1000
BM 6	18	60	1 - 10	1 - 100
BM 7	30	100	1 - 10	1 - 100
BM 8	15	50	1 - 8	1 - 1000
Case 9	50	200	1 - 10	1 - 10
Case 10	50	200	1 - 10	1 - 100
Case 11	50	200	1 - 10	1 - 1000
Case 12	80	200	1 - 5	1 - 100
Case 13	80	300	1 - 8	1 - 100
Case 14	80	400	1 - 10	1 - 100
Case 15	80	450	1 - 12	1 - 100
Case 16	80	500	1 - 14	1 - 100
Case 17	80	600	1 - 16	1 - 100
Case 18	100	500	1 - 10	1 - 100
Case 19	150	400	1 - 5	1 - 100
Case 20	200	300	1 - 2	1 - 100

Table 4.7: Parameter setting of the NCNN-VT approach for different instances, with $k = 0.9$, $\epsilon = 1/250$, $\alpha = 0.015$, $z(0) = 0.08$, $\beta_1 = 0.001$, and $W_1 = 1.0$.

Instance	W_2	W_3	$A[n(0)]$	β_2
1	1.0	0.7	0.02	0.001
2	1.0	0.7	0.02	0.001
3	0.3	0.7	0.02	0.001
4	0.3	0.7	0.02	0.001
5	0.4	0.7	0.02	0.0001
6	0.2	0.6	0.02	0.0001
7	0.2	0.6	0.01	0.0001
8	0.1	0.3	0.02	0.0001
9	0.2	0.4	0.02	0.0001
10	0.2	0.6	0.02	0.0001
11	0.2	0.6	0.01	0.0001
12	0.2	0.4	0.01	0.0001
13	0.2	0.4	0.01	0.0001
14	0.1	0.4	0.01	0.0001
15	0.1	0.5	0.01	0.0001
16	0.1	0.5	0.01	0.0001
17	0.1	0.5	0.01	0.0001
18	0.1	0.6	0.02	0.0001
19	0.1	0.6	0.02	0.0001
20	0.2	0.4	0.02	0.0001

can be slightly more sensitive to problems.

We run the NCNN-VT on each instance 1000 times with different randomly generated initial neuron states. We end the neuron update when a valid solution is found, i.e., when E_1 and E_2 vanish. Table 4.8 shows results, including the best largest interference, the percentage to reach the optimum (Opt rate), the average error from the optimal result, and the total interference when the optimum of the largest interference is found. The average numbers of iteration steps and standard deviations are also shown in this Table. The percentage at which the NCNN-VT reaches the optimum is evaluated only in the runs in which valid solutions are

Table 4.8: The performance of the NCNN-VT on 20 instances. “Opt rate” stands for the percentage at which the NCNN-VT reached the best known result in the 1000 runs. “SD” stands for “standard deviation”.

Instance	Largest interference			Total interference	Iterations mean±SD
	Best	Opt rate %	Average error		
1	30	100	0	100	378±16.8
2	4	100	0	13	478±22.6
3	7	100	0	85	1901±96.5
4	64	88.3	1.74	880	2157±252
5	640	37.1	37.3	7246	1819±156
6	35	25.8	4.78	1000	2016±484
7	61	10.4	18.1	2779	2891±465
8	695	26.6	182	15373	3769±402
9	9	73.8	0.8	618	4019±359
10	85	64.8	5.9	5187	4245±363
11	797	38.4	113.9	48951	4281±357
12	70	45.5	5.8	3418	4192±303
13	73	63.7	3.4	6183	4315±369
14	93	49.7	4.2	10261	4126±311
15	89	67.2	3.3	12475	4655±327
16	81	29.8	6.8	14272	4558±310
17	91	42.6	8.6	20731	4820±329
18	87	72.5	9.2	14629	5671±485
19	85	46.4	13.9	9836	5281±461
20	36	76.3	6.2	14264	5649±716

found. Furthermore, as shown in this Table, the NCNN-VT finds a feasible solution in nearly a constant number of iteration steps.

To present the effect of the convergence term $\frac{w_3}{2} E_3 = \frac{w_3}{2} \sum_{i=1}^N \sum_{j=1}^M x_{ij}(1 - x_{ij})$, we simulated NCNN-VTs without this term on case 3 and compared the results in Table 4.9. From the results, we can see this 0/1 term do help the NCNN-VT to converge and improved its performance on the FAP in terms of the rate of getting optima.

Table 4.9: Comparison on the effect of the 0/1 term in the energy function for the FAP.

	Largest interference			Total interference	Iterations mean	Iterations SD SD
	Best	Opt rate %	error			
$W_3 = 0$	7	28	1.47	85	3790	527
$W_3 = 0.3$	7	82.1	0.25	85	2485	175
$W_3 = 0.7$	7	100	0	85	1901	96.5

Table 4.10: Comparison of largest interference and total interference obtained by the NCNN-VT, TCNN-VT, GNN and HopSA for benchmarks 1 to 8. Results are shown as the best, average values and standard deviation (Best/ Ave/ SD).

Instance	GNN [104]			HopSA [107]			TCNN-VT			NCNN-VT		
	Largest (Best/ Ave)	Total (Best/ Ave)		Largest Best	Total Best		Largest (Best/ Ave)	Total (Best/ Ave)		Largest (Best/ Ave / SD)	Total (Best/ Ave / SD)	
BM 1	30/ 31.5	100/ 100.8		30	100		30/ 31.5	100/ 105		30 / 30 / 0	100 / 100 / 0	
BM 2	4/ 4.9	13/ 15.4		4	13		4/ 4.8	13/ 15.4		4 / 4 / 0	13 / 13.7 / 1.2	
BM 3	7/ 8.1	85/ 99.4		7	85		7/ 8.4	96/ 130.6		7 / 7 / 0	85 / 89.9 / 3.8	
BM 4	64/ 77.1	880/ 982.0		64	888		70/ 94.1	828/ 1145		64 / 65.7 / 2.6	880 / 903.6 / 35.1	
BM 5	640/ 766.8	8693/ 9413.9		817	6910		661/ 849	6910/ 9527		640 / 677 / 71.1	7246 / 8445 / 656.7	
BM 6	49	1218		45	1080		NA	NA		35 / 39.8 / 4.8	1000 / 1068 / 40.5	
BM 7	100	4633		98	3396		97/ 98.1	5839/ 5722		61 / 79.9 / 7.8	2779 / 2995 / 124	
BM 8	919	16192		741	13178		NA	NA		695 / 877 / 64.8	15373 / 18034 / 318	

Table 4.10 compares results on the largest and total interference obtained by the NCNN-VT, TCNN-VT, the GNN [104], and the HopSA [107]. Results of the GNN are from references [104, 107]. Results of the HopSA on benchmarks 1-8 are also from the reference [107]. Since [107] did not present the average value of the largest and total interference, only the best value is included in Table 4.10. For instances 9-20, we re-run the HopSA algorithm from the authors of [107]. For benchmark problems from BM 1 to BM 8, the NCNN-VT matches or improves results of the GNN and HopSA. Compared with the GNN, the best result computed by the NCNN-VT are the same for BM 1 to BM 4, but the NCNN-VT is more capable to reach this best known, hence offers further improvements over the GNN. For problems BM 5 to BM 8, the NCNN-VT obtains better results both in terms of the largest interference and the total interference. Compared with the HopSA, the NCNN-VT obtains smaller largest interference than the HopSA [107] did in all the benchmarks while maintaining comparable total interference. Between the two optimization objectives of the FAP, minimization of the largest interference is critical and has a higher priority over minimization of the total interference [104].

We further compare the HopSA with the NCNN-VT on randomly generated instances 9-20. We simulate the HopSA algorithm under the same simulation environment as the NCNN-VT. Results are summarized in Tables 4.11 and 4.12, which show that the HopSA needs several hours of computation for a 50-carrier-200-segment problem. The HopSA fails to obtain solutions in instances with larger size due to the excessive computation time. “N/A” in Table 4.11 represents situations that

Table 4.11: Comparisons of the largest interference and total interference on randomly generated instances between the HopSA [107] and the NCNN-VT. Results are displayed as mean±SD (standard deviation).

Instance	HopSA		NCNN-VT	
	Largest	Total	Largest	Total
9	10.5±0.6	658.3±15.3	9.8±0.4	627.3±12.9
10	124.3±10.2	6533±123.5	90.9±5.8	5374±60.4
11	989.2±73.5	49982±150.5	910.9±64.4	49388±120.2
12	N/A	N/A	75.8±2.5	3437±74.3
13	N/A	N/A	76.0±11.4	6295±151.8
14	N/A	N/A	97.2±0.8	10827±351.2
15	N/A	N/A	92.3±6.4	12644±273.7
16	N/A	N/A	87.0±8.4	14385±283.8
17	N/A	N/A	99.7±0.7	21894±776.4
18	N/A	N/A	96.9±3.1	14841±628.9
19	N/A	N/A	98.8±1.8	9924±387.9
20	N/A	N/A	42.9±22.3	14488±434.3

the algorithm cannot find solutions in one month. In comparison, our proposed NCNN-VT finds a feasible solution in less than one hour of CPU time on all the instances.

We listed the number of iteration steps and computation time of the NCNN-VT and the HopSA [107] on all the instances in Table 4.12 together with the convergence rate (the ratio at which the neural network finds a feasible solution). We consider the algorithm is non-convergent if a solution is not found in 15000 steps. One iteration step in the NCNN-VT means one loop that all neurons are updated. The computation time is measured in seconds (Sec.). Also, the computation time is calculated only for runs in which feasible solutions are found. Table 4.12 shows

Figure 4.4: Computation time of the HopSA and NCNN-VT for the 20 instances.

that our NCNN-VT achieves at least a 78% convergence rate. Parameter selection and instance complexity affect the convergence rate. We attempted to adjust the parameters so as to improve the convergence rates; however, it was not possible to achieve 100% for some large problems (see Table 4.12). This may be due to the high complexity in the large problems.

We plot the computation time of the HopSA and the NCNN-VT in Figure 4.4, which shows that compared to the HopSA, the computation time of the NCNN-VT increases more slowly as the problem size increases. As the HopSA needs several hours to find a solution for instances 9 to 11, the NCNN-VT needs no more than 200 seconds.

We have also solved the same FAPs using the TCNN and NCNN, with the energy function which consists of both the constraint terms and the objective term [26]:

$$E_4 = W_4 \sum_{i=1}^N \sum_{j=1}^M d_{ij} x_{ij} \quad (4.21)$$

Table 4.13 shows the comparison between the TCNN [111], NCNN [45], and NCNN-VT on benchmarks BM 1-8. For each algorithm on each instance, we list

Table 4.12: Comparison between the HopSA and the NCNN-VT on computation time in seconds and convergence rate. Results are displayed as mean±SD (standard deviation). η stands for the convergence rate.

	HopSA		NCNN-VT	
#	Time (Sec.) mean±SD	η (%)	Time (Sec.) mean±SD	η (%)
1	1.0±0.0	100	0.02±0.1	100
2	1.0±0.0	100	0.02±0.1	100
3	33.0±0.0	100	0.31±0.04	100
4	33.8±0.8	100	0.36±0.05	100
5	33.0±0.5	100	0.37±0.04	100
6	1529±156	100	11.8±0.6	93.6
7	2845±528	100	78.3±13.3	86.0
8	3738±615	75	143.2±12.5	80.8
9	5389±823	80	133.6±2.3	96.6
10	8916±1529	70	172.7±5.8	100
11	15544±2117	70	187.5±7.3	100
12	N/A	N/A	271.8±27.1	100
13	N/A	N/A	1034.0±56.1	93
14	N/A	N/A	1655±207.2	91
15	N/A	N/A	1852±353.9	100
16	N/A	N/A	1959±118.1	79
17	N/A	N/A	2621±118.4	78
18	N/A	N/A	2989±200.7	80
19	N/A	N/A	3550±233.3	85
20	N/A	N/A	2477±14.0	88

the convergence rate η , the average number of iteration steps T , the best and average values of the largest and total interference, respectively. It is shown that the NCNN-VT needs fewer iterations compared with the TCNN and NCNN. In addition, the NCNN-VT achieved solutions with smaller largest interference and total interference. Furthermore, the NCNN-VT has higher convergence rate compared to the NCNN and TCNN. The advantage of the NCNN-VT grows as the problem size increases.

Table 4.13: Comparison of the TCNN, NCNN, TCNN-VT and NCNN-VT on benchmarks BM 1-8 in 1000 runs. # denotes the instance number. η is the convergence rate. T is the average number of iteration steps. I_L is the largest interference and I_T is the total interference. The interference is shown as the best and average values (Best/ Ave).

#	TCNN			NCNN			TCNN-VT			NCNN-VT		
	η (%)	T	I_L Best/Ave	I_T Best/Ave	η (%)	T	I_L Best/Ave	I_T Best/Ave	η (%)	T	I_L Best/Ave	I_T Best/Ave
1	100	426.5	30/ 35.4	105/ 112.6	100	393.2	30/ 32.0	100/ 100.0	100	491	30/ 31.5	100/ 105
2	100	829.4	6/ 7.6	17/ 21.7	100	766.4	6/ 6.8	17 / 18.2	100	799	4/ 4.8	13/ 15.4
3	92.4	2485	8/ 10.9	112/ 142.2	98.8	2242	9/ 9.9	114/ 163	94.8	2351	7/ 8.4	96/ 130.6
4	87.5	2383	84/ 96.2	971/ 1145	97.3	2069	78/ 97.1	1025 /1444	89.3	2177	70/ 94.1	828/ 1145
5	61.0	2342	697/ 871	7574/ 9927	92.6	1994	674/ 945	9279/ 11613	86.6	3075	661/ 849	6910/ 9527
6	53.7	2651	42/ 46.8	1154/ 1491	80.0	2115	47/ 49.1	1057/ 1231	NA	NA	NA	NA
7	67.1	3716	91/ 97.3	3296/ 3826	85.9	3225	88/ 95.2	3173/ 3501	77.1	2528	78/ 83.9	3296/ 3826
8	52.2	4839	836/ 869	17659/ 20493	70.1	4021	800/ 835	16321/ 17342	NA	NA	NA	NA
									80.8	3769	695/ 877	15373/ 18034

The energy functions of the NCNN and TCNN have an additional term, i.e., optimization term E_4 , which is needed to fulfill the optimization objective of the FAP. With this optimization term in the energy function, the tuning of weighting coefficients will be more difficult because the E_1 to E_3 are much smaller compared to E_4 (Eqn. (4.21)), which makes the final solution very sensitive to the choice of W_4 . The magnitude of E_4 is related to cost matrix D which is problem-specific. Hence E_4 may vary greatly when the range of interference in the simulation changes from 1 – 100 to 1–1000. On the contrary, if we separate the optimization term from constraint terms as in our NCNN-VT approach, the balance among the remaining terms in the energy function and the selection of weighting coefficients become simpler and easier.

4.4 Conclusions

We propose a new approach, i.e., *transiently chaotic neural network with variable thresholds (TCNN-VT)* by varying the threshold in the self-feedback term of the TCNN model. With rich and complex dynamics, the new TCNN-VT maintains the advantage of the TCNN. We solve the FAP in satellite communications with the TCNN-VT. To minimize the largest interference, the neuron is designed to have a larger threshold if this neuron presents an assignment with smaller interference. The simulation results on 8 benchmark instances demonstrate that the TCNN-VT can find better solutions than existing algorithms.

Similarly a *noisy chaotic neural network with variable thresholds (NCNN-VT)* is used to solve the FAP in satellite communications. We let a neuron of the NCNN-

VT to have a larger bias input (the negative of the neuronal threshold) when the neuron presents a frequency assignment with smaller interference, so that the neuron is more likely to be selected for a frequency assignment. With this mapping scheme, optimization objectives of the problem are achieved by variable thresholds of the neural network, while the energy function is in charge of only constraints. As a result, weight tuning in the energy function becomes easier.

The objective of this NP-complete optimization problem is to minimize cochannel interference between two satellite systems by rearranging frequency assignments. The NCNN-VT model consists of $N \times M$ noisy chaotic neurons for an N -carrier- M -segment problem. The NCNN-VT facilitates interference minimization by mapping the objective to variable thresholds (biases) of the neurons.

We used a linear mapping of thresholds to separate the optimization objectives from constraints, thereby simplifying the formulation of the energy function and selection of weighting coefficients. The objective mapping scheme achieves the objectives of the FAP, and the update rule for neurons helps to find a feasible solution satisfying constraints through the complex neurodynamics (stochastic noise and flexible chaos) of the NCNN-VT model. The performance of the NCNN-VT is demonstrated through solving a set of benchmark problems and randomly generated test instances. Simulation results on the 20 instances show that the NCNN-VT obtains solutions with low computational cost compared to the previous methods. The NCNN-VT offers further improvements compared with the GNN [104] on the benchmark examples. Compared to the HopSA [107], the NCNN-VT finds optimal

or sub-optimal solutions with less computation cost. Furthermore, the NCNN-VT needs fewer iterations compared to transiently chaotic neural networks (TCNN) [2] and noisy chaotic neural networks (NCNN) [45].

Compared with other techniques, the neural network is a general tool for combinatorial optimization problems. Evolutionary computation as another kind of stochastic search, includes those techniques which are inspired by biology. There are evolutionary programming, genetic algorithms, evolutionary strategies. However in the literature review, we did not find any applications of evolutionary computation on the FAP in satellite communications.

Chapter 5

Transiently Chaotic Cellular Neural Networks

5.1 Introduction

Many modern systems developed these years are based on artificial intelligence and soft computing, where artificial neural networks play an important role, together with fuzzy logic, evolutionary computation, and chaos. Chua and Yang proposed a circuit architecture, called cellular neural networks (CNNs) [47, 48], which possess the ability to do parallel signal processing in real time. Any cell in a CNN is connected only to its neighboring cells. The r -neighborhood of neuron (i, j) ($1 \leq i \leq M, 1 \leq j \leq N$) is defined as $N_r(i, j) = \{(k, l) | \max\{|k - i|, |l - j|\} \leq r\}$, $1 \leq k \leq M, 1 \leq l \leq N$, shown in Fig. 5.1. This property of nearest neighbor interactions makes CNN much more amenable to VLSI implementation compared to general neural networks. Some rather promising applications of CNNs in image processing, communication systems and optimization problem have been reported in [114–122].

Figure 5.1: (a) A 2-dimensional 4-by-4 cellular neural network: the nine shaded squares are the r -neighborhood cells of the black cell $C(i, j)$ when $r = 1$; (b) The piecewise-linear activation function of CNNs.

Grassi [28] designed a discrete-time cellular neural network (DTCNN) which is globally asymptotically stable to behave as associative memories. Fantacci et al [123] exploited CNNs' capability to account for the optimization problem by implementing of a cell scheduling algorithm. They also showed that for a class of optimization problems, the performance of CNNs is comparable to the HNN [108]. Nakaguchi et al [124] proposed an architecture based on the hysteresis CNN and applied it on the N-Queen problem, a classic combinatorial optimization problem. The results showed the effectiveness of this network architecture.

Many problems in science and technology are combinatorial optimization problems, with energy or cost functions to be minimized under some constraints. Extensive research effort has gone into developing efficient techniques for finding minimum values of energy functions. Several methods such as stochastic simulated annealing [36], chaotic simulated annealing (CSA) with decaying self-connection [2], Wang and Smith's CSA with decaying time step [41] etc, have been proposed to improve the performance.

To solve combinatorial optimization problems (COP) efficiently, chaotic dynamics has been studied widely after the work of Nozawa [37], Chen and Aihara [2]. Nozawa modified Hopfield network by adding negative self-feedback connection, analyzed one-dimensional maps for the single neuron model under different control parameters, and demonstrated the existence of chaos. Two engineering applications were also presented to show the effectiveness of the new model. Chen and Aihara proposed the TCNN, which acted better in searching for globally optimal or near-optimal solutions compared with the conventional Hopfield-type neural network. Some found that chaotic dynamics are more effective for solving optimization problems than stochastic dynamics. They did several experiments on different applications to compare chaos and noise, and asserted that although a general judgment has not been formulated; chaotic dynamics often works better than random signals. To utilize the advantages of chaotic dynamics and the promising processing capability of CNNs, we intend to introduce chaotic dynamics into the CNN in this Chapter.

The complex dynamics in CNNs have already been reported in the past years. Chen et al [49] proposed a discrete-time cellular neural network (DTCNN) with neuronal input-output described by the logistic function; however, this type of neurons are rarely used in the literature. In a CNN with time-delays, Lu et al [125] investigated complex dynamics by simply adding a piecewise-linear delayed feedback to the linear system. Civalleri and Gilli [126] reviewed and discussed the stability of the original CNN model, including complete stability, stability almost everywhere

and global asymptotic stability. Zou and Nossek observed a chaotic attractor in a two-cell non-autonomous CNN [127]. They also presented bifurcation phenomena and chaotic attractor in autonomous CNNs but with space variant templates [50]. Gilli [128] analyzed a delayed cellular neural network (DCNN) which consist of 2 cells and then presented strange attractors generated by different delays. Gilli et al [129] explored the dynamic behavior in autonomous space-invariant CNNs. They showed limit cycle and chaotic attractor in a CNN composed by 9 (3×3) cells. Petras et al [130] found that boundary conditions will affect the stability of CNNs when off-diagonal template elements have opposite-signs. Biey, Gilli and Checco [131] investigated equilibrium point bifurcation and other complex dynamics in a first-order autonomous space-invariant CNN.

Analog realizations of CNNs have been widely discussed [132–135]. For example, Harrer and Nossek [132] designed and implemented a high-density layout for the DTCNN. Chou et al [133] proposed a mode chip with hardware annealing as a highly efficient method of finding globally optimal solutions with cellular neural networks. Linan et al [134] presented an analog programmable visual microprocessor chip, i. e., the *ACE4k*, which can work as a cellular neural network universal machine [136] and process complex spatio-temporal images in parallel. Rodriguez-Vazquez et al improved ACE chips as they designed and presented an *ACE16k* [135] with better performance, more functionalities, and lower power consumption.

In order to take advantage of both efficient search with chaos [51, 52] and the mature technology of VLSI implementation of cellular neural networks [137–139],

we add negative self-feedbacks into the Euler approximation of the continuous CNN model and demonstrate a variety of dynamic behaviors, including fixed points, periodic oscillations, and chaos, thereby creating a new *chaotic cellular neural network* (C-CNN). This new model shares the best features of both worlds: It has complex dynamics so possesses higher ability to search for optimal solutions for optimization problems; And furthermore, the local interconnection feature of CNNs made it especially suited for large scale analog implementation. Then we extend C-CNN to a *transiently chaotic cellular neural network* (TC-CNN) by decaying the self-feedback. Compared to existing chaotic CNNs, our model uses conventional neurons and is relatively easy to analyze mathematically when the network is large.

This chapter is organized as follows. We first review the cellular neural network in Section 5.2. Then in Section 5.3, the chaotic cellular neural network (C-CNN) is proposed by adding negative self-feedback to the Euler approximation of the continuous CNN. Then we extend C-CNN to the transiently chaotic cellular neural network. In Section 5.4, the stability analysis of the network is presented. The simulation result of a single neuron model is given in Section 5.5. Finally, we conclude this chapter in Section 5.6.

5.2 Cellular neural networks

The CNN can perform parallel real-time signal processing due to the continuous time feature, and on the other hand, it is especially suitable for very large scale integration (VLSI) implementation based on their local interconnection feature. Some rather

promising applications of CNNs in image processing and pattern recognition have been reported [140].

A CNN is completely characterized by the set of nonlinear differential equations below. The dynamics of a CNN includes both output feedback and input control mechanisms. Consider the realization in the circuits, the system equations of an $M \times N$ cellular neural network [47] can be described as follows:

$$C \frac{dx_{ij}(t)}{dt} = -\frac{1}{R_x} x_{ij}(t) + \sum_{C(k,l) \in N_r(i,j)} A(i,j;k,l) y_{kl}(t) + \sum_{C(k,l) \in N_r(i,j)} B(i,j;k,l) u_{kl} + I \quad (5.1)$$

where

x_{ij} = Internal state of neuron (i,j) ,

y_{ij} = Output of neuron (i,j) ,

u_{ij} = Input of neuron (i,j) ,

$A(i,j;k,l)$ = The output feedback parameter,

$B(i,j;k,l)$ = The input control parameter,

$N_r(i,j)$ = The r -neighborhood of a cell $C(i,j)$,

I = An independent voltage source functioned as a constant bias,

C = A linear capacitor,

R_x = A linear resistor.

The activation function of neuron (i,j) is a piecewise linear function, which is the

only nonlinear element in a neuron. The function is realized by a voltage-controlled current source:

$$y_{ij}(t) = \frac{1}{2}(|x_{ij}(t) + 1| - |x_{ij}(t) - 1|) \quad (5.2)$$

In the field of circuits and systems, Piecewise Linear (PWL) Functions is widely used. A PWL function is a combination of linear-affine functions. Hence PWL functions are not only capable of representing nonlinear models but also suitable for systematic methodologies [141].

Constrained conditions are:

$$|x_{ij}(0)| \leq 1, \quad 1 \leq i \leq M; 1 \leq j \leq N.$$

$$|u_{ij}| \leq 1, \quad 1 \leq i \leq M; 1 \leq j \leq N.$$

The network model described above is proven to be stable [47, 122] when the space templates are symmetrical i.e. $A(i, j; k, l) = A(k, l; i, j)$ and the self-feedback $A(i, j; i, j) > 1$.

A lot of works have been done to study the complex dynamics in the CNN, including the bifurcation phenomena and chaotic behavior [49, 50]. Chen et al [49] presented the bifurcation and chaos in the discrete-time cellular neural network (DTCNN) using locally coupled cells and logistic equations as output equations, however, the logistic equation is rarely used as an activation function due to its I/O property. In a kind of delayed cellular neural networks, these dynamics have been investigated by simply adding a piecewise-linear delayed feedback to the original

linear system [125]. These bifurcation phenomena and chaos have also been showed in small autonomous networks with only two or three cells [50].

The practical analogue realization of some kinds of CNN in discrete time have already been widely discussed [132, 142]. In [132] an architecture has been proposed and a layout has been designed then realized, whose result showed the circuit holds a high density. A mode chip with hardware annealing has been proposed as a highly efficient method of finding globally optimal solutions for cellular neural networks [133] in an edge detection problem.

5.3 Transiently chaotic cellular neural networks

Usually there are positive self-feedbacks in the CNN, i.e., $A(i, j, i, j) \geq 0$. Let us change it to negative by adding a negative self-feedback term for each neuron and take the difference equation version of equation (5.1) by Euler's method [38], then

the model becomes

$$\begin{aligned}
 x_{ij}(t+1) &= x_{ij}(t) + f(x_{ij}(t))\Delta t \\
 &= \left(1 - \frac{\Delta t}{CR_x}\right)x_{ij}(t) + \frac{\Delta t}{C} \sum_{C(k,l) \in N_r(i,j)} A(i,j;k,l)y_{kl}(t) \\
 &\quad + \frac{\Delta t}{C} \sum_{C(k,l) \in N_r(i,j)} B(i,j;k,l)u_{kl} + \frac{\Delta t}{C} I - z[y_{ij}(t) - I_0] \\
 &= p x_{ij}(t) - z[y_{ij}(t) - I_0] + \sum_{C(k,l) \in N_r(i,j)} a(i,j;k,l)y_{kl}(t) \\
 &\quad + \sum_{C(k,l) \in N_r(i,j)} b(i,j;k,l)u_{kl} + i
 \end{aligned} \tag{5.3}$$

$$y_{ij}(t) = \frac{1}{2} (|x_{ij}(t) + 1| - |x_{ij}(t) - 1|) \tag{5.4}$$

where

x_{ij} = Internal state of neuron (i, j) ,

y_{ij} = Output of neuron (i, j) ,

u_{ij} = Input of neuron (i, j) ,

$A(i, j; k, l)$ = The output feedback parameter,

$B(i, j; k, l)$ = The input control parameter,

$C(i, j)$ = The neuron (i, j) ,

$N_r(i, j)$ = $\{(k, l) | \max\{|k - i|, |l - j|\} \leq r,$

$1 \leq k \leq M, 1 \leq l \leq N\}$

The r -neighborhood of neuron (i, j) ,

$a(i, j; k, l) = A(i, j; k, l) \frac{\Delta t}{C},$

$b(i, j; k, l) = B(i, j; k, l) \frac{\Delta t}{C},$

$i = I \frac{\Delta t}{C},$

I = An independent voltage source,

$p = 1 - \frac{\Delta t}{CR_x},$

z = Self-feedback connection weight,

I_0 = A positive bias factor,

C = A linear capacitor,

R_x = A linear resistor.

Figure 5.2: Block diagram of a chaotic cellular neural network.

In the conventional CNN, there are usually positive self-feedbacks, i.e., $A(i, j, i, j) \geq 0$. When the feedback is positive, it has a stabilizing effects [143] and is equivalent to a hysteresis. When the feedback is negative, oscillations and chaos can occur, depending on the magnitude of the negative feedback, similar to the TCNN investigated by Chen and Aihara [2]. Unstable dynamic behaviors will occur if the magnitude of the negative self-feedback is sufficiently large. We may change the chaotic system to transiently chaotic system by gradually removing the negative self-feedback. The self-feedback can be reduced in any schedule, which may be designed for different problems. Here we set it to decay exponentially:

$$z(t+1) = (1 - \beta)z(t) \quad (5.5)$$

where β ($0 \leq \beta \leq 1$) is the damping factor of the time-dependent self-feedback $z(t)$.

Combining (5.3) with the activation function and the self-feedback decaying func-

tion, we obtain the definition of the new model:

$$x_{ij}(t+1) = p x_{ij}(t) + \sum_{C(k,l) \in N_r(i,j)} a'(i,j;k,l) y_{kl}(t) + \sum_{C(k,l) \in N_r(i,j)} b(i,j;k,l) u_{kl} + i' \quad (5.6)$$

$$y_{ij}(t) = \frac{1}{2} (|x_{ij}(t) + 1| - |x_{ij}(t) - 1|) \quad (5.7)$$

$$z(t+1) = (1 - \beta) z(t) \quad (5.8)$$

where

$$a'(i,j;i,j) = a(i,j;i,j) - z(t) \quad (5.9)$$

$$i' = i + z(t) I_0 \quad (5.10)$$

To compare with the continuous-time cellular neural network, the new TC-CNN model in (5.6)-(5.8) would correspond to the following differential equations:

$$\begin{aligned} \frac{dx_{ij}(t)}{dt} = & -z_{ij}(t)(y_{ij}(t) - I_0) + p x_{ij}(t) + \sum_{C(k,l) \in N_r(i,j)} A(i,j;k,l) y_{kl}(t) \\ & + \sum_{C(k,l) \in N_r(i,j)} B(i,j;k,l) u_{kl} + I \end{aligned} \quad (5.11)$$

$$y_{ij}(t) = \frac{1}{2} (|x_{ij}(t) + 1| - |x_{ij}(t) - 1|) \quad (5.12)$$

$$\frac{dz_{ij}}{dt} = -\beta z_{ij} \quad (5.13)$$

5.4 Stability of the TC-CNN

We use the following energy function for the TC-CNN:

$$\begin{aligned}
 E(t) = & -\frac{1}{2} \sum_{(i,j)} \sum_{(k,l)} a'(i,j;k,l) y_{ij}(t) y_{kl}(t) - \sum_{(i,j)} \sum_{(k,l)} b(i,j;k,l) y_{ij}(t) u_{kl} \\
 & - \sum_{(i,j)} i' y_{ij}(t) + \frac{1-p}{2} \sum_{(i,j)} y_{ij}(t)^2
 \end{aligned} \tag{5.14}$$

The first three terms are from the energy function for conventional continuous-time CNNs [47]. We modified the last term according to [41,144] for the new discrete-time model.

The difference between two time steps in the energy function is:

$$\begin{aligned}
 \Delta E = & -\frac{1}{2} \sum_{(i,j)} \sum_{(k,l)} a'(i,j;k,l) \Delta y_{ij}(t) \Delta y_{kl}(t) - \sum_{(i,j)} \sum_{(k,l)} a'(i,j;k,l) y_{kl}(t) \Delta y_{ij}(t) \\
 & - \sum_{(i,j)} \Delta y_{ij}(t) \left[\sum_{(k,l)} b(i,j;k,l) u_{kl} + i' \right] \\
 & + \frac{1-p}{2} \sum_{(i,j)} \Delta y_{ij}(t) [y_{ij}(t+1) + y_{ij}(t)]
 \end{aligned} \tag{5.15}$$

According to the cell circuit Equation (5.6), Equation (5.15) can be written as:

$$\begin{aligned}
 \Delta E = & -\frac{1}{2} \sum_{(i,j)} \sum_{(k,l)} a'(i,j;k,l) \Delta y_{ij}(t) \Delta y_{kl}(t) - \sum_{(i,j)} \Delta y_{ij}(t) [x_{ij}(t+1) - p x_{ij}(t)] \\
 & + \frac{1-p}{2} \sum_{(i,j)} \Delta y_{ij}(t) [y_{ij}(t+1) + y_{ij}(t)]
 \end{aligned} \tag{5.16}$$

From the output function given in Equation (5.7), we have:

$$y_{ij}(t) = x_{ij}(t), \text{ when } |x_{ij}(t)| < 1$$

$$\Delta y_{ij}(t) = 0, \text{ when } |x_{ij}(t)| \geq 1$$

then

$$\begin{aligned} \Delta E &= -\frac{1}{2} \sum_{|x_{ij}| < 1} \sum_{|x_{kl}| < 1} a'(i, j; k, l) \Delta y_{ij}(t) \Delta y_{kl}(t) - \sum_{|x_{ij}| < 1} \Delta y_{ij}(t) [y_{ij}(t+1) - p y_{ij}(t)] \\ &\quad + \frac{1-p}{2} \sum_{|x_{ij}| < 1} \Delta y_{ij}(t) [y_{ij}(t+1) + y_{ij}(t)] \\ &= -\frac{1}{2} \sum_{|x_{ij}| < 1} \sum_{|x_{kl}| < 1} [a'(i, j; k, l) + (1+p)\delta_{ik}\delta_{jl}] \Delta y_{ij}(t) \Delta y_{kl}(t) \end{aligned} \quad (5.17)$$

Therefore, according to Lyapunov theorem, if the matrix $[a'(i, j; k, l) + (1+p)\delta_{ik}\delta_{jl}]$ is positive-definite, i. e., $\Delta E \leq 0$, then the network is stable. Hence a sufficient stability condition for the TC-CNN is

$$a'(i, j; i, j) > -(1+p) = \frac{\Delta t}{CR_x} - 2 \quad (5.18)$$

where CR_x is the time constant of the dynamics of the circuit [47]. Substituting the equation (5.9) into equation (5.18):

$$a(i, j; i, j) - z(t) > -(1+p) \quad (5.19)$$

If $p = 0.7$, based on Equation (5.18), the system is stable and converges to a fixed point when

$$a(i, j; i, j) - z(t) > -(1+p) = -1.7$$

These facts are verified by the simulation result shown in Fig.5.8.

5.5 Simulation results

5.5.1 Chaotic cellular neural network

Deriving from (5.6)-(5.7), we obtain the single neuron model for the chaotic cellular neural network:

$$x(t+1) = -z(y(t) - I_0) + p x(t) + A y(t) + B u + I \quad (5.20)$$

$$y(t) = \frac{1}{2} (|x(t) + 1| - |x(t) - 1|) \quad (5.21)$$

In the following, we vary only the value of z to investigate the dynamics of the network while other parameters are fixed in Eqn.(5.20) as $p = 0.999$, $A = 0.1$, $Bu + I = 0$, $I_0 = 0.25$. Fig.5.3 shows the time evolutions for $y(t)$ when $z = 0.7, 2.0, 2.15, 5$, respectively, and the initiative value for neuron state x is randomly generated in $[-1, 1]$.

We observe the output of the neuron in 30 iteration steps. When $z < 2$, there is only one stable equilibrium for the output. When $z = 2$ the process begins oscillating between two different points (0.6 and -0.1) without converging to either (period-2 oscillations). When $z = 2.15$ the output oscillates between three different points (1, -0.4 and -0.6) forming a period-3 output - "Period-3 implies chaos" [145]. When z is larger, the output has chaotic dynamics.

Figure 5.3: Same template values can produce various dynamics with different self-feedback for C-CNN (a) stable dynamic with $z=0.7$; (b) period-2 oscillations with $z=2$; (c) period-3 oscillations with $z=2.15$; (d) chaotic dynamics with $z=5$.

5.5.2 Thresholds analysis of the C-CNN

We investigate the effect of variable thresholds in the C-CNN. We show that this threshold cannot be too large if one wishes to produce chaotic dynamics in the C-CNN, which is important to studies of chaotic communication and combinatorial optimization problem. We are particularly interested in variable thresholds because

Shi and Wang [106] showed that the objectives of the frequency assignment problem (FAP) can be mapped into thresholds of the neural network, which resulted in superior performance compared to traditional penalty approaches. Shi and Wang [106] showed that thresholds of the noisy chaotic neural network (NCNN) [45] can be related to the objective of an optimization problem. They defined the threshold so that the neuron with a smaller cost is more likely to be chosen for frequency assignments in satellite communication systems. Hence variable thresholds separate the optimization objective term from the constraint terms in the energy function and helps the network model to reach the optimal solution. They showed that NCNN with variable thresholds out-performed traditional penalty approaches.

Nonlinear dynamics is applied in various disciplines, such as biology, economics, physiology, and engineering. Andreyev et al [46] explored information processing capabilities of chaos and complex dynamics. They found that a model with chaotic dynamics is efficient in storage, retrieval, and recognition of information and has low computation cost. Aihara [51] has argued that chaotic dynamics has many possible useful functions. To model dynamics of the biological brain, more complex system is needed. Aihara had concluded that synchronous and asynchronous models with complex dynamics may provide parallel distributed processing which will be complementary to the present digital computers.

For the C-CNN, I_0 may affect chaotic dynamics. We simulate a single-neuron model of the C-CNN by employing the first order Euler approximation with $\Delta t = 0.0001$. From (5.20), the first order Euler approximation of a single-neuron model,

where $p = 1 - \Delta t$, we analyze the performance of various I_0 . In the simulation, the initial value for neuron state $x(0)$ is randomly generated in $[-1, 1]$ and the other parameters are set to [146]:

$$A = 1, \quad Bu = 0, \quad I = 0, \quad z = 5.$$

Lyapunov exponent [147] is computed to identify the deterministic chaos, which is defined in equation (5.22). For each fixed value of I_0 from 0.01 to 1 with an interval 0.01, we compute the Lyapunov exponent of the C-CNN dynamics with equations (5.6), (5.7), and (5.22) for sufficiently large iterations ($m = 10000$).

$$\lambda = \lim_{m \rightarrow +\infty} \frac{1}{m} \sum_{t=0}^{m-1} \ln \left| \frac{dx(t+1)}{dx(t)} \right| \quad (5.22)$$

Considering that the activation function, i.e., equation (5.7), can be written as:

$$y(t) = \begin{cases} 1 & 1 < x(t) \\ x(t) & -1 \leq x(t) \leq 1 \\ -1 & x(t) < -1 \end{cases} \quad (5.23)$$

From equations (5.20) and (5.23) we have:

$$\frac{dx(t+1)}{dx(t)} = \begin{cases} p & 1 < x(t) \\ p - z + A & -1 \leq x(t) \leq 1 \\ p & x(t) < -1 \end{cases} \quad (5.24)$$

Figure 5.4: Lyapunov exponents for different thresholds settings. For $I_0 > 0.8$, $\lambda = -0.0089$ (fixed points). For $I_0 < 0.8$, λ is mostly positive (chaos); however, there exist periodic windows with negative λ , for example, $I_0 = 0.27$ (period-3).

Lyapunov exponents on different thresholds settings are shown in Figure 5.4. For $I_0 > 0.8$, $\lambda = -0.0089$, which implies the system exhibits fixed points [148]. For $I_0 < 0.8$, λ is mostly positive, which means the network dynamics is chaotic [148]; however, there exist periodic windows with negative λ , for example, $I_0 = 0.27$ (period-3). We find that we should have $I_0 < [z - A(i, j; i, j) - I]/z$ in order to produce chaos, i.e., $I_0 < 0.75$ if $z = 4$, though without proof.

We further observe the output of the neuron for fixed I_0 . Figure 5.5 shows neuron output y as a function of time t in the single-neuron C-CNN, for different choices of I_0 (but fixed against time t). For each I_0 , we present the output in 40 iteration steps with randomly generated initial states. Figure 5.5(a) shows a chaotic output when $I_0 = 0.18$. In Figure 5.5(b) with $I_0 = 0.27$, the neuron output begins oscillating between 1, 1, and -1 (period-3 oscillations) - “Period-3 implies chaos” [145]. When $I_0 = 0.8$, the network approaches a fixed point.

We now let I_0 decay exponentially, i.e., $I_0(t + 1) = (1 - \beta)I_0(t)$, where β is

the decaying factor. We start from the initial value $I_0(0) = 1$ and $\beta = 0.0015$. The internal state and output of the neuron are shown in Figure 5.6 (a) and (b). Consistent with the result in Figure 5.4, in the first about 150 iterations, i.e., when $I_0 > 0.8$, there is only one stable equilibrium for the output. Chaos emerges as I_0 goes smaller.

Different applications may require different durations of chaotic dynamics (determined by the damping factor β) or different types of damping scheme (we use exponential damping). These parameters are required to be set properly corresponding to the user requirement.

5.5.3 Transiently chaotic dynamics of the single neuron model

The single neuron model for the TC-CNN obtained from equations (5.6)–(5.8) is:

$$x(t+1) = -z(t)(y(t) - I_0) + p x(t) + A y(t) + B u + I \quad (5.25)$$

$$y(t) = \frac{1}{2}(|x(t) + 1| - |x(t) - 1|) \quad (5.26)$$

$$z(t+1) = (1 - \beta)z(t) \quad (5.27)$$

Let $B u + I = I'$. The one-dimensional mapping function from $x(t)$ to $x(t+1)$ becomes, with $z(t)$ fixed at z_0 :

$$x(t+1) = \begin{cases} -z_0(1-I_0) + p x(t) + A + I' & 1 \leq x(t) \\ -z_0(x(t) - I_0) + (p+A) x(t) + I' & -1 \leq x(t) \leq 1 \\ -z_0(-1-I_0) + p x(t) - A + I' & x(t) \leq -1 \end{cases} \quad (5.28)$$

Figure 5.5: Neuron output y as a function of time t in the single-neuron chaotic cellular neural network, for different choices of I_0 (but fixed against time t). (a) $I_0 = 0.18$, chaotic output; (b) $I_0 = 0.27$, period-3 oscillations between 1, 1, and -1 ; (c) $I_0 = 0.8$, fixed point.

Figure 5.6: The internal state and output of a single-neuron chaotic cellular neural network with an exponential decaying I_0 (a) the internal state of neuron x ; (b) the output of neuron y ; (c) exponential decaying I_0 .

Figure 5.7: The one-dimensional mapping function from $x(t)$ to $x(t + 1)$ for the TC-CNN: (a) zoom out (b) zoom in.

Fig.5.7 presents an example of the one-dimensional mapping function from $x(k)$ to $x(k + 1)$ when $p = 0.7, A = 0.1, I' = 0.08, I_0 = 0.35, z_0 = 5$; It shows that the slop of the mapping function at the intersection is less than -1, which implies that the system is unstable [148].

To simulate the single neuron model, we choose parameters as follows for (5.25), $A = 0.1, B = -1.8, p = 0.7, I = 0.1, I_0 = 0.35, u = 0.1, \beta = 0.001, z(1) = 5, x(1)$ is random number whose element is distributed in the interval $(0, 1)$. The result in the Fig.5.8 shows the time evolutions of $y(t), z(t)$ and the Lyapunov exponent λ of $y(t)$. With exponential damping of $z(t)$, the dynamics of the network transit from chaos to periodic attractors and then fixed point through reversed discontinuity induced bifurcations [149]. It is different from the reversed period-doubling bifurcations presented by Chen and Aihara [2], as Chen and Aihara used a nonlinear output function whereas CNNs have a piece-wise linear output function [150].

Lyapunov exponent [147] is a crucial index to identify deterministic chaos. The definition is shown in (5.22).

Figure 5.8: Dynamics of the single neuron model (a) the output of the neuron; (b) the self-feedback in network.

A positive λ indicates chaos. Figure 5.9 presents the relationship between the Lyapunov exponent and the negative self-feedback parameter z . The Lyapunov exponent is positive ($\lambda > 0$) when the self-feedback parameter $z > 2.3$ (or when $t < 500$). In the meantime, we can see the chaotic behavior clearly in the Figure 5.8(a) in this domain. To study the multi-neuron dynamics, we have also simulated a two-dimensional 3×3 network. This multi-neuron CNN shows identical bifurcation diagrams as in Figure 5.10 with the single neuron model.

5.6 Conclusions

When we add negative self-feedbacks, our discrete chaotic cellular neural networks (C-CNNs) have various complex dynamics (i.e. limit cycles, bifurcation processes

Figure 5.9: Dynamics of the single neuron model: the relationship between the Lyapunov exponent and the self-feedback.

Figure 5.10: Dynamics of a multi-neuron CNN(a two-dimensional 3×3 network).

and chaotic attractors) depending on the magnitude of the self-feedback. The stability condition is proved.

We have shown in [146] that the C-CNN model with a variable neuronal self-feedback is able to produce various dynamics, including chaos, periodic oscillations, and stable fixed points. In this Chapter, we have shown that variable thresholds in the C-CNN model has a great effect on producing chaos. Chaos exist when the threshold is small and there is no chaotic dynamics if the threshold $I_0 > [z - A(i, j; i, j) - I]/z$.

We propose a discrete-time transiently chaotic cellular neural network. We show that the Euler approximation of the continuous-time cellular neural networks may have complex dynamics when we add negative self-feedbacks, depending on the magnitude of the self-feedback. The length of the chaotic dynamics can be determined by the initial value of the self-feedback $z(0)$ and the damping factor β . The network may benefit more from the ergodic nature of chaos if the chaotic dynamics last longer, i.e., a larger $z(0)$ or a smaller β ; However, the network needs more computation time to converge and to search for a solution. More study should be done to analyze the relationship between the performance of TC-CNN model and its chaotic duration. Different applications may require different lengths of chaotic dynamic and different types of damping scheme.

An analog circuit structure and a layout for the realization of DTCNNs were

introduced in [132], the algorithm is:

$$x_{ij}(t+1) = \sum_{C(k,l) \in N_r(i,j)} A(i,j;k,l)y_{kl}(t) + \sum_{C(k,l) \in N_r(i,j)} B(i,j;k,l)u_{kl} + I_{ij} \quad (5.29)$$

Contrasting the new model (5.6) with (5.29), we see the following two differences between the TC-CNN model and DTCNNs: firstly, the new model has a new term $px_{ij}(t)$; secondly, there should be negative self-feedback in the new model, and the value needs to damp exponentially. As the $x_{ij}(t)$ represents the internal state of the neuron, the current $px_{ij}(t)$ can be implemented through a buffer and an amplifier with a gain of p . Furthermore, $\sum a(i,j;k,l)y_{kl}(t)$ is implemented by the current supply in the analog implementation of DTCNNs, and the positive or negative of the value is depend on the direction of the current supply. On the other hand, the damping of the value can be achieved by adjusting the parameters of the switched-capacitor. Therefore, our new model can be easily implemented in hardware.

The new *transiently chaotic cellular neural network* has two distinctive properties: It has complex dynamics and therefore may have encouraging capability in solving combinatorial optimization problems. Furthermore, the local connectivity of the TC-CNN makes it especially suitable for large scale circuit implementation.

Cellular neural networks with chaotic dynamics may help in the development of image processing techniques. Generally, the CNN converges to only binary outputs of -1 and 1 according to network dynamic functions with simple feedback and control templates. Hence only a fully saturated or a black color can be obtained from the CNN [151]. With complex nonlinear dynamics, the TC-CNN has potential

applications in grey images and even color images. In the future, we will further explore the C-CNN and TC-CNN model, with applications in chaotic communications and combinatorial optimization problems.

Chapter 6

Conclusions

6.1 Outcomes of the thesis

This thesis has been concerned with the use of neural networks for solving optimization problems in communications, especially those with practical significance.

The equation governing the dynamics of the neural network and formulation of the QoS (delay) constrained multicast routing problem in terms of the NCNN architecture are discussed. Simulation results show that the NCNN model is efficient on finding the optimal delay constrained routing for large scale communication networks. Transient chaos and noise play an important role in the searching process for optimal routing as they help to avoid the local minimal and continue the search for better solutions. However, similar to other neural network methods for combinatorial optimization problems, in order to converge to valid states and preferably a quality solution, the value of penalty terms in the energy function is very important. Hence tuning of parameters is required, which is a trial-and-error procedure.

We further explored the transiently chaotic neural network with variable thresholds (TCNN-VT) and the noisy chaotic neural network with variable thresholds

(NCNN-VT), whose thresholds are varied with optimization objectives. With rich and complex dynamics, the new models maintains the advantage of their original counterparts. We solve the FAP in satellite communications with the TCNN-VT and the NCNN-VT. To minimize the largest interference, the neuron is designed to have a larger threshold if this neuron presents an assignment with smaller interference. The simulation results on several benchmark instances demonstrate that the NCNN-VT offers further improvements over the TCNN-VT, and they both can find better solutions than existing algorithms.

CNNs may have complex dynamics when we add on negative self-feedbacks depending on the magnitude of the self-feedback. The new *transiently chaotic cellular neural network* we proposed has two distinctive properties: It has complex dynamics and therefore more powerful ability to search for optimal solutions. And furthermore, the local connectivity of the TC-CNN makes it especially suited for large scale circuit implementation.

6.2 Suggestions for future research

Real-time multimedia application groups has an inherent difficulty: heterogeneity, which is caused by the wide range of the communication rate and the differences in computing power of the network. Multicast group members may have significant characteristics, such as bandwidth availability or delay requirements. Hence the source node is required to transmit in a certain way that matches the most constrained receiver. Instead, it would be advantageous to send data to multiple

receivers at heterogeneous rates and try to match the QoS requirements of each individual receiver [83]. Neural networks have the potential to solve such heterogeneity multicast problem. The f-type neuron which is described by equation 3.10 in Chapter 3 may be updated to a new expression to count in the cost or delay of the link.

The NCNN method has the ability to work on directed graph and the cost and delay can be real numbers. Further more, the algorithm is promising in solving other type QoS constrained problems, such like delay jitter, or bandwidth. As described in section 1.1.1, the metric reliability is multiplicative, and the metric bandwidth is concave, the energy function needs modification if one wants to apply the NCNN on those QoS constrained routing problems.

One of reviewer proposed to simplify the energy function for routing problems by using a term like $C_{ij}X_{ij}/(P_{ij} + \epsilon)$ where ϵ represented a small number. This is practicable if we set P_{ij} to 0 when the link from node i to j does not exist, and set P_{ij} to 1 otherwise. We will try to improve the energy function for neural network more in our future research.

Pornavalai et al [152] presented that employing weighted fair queueing (WFQ) service discipline could greatly reduce the complexity of the QoS constrained routing problem from NP-complete to P-solvable problem. In the near future, it may be possible to operate new multimedia applications in a global environment and support new services on a flexible platform without upgrading of the physical infrastructure.

The CNN is claimed as a powerful technique to mimic the function of biolog-

ical neural circuits for real-time image or video processing. Fantacci, Forti and Pancani [123] presented the CNN has the ability to control the location of the stable equilibrium points, which has been shown to be a crucial property in order to suppress spurious responses and optimize the neural switching fabric performance. The CNN switching fabric is able to achieve nearly optimal performances, in terms of throughput maximization, and optimization of a weight function measuring the switching delay and priority of the cells. The capability to account for the priority is a useful feature that can be exploited to meet a variety of QoS requirements in an asynchronous transfer mode (ATM) communication network handling multimedia traffic.

Author's Publications

Journal Paper

1. Lipo Wang, Wen Liu, Haixiang Shi, and Jacek M. Zurada, "Transiently chaotic cellular neural networks," *IEEE Transactions on Circuits and Systems II*, Vol.54 , No.5, pp.440-444, May, 2007.
2. Lipo Wang, Wen Liu, and Haixiang Shi, "Noisy Chaotic Neural Networks With Variable Thresholds for the Frequency Assignment Problem in Satellite Communications," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol.38, No.2, pp.209-217, March 2008.
3. Lipo Wang and Wen Liu, "Delay Constrained Multicast Routing Using Chaotic Neural Networks," *IEEE/ACM Transactions on computers*, 25 July 2008. IEEE Computer Society Digital Library. *IEEE Computer Society*, 29 August 2008 (<http://doi.ieeecomputersociety.org/10.1109/TC.2008.127>)
4. Wen Liu, Lipo Wang, and Haixiang Shi, "Satellite Frequency Assignments Using Transiently Chaotic Neural Networks with Variable Thresholds," Accepted

by *International Journal of Information Technology and Intelligent Computing*, Poland.

5. Wen Liu and Lipo Wang, "Noisy Hopfield Neural Network for the Shortest Path Routing Problem," submitted to *Neural Computing and Applications*, 2007. under second review.

Conference Paper

1. Lipo Wang, Lei Zhou, and Wen Liu, "FPGA segmented channel routing using genetic algorithms," Proc. IEEE Congress on Evolutionary Computation (CEC05, September 2-5, Edinburgh, U.K.), vol.3, pp.2161-2165, 2005.
2. Wen Liu, Haixiang Shi, Lipo Wang, and J. M. Zurada, "Chaotic cellular neural networks with negative self-feedback," in Eighth Intern. Conf. Artificial Intelligence and Soft Computing, vol. 4029 of Lecture Notes in Computer Science, (Zakopane, Poland), pp.66-75, Springer Berlin / Heidelberg, June 25-29 2006
3. Wen Liu, Lipo Wang and Haixiang Shi, "Noisy Chaotic Neural Networks for Delay Constrained Multicast Routing," Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling, (CISched 2007), 1-5 April 2007, Hawaii, USA. pp.267-270.
4. Wen Liu and Lipo Wang, "Solving the Delay Constrained Multicast Routing Problem Using the Transiently Chaotic Neural Network," *Advances in Neural*

- Networks - ISNN 2007, 4th International Symposium on Neural Networks, ISNN 2007, Nanjing, China, June 3-7, 2007, Proceedings, Part II, pp.57-62.
5. Wen Liu and Lipo Wang, "Variable Thresholds in the Chaotic Cellular Neural Network," Proceedings of the International Joint Conference on Neural Networks, (IJCNN 2007), Florida, USA, August 12-17, 2007, pp.711-714.
 6. Wen Liu, Haixiang Shi, and Lipo Wang, "Minimizing interference in satellite communications using chaotic neural networks," the 3rd International Conference on Natural Computation (ICNC'07), Haikou, China, August 24-27, 2007. Vol.2, pp.441-444
 7. Wen Liu, Haixiang Shi, and Lipo Wang, "Transiently Chaotic Neural Network with Variable Thresholds for the Frequency Assignment Problem in Satellite Communications," submitted to the International Conference on Cognitive Neurodynamics (ICCN'07) Shanghai, China, November 17-21, 2007.

Book chapter

1. Lipo Wang, Haixiang Shi, and Wen Liu, "Noisy Chaotic Neural Networks for Combinatorial Optimization," *Challenges for Computational Intelligence*, edited by Wlodzislaw Duch, Jacek Mandziuk, Springer Berlin / Heidelberg, 2007

Bibliography

- [1] H. Tang, K. C. Tan, and Z. Yi, “A columnar competitive model for solving combinatorial optimization problems,” *IEEE Transactions on Neural Networks*, vol. 15, no. 6, pp. 1568–1573, 2004.
- [2] 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.
- [3] L. P. Wang and D. L. Alkon, *Artificial Neural Networks: Oscillations, Chaos, and Sequence Processing*. Los Alamitos, CA: IEEE Comput. Soc. Press, 1993.
- [4] T. Kwok and K. A. Smith, “A noisy self-organizing neural network with bifurcation dynamics for combinatorial optimization,” *IEEE Transactions on Neural Networks*, vol. 15, no. 1, pp. 84–98, Jan 2004.
- [5] K. Bharath-Kumar and J. Jaffe, “Routing to multiple destinations in computer networks,” *IEEE Transactions on Communications*, vol. 31, no. 3, pp. 343–351, 1983.

- [6] 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.
- [7] A. Roy, N. Banerjee, and S. K. Das, "An efficient multi-objective QoS routing algorithm for real-time wireless multicasting," in *Proceedings of IEEE Vehicular Technology Conferenc*, May. 2002, pp. 1160–1164.
- [8] H. Y. Tyan, J. C. Hou, and B. Wang, "Many to many multicast routing with temporal quality of service guarantees," *IEEE Transactions on Computers*, vol. 52, no. 6, pp. 826–832, 2003.
- [9] C. Pornavalai, G. Chakraborty, and N. Shiratori, "A neural network approach to multicast routing in real-time communication networks," in *International Conference on Network Protocols (ICNP-95)*, 1995, pp. 332–339.
- [10] 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.
- [11] J. Chen, S. H. G. Chan, and V. O. K. Li, "Multipath routing for video delivery over bandwidth-limited networks," *IEEE Transactions on Selected Areas in Communications*, vol. 22, no. 10, pp. 1920–1932, Dec 2004.
- [12] X. P. Xiao and L. M. Ni, "Internet QoS: A big picture," *IEEE Network Magazine*, vol. 13, no. 2, pp. 8–18, Mar. 1999.

- [13] D. S. Reeves and H. F. Salama, "A distributed algorithm for delay-constrained unicast routing," *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 239–250, 2000.
- [14] Z. Wang and J. Crowcroft, "QoS routing for supporting multimedia applications," *IEEE Journal of Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, Sep. 1996.
- [15] R. A. Guerin and A. Orda, "QoS routing in networks with inaccurate information: theory and algorithms," *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 350–364, June. 1999.
- [16] D. Ghosh, V. Sarangan, and R. Acharya, "Quality-of-service routing in IP networks," *IEEE Transactions on Multimedia*, vol. 3, no. 2, pp. 200–208, June 2001.
- [17] Y. K. Park and G. Lee, "Applications of neural networks in highspeed communication networks," *IEEE Communications*, vol. 33, pp. 68–74, 1995.
- [18] C. J. Wang and P. N. Weisseler, "The use of artificial neural networks for optimal message routing," *IEEE Network Magazine*, pp. 16–24, 1995.
- [19] H. F. Salama, D. S. Reeves, and Y. Viniotis, "Evaluation of multicast routing algorithms for real-time communication on high-speed networks," *IEEE J. Select. Areas Communication.*, vol. 15, no. 3, pp. 332–345, 1997.

- [20] E. D. Re, R. Fantacci, and L. Ronga, "A dynamic channel allocation technique based on hopfield neural networks," *IEEE Transactions on Vehicular Technology*, vol. 45, no. 1, pp. 26–32, Feb 1996.
- [21] K. Smith and M. Palaniswami, "Static and dynamic channel assignment using neural networks," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 2, pp. 238–249, Feb 1997.
- [22] R. A. Murphey, P. M. Pardalos, and M. G. C. Resende, *Frequency assignment problems*. Kluwer Academic Publishers, 1999, vol. Supplement Volume A.
- [23] R. Montemanni, J. N. Moon, and D. H. Smith, "An improved tabu search algorithm for the fixed-spectrum frequency-assignment problem," *IEEE Transactions on Vehicular Technology*, vol. 52, no. 3, pp. 891–901, May 2003.
- [24] B. Pontano, "Interference into angel-modulated systems carrying multichannel telephony signals," *IEEE Trans. Communications*, vol. 21, 1973.
- [25] M. Jeruchim, "A survey of interference problems and applications to geostationary satellite networks," in *Proceedings IEEE*, 1977, pp. 317–331.
- [26] T. Mizuike and Y. Ito, "Optimization of frequency assignment," *IEEE Trans. Communications*, vol. 37, no. 10, pp. 1031–1041, 1989.
- [27] S. Haykin, *Neural Networks-A comprehensive Foundation (second edition)*. Hamilton, Canada: Prentice Hall International Inc., 1999.

- [28] G. Grassi, "On discrete-time cellular neural networks for associative memories," *IEEE Transactions on Circuits and Systems*, vol. 48, no. 1, pp. 107–111, Jan. 2001.
- [29] L. P. Wang, "Multi-associative neural networks and their applications to learning and retrieving complex spatio-temporal sequences," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 29B, no. 1, pp. 73–82, Feb.1999.
- [30] S. H. Huang and H.-C. zhang, "Artificial neural networks in manufacturing: concepts, applications, and perspectives," *IEEE Trans. on Components, Packaging, and Manufacturing Technology-Part A*, vol. 17, no. 2, pp. 212–228, 1994.
- [31] T. Kohonen, "An introduction to neural computing," *Neural Networks*, vol. 1, no. 1, pp. 3–16, 1988.
- [32] H. E. Rauch and T. Winarske, "Neural networks for routing communication traffic," *IEEE Cont. Syst. Mag.*, vol. 8, no. 2, pp. 26–31, 1988.
- [33] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Academy Science*, vol. 79, pp. 2554–2558, 1982.
- [34] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 7, pp. 115–133, 1943.
- [35] J. J. Hopfield and D. W. Tank, "Neural computation of decisions inoptimiza-
tion problems," *Biological Cybernetics*, vol. 52, pp. 141–152, 1985.

- [36] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
- [37] 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.
- [38] ———, "Solution of the optimization problem using the neural-network model as a globally coupled map," *Physica D*, vol. 75, pp. 179–189, 1994.
- [39] K. Aihara, T. Takabe, and M. Toyoda, "Chaotic neural networks," *Phys. Lett. A*, vol. 144, no. 6,7, pp. 333–340, 1990.
- [40] Z. He, Y. Zhang, C. Wei, and J. Wang, "A multistage self-organizing algorithm combined transiently chaotic neural network for cellular channel assignment," *IEEE Transactions on Vehicular Technology*, vol. 51, no. 6, pp. 1386 – 1396, 2002.
- [41] L. P. Wang and K. Smith, "On chaotic simulated annealing," *IEEE Transactions on Neural Networks*, vol. 9, no. 4, pp. 716–718, July. 1998.
- [42] L. N. Chen and K. Aihara, "Chaos and asymptotical stability in discrete-time neural networks," *Physica D*, vol. 104, pp. 286–325, June. 1997.
- [43] Y. He, "Chaotic simulated annealing with decaying chaotic noise," *IEEE Transactions on Neural Networks*, vol. 13, no. 6, pp. 1526–1531, 2002.

- [44] T. Kwok and K. A. Smith, "Experimental analysis of chaotic neural network models for combinatorial optimization under a unifying framework," *Neural Networks*, vol. 13, no. 7, pp. 731–744, Sep 2000.
- [45] 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 Transactions on System, Man, and Cybernetics-Part B: Cybernetics*, vol. 34, no. 5, pp. 2119–2125, 2004.
- [46] Y. V. Andreyev, Y. L. Belsky, A. S. Dmitriev, and D. A. Kuminov, "Information processing using dynamical chaos," *IEEE Transactions on Neural Networks*, vol. 7, no. 2, pp. 290–299, 1996.
- [47] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Transactions on Circuits and Systems I*, vol. 35, no. 10, pp. 1257–1272, 1988.
- [48] ———, "Cellular neural networks: Applications," *IEEE Transactions on Circuits and Systems I*, vol. 35, no. 10, pp. 1273–1290, 1988.
- [49] H. Chen, M. D. Dai, and X. Y. Wu, "Bifurcation and chaos in discrete-time cellular neural networks," in *Third IEEE international Workshop on Cellular Neural Networks and their Applications*, Dec. 1994, pp. 309–314.
- [50] F. Zou and J. A. Nossek, "Bifurcation and chaos in cellular neural networks," *IEEE Transactions on Circuits and Systems I*, vol. 40, no. 3, pp. 166–173, 1993.

- [51] K. Aihara, "Chaos engineering and its application to parallel distributed processing with chaotic neural networks," *Proceedings of the IEEE*, vol. 90, no. 5, pp. 919–930, 2002.
- [52] M. Delgado-Restituto and A. Rodriguez-Vazquez, "Integrated chaos generators," *Proceedings of the IEEE*, vol. 90, no. 5, pp. 747–767, 2002.
- [53] I. Tokuda, K. Aihara, and T. Nagashima, "Adaptive annealing for chaotic optimization," *Physical Review E*, vol. 58, no. 4, pp. 5157–5160, October 1998.
- [54] S. Sharma, "An exploratory study of chaos in human-machine system dynamics," *IEEE Trans. Systems, Man, and Cybernetics-Part A: Systems and humans*, vol. 36, no. 2, pp. 319–326, march 2006.
- [55] L. O. Chua, C. W. Wu, A. Huang, and G.-Q. Zhong, "A universal circuit for studying and generating chaos—part i: Routes to chaos," *IEEE Transactions on Circuits and Systems I: Fundamental theory and applications*, vol. 40, no. 10, pp. 732–744, Oct. 1993.
- [56] Y. Hayakawa, A. Marumoto, and Y. Sawada, "Effects of the chaotic noise on the performance of a neural network model for optimization problems," *Physical review E*, vol. 51, no. 4, pp. R2693–R2696, 1995.
- [57] M. Bucolo, R. Caponetto, L. Fortuna, M. Frasca, and A. Rizzo, "Does chaos work better than noise?" *Circuits and Systems Magazine, IEEE*, vol. 2, no. 3, pp. 4–19, 2002.

- [58] V. Pavlovic, D. Schonfeld, and G. Friedman, "Stochastic noise process enhancement of hopfield neural networks," *IEEE Transactions on Circuits and Systems II*, vol. 52, no. 4, pp. 213–217, 2005.
- [59] L. P. Wang and F. Y. Tian, "Noisy chaotic neural networks for solving combinatorial optimization problems," in *Proc. International Joint Conference on Neural Networks*, vol. 4, Como, Italy, July 24-27 2000, pp. 37–40.
- [60] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equations of state calculations by fast computing machines," *Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, June 1953.
- [61] B. W. Lee and B. J. Sheu, "Hardware annealing in electronic neural networks," *IEEE Transactions on Circuits and Systems*, vol. 38, pp. 134–137, 1991.
- [62] Y. Akiyama, A. Yamashita, M. Kajiura, Y. Anzai, and H. Aiso, "The gaussian machine: A stochastic neural network for solving assignment problems," *Journal of Neural Network Computing*, vol. 2, pp. 43–51, 1991.
- [63] C. Peterson and J. R. Anderson, "A mean field theory algorithm for neural networks," *Complex Systems*, vol. 1, pp. 995–1019, 1987.
- [64] R. E. Bellman, *Dynamic Programming*. Princeton, New Jersey.: Princeton University Press, 1957.
- [65] L. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton, New Jersey.: Princeton University Press, 1962.

- [66] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematic*, vol. 1, pp. 269–271, 1959.
- [67] C. W. Ahn and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," *IEEE Transactions on Evolutionary computation*, vol. 6, no. 6, pp. 566–579, 2002.
- [68] A. Roy and S. K. Das, "Optimizing qos-based multicast routing in wireless networks: A multi-objective genetic algorithmic approach," in *Networking 2002*, vol. LNCS 2345, Springer-Verlag Berlin Heidelberg, 2002, pp. 28–48.
- [69] G. Lu and Z. Liu, "Multicast routing based on ant-algorithm with delay and delayvariation constraints," in *Circuits and Systems. IEEE APCCAS 2000*, 2000, pp. 243–246.
- [70] M. K. M. Ali and F. Kamoun, "Neural networks for shortest path computation and routing in computer networks," *IEEE Transactions on Neural Networks*, vol. 4, no. 6, pp. 941–954, 1993.
- [71] F. A. Kuipers, T. Korkmaz, M. Krunz, and P. V. Mieghem, "An overview of constraint-based path selection algorithms for QoS routing," *IEEE Commun. Mag.*, vol. 40, pp. 50–55, 2002.
- [72] D. C. Park and S. E. Choi, "A neural network based multi-destination routing algorithm for communication network," in *Proc. of 1998 IEEE Int. Joint Conf. on Neural Networks*, 1998, pp. 1673–1678.

- [73] F. Araujo, B. Ribeiro, and L. Rodrigues, "A neural network for shortest path computation," *IEEE Transactions on Neural Networks*, vol. 12, no. 5, pp. 1067–1073, 2001.
- [74] 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.
- [75] K. Jim, C. L. Giles, and B. Horne, "An analysis of noise in recurrent neural networks: Convergence and generalization," *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1424–1438, Nov 1996.
- [76] S. Das and O. Olurotimi, "Noisy recurrent neural network: The continuous-time case," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 913–936, 1998.
- [77] O. Olurotimi and S. Das, "Noisy recurrent neural network: The discrete-time case," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 937–946, Sep 1998.
- [78] R.-S. Chang and C.-D. Wang, "Improved WWW multimedia transmission performance in HTTP/TCP over ATM networks," *IEEE Transactions on Multimedia*, vol. 1, no. 3, pp. 278–290, September 1999.
- [79] T. Korkmaz and M. Krunz, "Routing multimedia traffic with QoS guarantees," *IEEE Transactions on Multimedia*, vol. 5, no. 3, pp. 429–443, Sep. 2003.

- [80] A. Ganjam and H. Zhang, "Internet multicast video delivery," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 159–170, 2005.
- [81] K. M. F. Elsayed, "A framework for end-to-end deterministic-delay service provisioning in multiservice packet networks," *IEEE Transactions on Multimedia*, vol. 7, no. 3, pp. 563–571, 2005.
- [82] V. Sarangan, D. Ghosh, and R. Acharya, "Capacity-aware state aggregation for interdomain QoS routing," *IEEE Transactions on Multimedia*, vol. 8, no. 4, pp. 792–808, August 2006.
- [83] M. Charikar, J. S. Naor, and B. Schieber, "Resource optimization in QoS multicast routing of real-time multimedia," *IEEE Transactions on Networking*, vol. 12, no. 2, pp. 340–348, April 2004.
- [84] C. P. Low and X. Y. Song, "On finding feasible solutions for the delay constrained group multicast routing problem," *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 581–588, 2002.
- [85] S. Chen and K. Nahrstedt, "Distributed quality-of-service routing in Ad-Hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1–19, Aug 1999.
- [86] D. Chakraborty, G. Chakraborty, and N. Shiratori, "A dynamic multicast routing satisfying multiple QoS constraints," *Int. Journal of Network Management*, vol. 13, no. 5, pp. 321–335, 2003.

- [87] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, "Multicast routing for multimedia communication," *IEEE/ACM Transactions Networking*, vol. 1, no. 3, pp. 286–292, June 1993.
- [88] R. Widyono, "The design and evaluation of routing algorithms for real-time channels," University of California at Berkeley & International Computer Science Institute, Tech. Rep. 94-024, June 1994.
- [89] Q. Sun and H. Langendoerfer, "Efficient multicast routing for delay-sensitive applications," in *Proc. Second Workshop Protocols Multimedia Systems*, 1995, pp. 452–458.
- [90] Q. Zhu, M. Parsa, and J. J. Garcia-Luna-Aceves, "A source-based algorithm for near-optimum delay-constrained multicasting," in *Proc. IEEE Infocom 95*, 1995, pp. 377–385.
- [91] M. Parsa, Q. Zhu, and J. J. Garcia-Luna-Aceves, "An iterative algorithm for delay-constrained minimum-cost multicasting," *IEEE Transactions on Networking*, vol. 6, no. 4, pp. 461–474, 1998.
- [92] J. Y. Yen, "Fiding the K shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.
- [93] P. Manyem, *Routing problems in multicast network*. Raleigh, NC, USA: Ph.D. dissertation, North Carolina State University, 1996, ch. 3, pp. 28–39.

- [94] A. H. Gee and R. W. Prager, "Limitations of neural networks for solving traveling salesman problems," *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 280–282, 1995.
- [95] Z. Ding, H. Leung, and Z. Zhu, "A study of the transiently chaotic neural network for combinatorial optimization," *Mathematical and Computer Modelling*, vol. 36, no. 9, pp. 1007–1020, 2002.
- [96] X. Xu, Z. Tang, and J. Wang, "A method to improve the transiently chaotic neural network," *Neurocomputing*, vol. 67, pp. 456–463, March 2005.
- [97] L. P. Wang and H. Shi, "A gradual noisy chaotic neural network for solving the broadcast scheduling problem in packet radio networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 989 – 1000, 2006.
- [98] 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.
- [99] B. Waxman, "Routing of multipoint connections," *IEEE J. select. Areas Communication*, vol. 6, no. 9, pp. 1617–1622, 1988.
- [100] Z. Wang and B. Shi, "Solution to qos multicast routing problem based on heuristic genetic algorithm," *Journal of Computer*, vol. 24, no. 1, pp. 55–61, 2001 (in Chinese).

- [101] Q. Zhang and Y. W. Leung, "An orthogonal genetic algorithm for multimedia multicast routing," *IEEE Transactions on Evolutionary Computatio*, vol. 3, pp. 53–62, April 1999.
- [102] C. H. Chu, J. H. Gu, X. D. Hou, and Q. J. Gu, "A heuristic ant algorithm for solving qos multicast routing problem," in *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 2, Honolulu HI USA, 2002, pp. 1630–1635.
- [103] D. J. Whalen, "Communications satellites: Making the global village possible," <http://www.hq.nasa.gov/office/pao/History/satcomhistory.html>.
- [104] N. Funabiki and S. Nishikawa, "A gradual neural-network approach for frequency assignment in satellite communication systems," *IEEE Trans. Neural Networks*, vol. 8, no. 6, pp. 1359–1370, 1997.
- [105] S. Salcedo-Sanz and C. B-Calzón, "A hybrid neural-genetic algorithm for the frequency assignment problem in satellite communications," *Applied Intelligence*, vol. 22, no. 3, pp. 207–217, May 2005.
- [106] H. Shi and L. P. Wang, "An adaptive noisy chaotic neural network approach for frequency assignment in satellite communications systems," in *The 12th International Conference on Neural Information Processing, ICONIP'05*, Taipei, October 30 - November 2 2005, pp. 150–153.
- [107] S. Salcedo-Sanz, R. Santiago-Mozos, and C. B-Calzón, "A hybrid hopfield network-simulated annealing approach for frequency assignment in satellite

- communications systems,” *IEEE Trans. Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 34, no. 2, pp. 1108–1116, 2004.
- [108] J. J. Hopfield, “Neurons with graded response have collective computational properties like those of two-state neurons,” *Proc. Nat. Academy Science*, vol. 81, pp. 3088–3092, May. 1984.
- [109] L. P. Wang and J. Ross, “A variable threshold as a model of selective attention, (de)sensitization, and anesthesia in associative neural networks,” *Biological Cybernetics*, vol. 64, pp. 231–241, 1991.
- [110] T. Kurokawa and S. Kozuka, “A proposal of neural networks for the optimum frequency assignment problem,” *Trans. IEICE*, vol. J76-B-II, no. 10, pp. 811–819, Oct. 1993.
- [111] L. Chen and K. Aihara, *Transient chaotic neural networks and chaotic simulated annealing*. Amsterdam: Elsevier Science Publishers, 1994, pp. 347–352.
- [112] H. Shi, “Solving optimization problems in communications networks using computational intelligence,” Ph.D. dissertation, School of EEE, Nanyang Technological University, Singapore, Nov 2006.
- [113] L. Wang, W. Liu, and H. Shi, “Noisy chaotic neural networks with variable thresholds for the frequency assignment problem in satellite communications,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 38, no. 2, pp. 209–217, March 2008.

- [114] G. Manganaro and J. Pineda de Gyvez, "One-dimensional discrete-time CNN with multiplexed template-hardware," *IEEE Transactions on Circuits and Systems I*, vol. 47, no. 5, pp. 764–769, 2000.
- [115] R. Fantacci, M. Forti, and M. Marini, "A cellular neural network for packet selection in a fast packet switching fabric with input buffers," *IEEE Transactions on Communications*, vol. 44, no. 12, pp. 1649–1652, 1996.
- [116] S. H. Bang, B. J. Sheu, and E. Y. Chou, "A hardware annealing method for optimal solutions on cellular neural networks," *IEEE Transactions on Circuits and Systems*, vol. 43, no. 6, pp. 409–421, 1996.
- [117] R. Caponetto, L. Fortuna, L. Occhipinti, and M. G. Xibilia, "SC-CNNs for chaotic signal applications in secure communication systems," *International Journal of Neural Systems*, vol. 13, no. 6, pp. 461–468, 2003.
- [118] N. Takahashi, T. Otake, and M. Tanaka, "The template optimization of discrete time CNN for image compression and reconstruction," in *IEEE International Symposium on Circuits and Systems, ISCAS*, 2002, pp. 237–240.
- [119] R. Bise, N. Takahashi, and T. Nishi, "An improvement of the design method of cellular neural networks based on generalized eigenvalue minimization," *IEEE Transactions on Circuits and Systems I*, vol. 50, no. 12, pp. 1569–1574, 2003.
- [120] S. Wang and M. Wang, "A new detection algorithm (NDA) based on fuzzy cellular neural networks for white blood cell detection," *IEEE Transactions on Information Technology in Biomedicine*, vol. 10, no. 1, pp. 5–10, Jan. 2006.

- [121] G. Grassi, E. D. Sciascio, L. A. Grieco, and P. Vecchio, "New object-oriented segmentation algorithm based on the CNN paradigm," *IEEE Transactions on Circuits and Systems II*, vol. 53, no. 4, pp. 259–263, 2006.
- [122] X. Li, C. Ma, and L. Huang, "Invariance principle and complete stability for cellular neural networks," *IEEE Transactions on Circuits and Systems II*, vol. 53, no. 3, pp. 202–206, 2006.
- [123] R. Fantacci, M. Forti, and L. Pancani, "Cellular neural network approach to a class of communication problems," *IEEE Transactions Circuits and Systems I*, vol. 46, no. 12, pp. 1457–1467, 1999.
- [124] T. Nakaguchi, K. Omiya, and M. Tanaka, "Hysteresis cellular neural networks for solving combinatorial optimization problems," in *Proc. of CNNA 2002*, 2002, pp. 539–546.
- [125] H. T. Lu, Y. B. He, and Z. Y. He, "A chaos-generator: analyses of complex dynamics of a cell equation in delayed cellular neural networks," *IEEE Transactions on Circuits and Systems I*, vol. 45, no. 2, pp. 178–181, 1998.
- [126] P. P. Civalleri and M. Gilli, "On stability of cellular neural networks," *Journal of VLSI Signal Processing*, vol. 23, no. 2-3, pp. 429–435, November 1999.
- [127] F. Zou and J. A. Nossek, "A chaotic attractor with cellular neural networks," *IEEE Transactions on Circuits and Systems I*, vol. 38, no. 7, pp. 811–812, 1991.

- [128] M. Gilli, "Strange attractors in delayed cellular neural networks," *IEEE Transactions on Circuits and Systems I*, vol. 40, no. 11, pp. 849–853, 1993.
- [129] M. Gilli, M. Biey, P. Civalleri, and P. Checco, "Complex dynamics in cellular neural networks," in *Proc. of IEEE International Symposium on Circuits and Systems*, 2001, pp. 45–48.
- [130] I. Petras, P. Checco, M. Gilli, T. Roska, and M. biey, "On the effect of boundary condition on CNN dynamics: Stability and instability; bifurcation processes and chaotic phenomena," in *Proc. of ISCAS 2003*, 2003, pp. 590–592.
- [131] M. Biey, M. Gilli, and P. Checco, "Complex dynamic phenomena in space-invariant cellular neural networks," *IEEE Transactions on Circuits and Systems I*, vol. 49, no. 3, pp. 340–345, 2002.
- [132] H. Harrer and J. A. Nossek, "An analog implementation of discrete-time cellular neural networks," *IEEE Transactions on Neural Networks*, vol. 3, no. 3, pp. 466–475, 1992.
- [133] E. Y. Chou, B. J. Sheu, and R. C. Chang, "VLSI design of optimization and image processing cellular neural networks," *IEEE Transactions Circuits and Systems I*, vol. 44, no. 1, pp. 12–20, 1997.
- [134] G. Linan, S. Espejo, R. Dominguez-Castro, and A. Rodriguez-Vazquez, "ACE4k: an analog I/O 64x64 visual microprocessor chip with 7-bit analog accuracy," *International J. of Circuit Theory and Applications*, vol. 30, pp. 89–116, 2002.

- [135] A. Rodriguez-Vazquez, G. Linan-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galan, F. Jimenez-Garrido, R. Dominguez-Castro, and S. E. Meana, "ACE16k: the third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs," *IEEE Transactions on Circuits and Systems I*, vol. 51, no. 5, pp. 851–863, 2004.
- [136] T. Roska and L. Chua, "The CNN universal machine: an analogic array computer," *IEEE Transactions on Circuits and Systems II*, vol. 40, no. 3, pp. 163–173, 1993.
- [137] T. Roska and A. Rodriguez-Vazquez, "Toward visual microprocessors," *Proceedings of the IEEE*, vol. 90, no. 7, pp. 1244–1257, 2002.
- [138] V. M. Brea, D. L. Vilarino, A. Paasio, and D. Cabello, "Design of the processing core of a mixed-signal CMOS DTCNN chip for pixel-level snakes," *IEEE Transactions on Circuits and Systems I*, vol. 51, no. 5, pp. 997–1013, 2004.
- [139] S. X. de Souza, M. E. Yalcin, J. A. K. Suykens, and J. Vandewalle, "Toward CNN chip-specific robustness," *IEEE Transactions on Circuits and Systems I*, vol. 51, no. 5, pp. 892–902, 2004.
- [140] C.-T. Lin, C.-L. Chang, and J.-F. Chung, "New horizon for CNN: with fuzzy paradigms for multimedia," *IEEE Circuits and Systems Magazine*, vol. second quarter, pp. 20–35, 2005.
- [141] P. Julian and R. D. L. Chua, "Exploiting piecewise linear features: multinested and simplicial cellular neural/nonlinear networks," *Proceedings of the 2003 In-*

ternational Symposium on Circuits and Systems, 2003. ISCAS apos;03., vol. 3, pp. 650–653, May 2003.

- [142] H. Harrer and J. A. Nossek, “Discrete-time cellular neural networks,” *Circuit Theory and Applications*, vol. 20, pp. 453–467, 1992.
- [143] L. P. Wang and J. Ross, “Synchronous neural networks of nonlinear threshold elements with hysteresis,” *Proc. Natl. Acad. Sci. USA*,, vol. 87, pp. 988–992, 1990.
- [144] L. P. Wang, “Discrete-time convergence theory and updating rules for neural networks with energy functions,” *IEEE Transactions on Neural Networks*,, vol. 8, no. 2, pp. 445–447, 1997.
- [145] T. Li and J. Yorke, “Period-3 implies chaos.” *Am. Math. Monthly*, vol. 82, pp. 985–992, 1975.
- [146] W. Liu, H. Shi, L. P. Wang, and J. M. Zurada, “Chaotic cellular neural networks with negative self-feedback,” in *Eighth Intern. Conf. Artificial Intelligence and Soft Computing*, ser. Lecture Notes in Computer Science, vol. 4029. Zakopane, Poland: Springer Berlin / Heidelberg, June 25–29 2006, pp. 66–75.
- [147] L. P. Wang, “Oscillatory and chaotic dynamics in neural networks under varying operating conditions,” *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1382–1388, 1996.
- [148] R. L. Ingraham, *A survey of nonlinear dynamics: chaos theory*. World Scientific, 1992.

- [149] P. Kowalczyk, M. D. Bernardo, A. R. Champneys, S. J. Hogan, M. Homer, P. T. Piiroinen, Y. A. Kuznetsov, and A. Nordmark, “Two-parameter discontinuity-induced bifurcations of limit cycles: Classification and open problems,” *International Journal of Bifurcation and Chaos*, vol. 16, no. 3, pp. 601–629, 2006.
- [150] G. M. Maggio, M. di Bernardo, and M. Kennedy, “Nonsmooth bifurcations in a piecewise-linear model of the colpitts oscillator,” *IEEE Transactions on Circuits and Systems I*, vol. 47, no. 8, pp. 1160–1177, 2000.
- [151] C.-C. Lee and J. P. de Gyvez, “Color image processing in a cellular neural-network environment,” *IEEE Transactions on Neural Networks*, vol. 7, no. 5, pp. 1086–1098, Sep 1996.
- [152] C. Pornavalai, G. Chakraborty, and N. Shiratori, “Routing with multiple QoS requirements for supporting multimedia applications,” *Telecommunication Systems*, vol. 9, no. 3–4, pp. 357–373, 1998.

Appendix I

The following procedure describes the instance generation algorithm for a N-carrier-M-segment FAP.

- Choose the number of carriers N and the number of segments M for the instance.
- Select the values of the range of carrier length and the range of interference between the two systems.
- Generate the set of carrier lengths c_i , ($i = 1, \dots, N$) and the interference matrix $E^{(I)}$ ($M \times M$) using uniformly distributed random values in the range that we defined, respectively.

Appendix II

The following procedure describes the TCNN-VT algorithm on the FAP.

For $RUN = 1, \dots, 1000$:

- STEP 1: Initialization. $T = 1$. Read in the instance information.
Set up the parameters. Compute the threshold value for each neuron. and initialize the neural network status.
- STEP 2: Do neuron updating.

- STEP 3: Validate outputs of neurons. $T++$.
 - IF a valid solution is found, i.e., E_1 and E_2 equal to zero, then store the result.
 - Otherwise IF the number of iterations T reaches T_MAX, then the algorithm does not converge.
 - Otherwise return to STEP 2

IF $Run < 1000$, $Run++$. Go back to STEP 1.

Otherwise go to STEP 4

- STEP 4 :Calculate the statistical result for the performance of the TCNN-VT, including the convergence rate and the best, average, and standard deviation of interference results.