Topology Optimization in Cellular Neural Networks

Varsha Bhambhani and Herbert G. Tanner

UD MEEG Technical Report
Number MEEG TR-2010-0003
July 2010

Department of Mechanical Engineering
University of Delaware
Newark, Delaware
URL: http://www.me.udel.edu

Topology Optimization in Cellular Neural Networks

Varsha Bhambhani and Herbert G. Tanner

University of Delaware, Department of Mechanical Engineering
MEEG TR-2010-0003

July 2010

Abstract

This paper establishes a new constrained combinatorial optimization approach to the design of cellular neural networks with sparse connectivity. This strategy is applicable to cases where maintaining links between neurons incurs a cost, which could possibly vary between these links. The cellular neural network's interconnection topology is diluted without significantly degrading its performance, the network quantified by the average recall probability for the desired patterns engraved into its associative memory. The dilution process selectively removes the links that contribute the least to a metric related to the size of system's desired memory pattern attraction regions. The metric used here is the magnitude of the network's nodes' stability parameters, which have been proposed as a measure for the quality of memorization. Further, the efficiency of the method is justified by comparing it with an alternative dilution approach based on probability theory and randomized algorithms. We demonstrate by means of an example that this method of network dilution based on combinatorial optimization produces cheaper associative memories that in general trade off performance for cost, and in many cases the performance of the diluted network is on par with the original system. Also the randomized algorithm based method results in same network performance in terms of network recall probability.

1 Introduction

Research on the design and synthesis of cellular neural networks (CNN) has gained increasing attention in recent years because of their topology being relatively sparse, which makes them a promising choice for very-large-scale integration implementations. A cellular neural network is a nonlinear dynamical system, implementing an associative memory, where neuron interactions are limited to the neighboring units only [1]. Several synthesis procedures for designing symmetric/non-symmetric neural network have been proposed [2, 3, 4], and one of the main directions of current research on the topic is the stability behavior of these systems in the presence of time delays (see for example [5, 6]), which is however beyond the scope of this paper.

Strengths and weaknesses of network design techniques such as *outer product method*, *projection learning rules*, *eigenstructure method* and design methods for *networks with non-symmetric matrices* have been pointed out in [2]. The analysis results shown in [3] allow one to locate in a systematic manner all equilibrium points of the neural network and to determine the stability properties of the equilibrium points. The above mentioned methods have been generalized in [7] to develop a design procedure for neural networks with sparse connectivity. Their results guarantee that the synthesized neural networks have predetermined sparse interconnection structures and store an almost arbitrary set of desired memory patterns as reachable memory vectors. The authors show that a sufficient condition for the existence of such a sparse neural network design is self feedback for every neuron in the network. Critical network issues like robustness and invariance to perturbation have been addressed in [8] where the authors provide upper bounds for the perturbations of parameters under which desired memories stored in a neural network are preserved. This type of information is of great practical interest during the implementation process of such networks. In [4] the problem of realizing associative memories by designing a CNN that can store given binary vectors with improved performance is formulated as a constrained optimization problem. It is shown that this optimization problem cannot be solved using simple linear programming methods, but it can be reduced to a generalized eigenvalue problem (GEVP), which can be efficiently solved by interior point methods [9]. Networks designed using this method exhibit less spurious patterns and higher recall probabilities [4].

The quality of memorization in associative memories is typically measured by the *average recall probability* [8, 4], which is an approximation of the probability that the network will recall correctly one of the patterns stored in its memory, if presented with a version of this pattern contaminated by noise. This number is defined as the ratio of number of recovered memory patterns (perturbed initial condition vectors which result in same output as the stored memory vector) to the total number of perturbed initial condition vectors. Although this quantification method measures the network's performance accurately and unambiguously, it can only be applied after the network has been designed and tested on a significant number of test inputs. In other words, it does not allow the designer to *predict* the network's performance without experimentation. The concept of stability parameters appears in some earlier work on associative memory networks [10, 11, 12], as a measure of quality of memorization. Specifically, it has been demonstrated [13, 14] that the sign as well as the magnitude of these numbers are related to the size of the attraction regions of the desired memory patters. Although the applicability of this concept as a universal measure of memorization quality, (specifically, the use of their size as a direct measure of the absolute sizes of the attraction regions) has long been debated [15], these parameters are generally accepted as a reasonably good metric for the network's performance [16].

Reported approaches on the design of CNNs (without time delays) start with, and build on some given network topology. The choice of this starting point is justified from the fact that typically the physical platform on which the network is implemented is fixed. In this paper, we raise the question of how could the design process be different if the network topology was also part of the design. In particular, it could be the case that implementing and maintaining certain neuron connections might be more expensive than others, and then one may be forced to strike a balance between connectivity and performance. Simple stability analysis of neural networks implementing associative memories suggests similar links between the network of connections and the stability of the dynamical system as in consensus networks. In the case of neural networks, however, the topologies used are typically standard: either complete graphs (in Hopfield networks) or grid-like structures (in CNNs). Yet, available design methodologies do not place specific conditions on the network's structure. In addition, the cost of communication (delayed or otherwise) between neurons is commonly ignored. But if neurons communicate at a nontrivial cost, there is benefit in designing networks that perform just as well, but are less expensive.

This paper presents an approach to the optimization of the network topology of CNNs in which communication links between neurons may incur variable cost. We selectively "trim" network links in an effort to trade network performance for smaller communication cost. Based on existing design tools, we perform combinatorial optimization on a portion of a CNN that includes the *most expensive* interconnection links. This produces a sparser CNN, the performance of which can be comparable to the original network. Also, an alternative approach suggested in [17] is used to decide the bound of independent identically distributed (i.i.d) network topologies needed to find the near maxima of the stability parameter of the original network. Finally the performance of the topology which results in maximum stability parameter is compared to the network topology obtained by combinatorial optimization approach. This performance is quantified in terms of the network recall probability, and in the proposed optimization algorithm approach is captured by the neural network's stability parameters.

The paper is organized as follows: Section 2 introduces some preliminary material needed for the statement of the problem and the presentation of the methods. The section reviews the two-dimensional continuous time zero-input CNN, the generalized eigenvalue problem associated with the selection of the network's weights and bias, and the determination of the stability parameters. Section 3 focusses on design of an interconnection topology sparser than that of a lattice, without significantly degrading the performance of the network. Further, Section 4 presents an alternative approach based on probability and random algorithm to determine near maxima of stability parameter and compares the results in terms of recall probability of the network obtained by the method suggested in Section 3. We use as an illustrating example, a 24-node CNN to compare this sparser topology to the original design, and the numerical results are presented and discussed in Section 5. Finally Section 6 concludes the paper by summarizing our findings and outlining the future research work.

2 Preliminaries

A two-dimensional continuous time zero-input CNN consisting of $n = MN$ cells in a $M \times N$ array, as first introduced by Chua and Yang [1], can be described by

$$\dot{x}_{ij} = -x_{ij} + \sum_{(k,l)\in N_r(i,j)} W_{ij,kl} y_{kl} + d_{ij}, \tag{1}$$
$$y_{ij} = \mathrm{sat}(x_{ij}),$$

where $1 \leq i \leq M$, $1 \leq j \leq N$, and

$$\mathrm{sat}(x_{ij}) \triangleq \frac{1}{2}\left(\left|x_{ij}+1\right| - \left|x_{ij}-1\right|\right). \tag{2}$$

Here x_{ij} and y_{ij} are the state and output of the (i,j)th cell respectively, and $N_r(i,j)$ is an r-neighborhood of the (i,j)th cell defined as:

$$N_r(i,j) \triangleq \{(k,l) : \max\{|k-i|, |l-j|\} \leq r\} \tag{3}$$
$$1 \leq i \leq M, 1 \leq j \leq N.$$

The above system can be expressed in a matrix-vector form as

$$\dot{x} = -x + T\ \mathrm{sat}(x) + b, \tag{4a}$$
$$y = \mathrm{sat}(x) \tag{4b}$$

where $x = [x_{11}, x_{12}, \ldots, x_{21}, \ldots, x_{MN}]^T \in \mathbb{R}^n$ is the stack vector of all neuron states , $y = [y_{11}, y_{12}, \ldots, y_{21}, \ldots, y_{MN}]^T \in H^n$ is the output vector (H^n is the n-dimensional hypercube $[-1,+1]^n$), $T = [T_{ij}] \in \mathbb{R}^{n\times n}$ is the network connection weight matrix, $b \in \mathbb{R}^n$ is the network's bias vector and for vector arguments, the saturation function is defined elementwise:

$$\mathrm{sat}(x) \triangleq [\mathrm{sat}(x_1), \ldots, \mathrm{sat}(x_n)]^T.$$

Let B^n denote the set of bipolar vectors in H^n, namely those whose elements are either $+1$ or -1. For $i = 1, \ldots, n$, the initial condition vectors of (4) should always satisfy $|x_i(0)| \leq 1$. The interconnection topology of the network can be described by an adjacency matrix S, and thus a weight T_{ij} is non-zero only if $S_{ij} = 1$. Vector $\alpha \in H^n$ is then a *memory vector* for the network (4) if there exists an asymptotically stable equilibrium point $\beta \in \mathbb{R}^n$ of (4) such that $\alpha = \mathrm{sat}(\beta)$ [1].

The synthesis problem for a CNN can be stated as follows:

Problem 1 (Synthesis)**.** *Given a* CNN (4)*, implemented on network expressed by an adjacency matrix S, along with the set of desired bipolar memory vectors $\alpha^1, \ldots \alpha^m \in B^n$, determine the network connection weights T_{ij} and bias parameters b_i so that the obtained neural network can store all desired memory patterns as reachable memory vectors.*

The adjacency matrix S which determines which T_{ij} are zero and which are not, typically encodes a lattice structure and can be selected arbitrarily as long as certain stability criteria of the CNN (4), are satisfied [1, 3, 4]:

1. If $\alpha \in B^n$ and $\beta = T\alpha + b$ are such that

$$\alpha_i \beta_i = \alpha_i \Big(\sum_{j=1}^{n} T_{ij}\alpha_j + b_i\Big) > 1, \quad \forall i = 1, \ldots, n, \tag{5}$$

   then (α, β) is a pair of a memory vector and an asymptotically stable equilibrium point of system (4). Also if $\alpha \in B^n$ and $\beta = T\alpha + b$ are such that for any $i = 1, \ldots, n$,

$$\alpha_i \beta_i = \alpha_i \Big(\sum_{j=1}^{n} T_{ij}\alpha_j + b_i\Big) < 1, \tag{6}$$

   then $\alpha \in B^n$ cannot be a memory vector.

2. System (4) is globally stable if T is symmetric.

In addition to aforementioned stability criteria, specific stability and robustness properties for these networks are established in terms of the elements of the network's weight matrix T and bias vector b [1, 3, 4]:

- Let $\alpha \in B^n$ be a memory vector of system (4) and let $k \geq 1$ be an integer. If $\widetilde{T} = T - I_n$ and b satisfy

$$\alpha_i \left(\sum_{j=1}^{n} (\widetilde{T}_{ij}\alpha_j + b_i) \right) > 2(k-1) \max_{1 \leq j \leq n} \left| \widetilde{T}_{ij} \right|, \quad i = 1, \ldots, n, \tag{7}$$

  then any binary vector $\alpha^* \in B^n$ such that $1 \leq h(\alpha^*, \alpha) \leq k$ ($h(\alpha^*, \alpha) \triangleq \sum_i |\alpha_i^* - \alpha_i|$ denotes the Hamming distance) has the following properties:

  1. α^* is not a memory vector (asymptotically stable equilibrium point for (4)).
  2. if $x(0) = \alpha^*$ and $\alpha_i^* \neq \alpha_i$, then $x_i(t)$ converges to α_i.

- Consider the perturbed system described by

$$\dot{x} = -(I_n + \Delta A)x + (T + \Delta T)\mathrm{sat}(x) + (b + \Delta b), \tag{8a}$$
$$y = \mathrm{sat}(x) \tag{8b}$$

  where $\Delta A = \mathrm{diag}[\Delta a_1, \ldots, \Delta a_n]$ with $1 + \Delta a_i > 0$, $i \in \{1, \ldots, n\}$, $\Delta T \in \mathbb{R}^{n \times n}$ and $\Delta b \in \mathbb{R}^n$. Let $\alpha \in B^n$ be the memory vector of system (4); if

$$\alpha_i \left(\sum_{j=1}^{n} \widetilde{T}_{ij}\alpha_j + b_i \right) > \xi > 0, \quad i = 1, \ldots, n,$$

  then α is also the memory vector of the perturbed system (8) provided that $\|\Delta A\|_\infty + \|\Delta T\|_\infty + \|\Delta b\|_\infty < \xi$.

Satisfying (7) with large k increases both the attractivity and the robustness of the stored memory vector $\alpha \in B^n$ and decrease the probability of existence of spurious patterns in vertices near α [4].

Problem 1 admits several solutions, among which a computationally efficient one proposed in [4], is formulated as a GEVP:

$$\min(-\delta), \quad \text{s.t.} \tag{9a}$$
$$(-\delta)\mathrm{diag}[2q_1, \ldots, 2q_n] - \mathrm{diag}[-p_1, \ldots, -p_n] > 0 \tag{9b}$$
$$\alpha_i^{(k)} \left(\sum_{j=1}^{n} \widetilde{T}_{ij}\alpha_j^{(k)} + b_i \right) - p_i > 0, \quad i = 1, \ldots, n,\ k = 1, \ldots, m \tag{9c}$$
$$q_i - \widetilde{T}_{ij} > 0, \quad i = 1, \ldots, n, \quad j = 1, \ldots, n \tag{9d}$$
$$\widetilde{T}_{ij} + q_i > 0, \quad i = 1, \ldots, n, \quad j = 1, \ldots, n \tag{9e}$$
$$\widetilde{T}_{ii} = 0, \quad i = 1, \ldots, n, \tag{9f}$$
$$\widetilde{T}_{ij} = \widetilde{T}_{ij}^T = \widetilde{T}_{ij}|_S \tag{9g}$$
$$L < q_i < U, \quad i = 1, \ldots, n, \tag{9h}$$

where p_i and q_i for $i = 1, \ldots, n$ are additional slack variables used to cast the design problem as a linear matrix inequality (LMI) [18], and L and U are the lower and upper bounds for the design variables in the GEVP.

In this paper we quantify the quality of memorization in the network in terms of the magnitude of the stability parameters $K_{i\mu}$ defined as:

$$K_{i\mu} = \alpha_i^{(\mu)} h_i = \frac{\alpha_i^{(\mu)} \sum_j c_{ij}\alpha_j^{(\mu)}}{||c_i||_2}, \tag{10}$$

where

$$c_{ij} = T_{ij}$$
$$\|c_i\| = \sqrt{\sum_j c_{ij}^2},$$

and superscipt $(\cdot)^{(\mu)}$ indexes the set of desired memory vectors. The stability parameters $K_{i\mu}$, (one for each pair of neuron node and memory vector) are numbers which have been proposed as a measure of quality of memorization and it has been hypothesized that they are linked to the size of the attraction regions of the neural network [16].

The problem addressed in this paper is the following variant of Problem 1:

Problem 2 (Topology optimization)**.** *Given a* CNN (4)*, implemented on network expressed by a* weighted *graph with adjacency matrix $\hat{S}$, along with the set of desired bipolar memory vectors $\alpha^1, \dots \alpha^m \in B^n$, determine the connection weights T_{ij} and bias parameters b_i of a subnetwork of that of S, so that this subnetwork stores all desired memory patterns as reachable memory vectors, and recalls them (almost) as well, in terms of a performance metric, as the complete network.*

In this paper, the performance metric is formulated using the values of the neural network's stability parameter values. The design process includes an optimization stage, in which the high cost edges that contribute the least to the engraving of the desired patterns on the network's memory, are selectively trimmed.

3 Dilution of Network Connectivity

A CNN can be viewed as a collection of sub-networks that are linked to each other through communication links. We assume that communication incurs a cost, and in a particular scenario of interest, information flow across sub-networks is more expensive compared to communication within each subnetwork. In this case, one may be interested in minimizing communication cost, while maintaining the functionality and performance of the whole network above a certain threshold.

In this paper, reducing communication cost is achieved by trimming those expensive neuron links which do not contribute significantly to the ability of the network to retrieve its memorized information. The communication costs are captured by the weights of the weighted adjacency matrix $\widehat{S}$. Given the (unweighted) adjacency matrix S of the CNN, along with the set of desired bipolar memory vectors $\alpha^1, \dots \alpha^m \in B^n$, the design process begins by determining the network parameters T_{ij} and b_i through the solution of the GEVP (9). The resulting network maximizes the recall probability of the patterns it has been designed for, without considering the cost of using the different network links. The next step is to dilute the connectivity of S, by judiciously removing some of the high cost links identified in $\widehat{S}$, in a way so that the quality of memorization is not severely affected. Balancing performance against communication cost is achieved through (combinatorial) optimization over the network links, subject to the stability constraints stated in Section 2.

The optimization process is iterative. For a given (intermediate) topology S, the algorithm determines the neural network weights T_{ij} for $i, j = 1 \dots, n$ and biases b, and based on the selected patterns $\alpha_i^{(\mu)}$ for $\mu = 1, \dots, m$ to be memorized, an $n \times m$ stability parameter matrix is formed by repeated application of (10):

$$K = [K_{i\mu}]_{i=1,\dots,n;\mu=1,\dots,m}.$$

For the prescribed patterns $\alpha^1 \dots \alpha^m$ to be stored effectively in the network's memory, all stability parameters $K_{i\mu}$ must be nonnegative and as large as possible. For the network topology encoded in S, the value of the following objective function is evaluated:

$$\bar{K} = \sum_{i=1}^{n} \sum_{\mu=1}^{m} K_{i\mu}. \tag{11}$$

Thus $\bar{K}$, being the sum of all nodes' stability parameters for all chosen memory vectors, quantifies the collective ability of the network to recall all desired memories.

Remark 1. *Several performance metrics have been explored as alternatives to* (11)*, based on different norms of the stability parameter matrix K, such as the minimum row (or column) sums, the (absolute) minimum element of K,*

Algorithm 1 Topology optimization

```
Require: Matrices $\alpha$, $S$, $\widehat{S}$, constants $\nu$, $\kappa$.
Ensure: Matrix $A$ of $\bar{K}$ values for each edge that may be removed.
 1: $R \leftarrow \frac{1}{2}\left(\operatorname{sign}(\widehat{S} - \nu S) + S\right)$.
 2: $n \leftarrow$ rowlength$(\alpha)$
 3: $m \leftarrow$ columnlength$(\alpha)$
 4: $A \leftarrow [0]_{n\times m}$
 5: $C \leftarrow [0]_{n\times m}$
 6: while $\max(R) \neq 0$ do
 7:   For each $(i,j)$ such that $R(i,j) \neq 0$, do
 8:   $S' \leftarrow \{S : S(i,j) \leftarrow 0, S(j,i) \leftarrow 0\}$
 9:   Compute $T$ and $b$, given $S'$ and $\alpha^{(\mu)}$
10:   $\bar{K} \leftarrow \sum_{i=1}^{n}\sum_{\mu=1}^{m} K_{i\mu}$
11:   for $\mu = 1$ to $m$ do
12:     for $t = 1$ to $n$ do
13:       $C(t,\mu) \leftarrow \alpha_t^{(\mu)}\left(\sum_{j=1}^{n} T_{tj}\alpha_j^{(\mu)} + b_t\right)$
14:     end for
15:   end for
16:   if $\bar{K} > \kappa \wedge \min C > 1$ then
17:     $A(i,j) \leftarrow \bar{K}$ ; $A(j,i) \leftarrow \bar{K}$
18:   end if
19:   $R(i,j) \leftarrow 0$; $R(j,i) \leftarrow 0$
20: end while
```

etc. *When the average recall probability of each design was evaluated, it was determined that the sum of all stability parameters captured more accurately the ability of the network to recall memory patters.*

On the other hand, a naive implementation of a "branch-and-bound" approach, where links are divided into "promising" and "not-promising" for deletion groups according to their associated $\bar{K}$ value, and only the "promising" possibilities are explored in the subsequent steps, will generally fail. This is because due to the combinatorial nature of the problem, an edge whose sole deletion has an adverse effect on the stability parameters may even improve the value of $\bar{K}$ when combined with additional edge removals.

For a given cost threshold ν, and performance threshold κ, the high-cost edges that are candidates for deletion are identified in the residual adjacency matrix

$$R = \frac{1}{2}\left(\operatorname{sign}(\hat{S} - \nu S) + S\right),$$

where the sign function is evaluated element-wise on the matrix argument. For every nonzero (i,j) element in R, we remove the associated (i,j) edge from S. If (7) is satisfied then we store the resulting value of $\bar{K}$. The high-cost edge associated with the highest stored $\bar{K}$ value is marked for deletion, and the step is repeated for the topology S', $\hat{S}'$ in which neurons i and j are not linked.

The process for evaluating the edges which are candidate for deletion is outlined using pseudo-code in Algorithm 1.

Upon completion, Algorithm 1 provides the (nonzero) performance indices of the network produced after each potential expensive edge which may be trimmed. The (i,j) edge with the highest $\bar{K}$ value in A is removed from the network and the process is repeated until the while loop of Algorithm 1 is no longer executed (R is a matrix of zeros). Then either all edges with cost above the threshold ν are removed, or their removal violates the stability condition 6, or results in an unacceptable performance metric. Although, in general, performance deteriorates as more neuron links are removed, it is possible (and demonstrated in Fig. 3) that one may actually increase the objective function using fewer network connections.

4 A Randomized Algorithm: An Alternative Approach

This section introduces an alternative approach to find the near maxima of the objective function $\bar{K}$ described in section 3. Given the unweighted adjacency matrix S, the weighted adjacency matrix $\hat{S}$ and the residual adjacency matrix R, then for every non-zero (i, j) element in R, let the probability of connection of the (i, j) edge be $\boldsymbol{p}$. For determining the near maxima of the stability parameter $\bar{K}$ of the given network topology S, generate ξ number of random independent identically distributed (i.i.d) networks $\ddot{S}_1$, $\ddot{S}_2 \ldots \ddot{S}_\xi \in \ddot{S}$ distributed according to probability measures $P_{\ddot{S}}$. The integer ξ is computed such that [17]:

$$\xi \geq \frac{lg(1/\acute{\delta})}{lg[1/(1-\acute{\alpha})]}, \tag{12}$$

Where, $\acute{\alpha}, \acute{\delta} \in (0, 1)$ are the specified level and confidence interval respectively.

To generate the random network $\ddot{S}_i$ for $i = 1$ to ξ, generate a $n \times m$ matrix using "*randsrc*" command in matlab and define its elements such that the probability of occurrence of 1 is $\boldsymbol{p}$ and that of 0 is ***1-p***. Replace all non high cost edges ($R_{i,j} = 0$) by the corresponding edge values defined in the given network topology S. Thus the resulting network given by $\ddot{S}_i$ has the same connections except the high cost edges which are randomly generated. Check for the stability condition. We need ξ such random networks to correctly predict the near maxima of the stability function $\bar{K}$ [17]. For each network $\ddot{S}_i$, we determine the maximum stability parameter using (11). Let this be represented by $\ddot{K}_i$. Then near maxima of the stability parameter is given by:

$$\ddot{\bar{K}} = max_{1 \leq i \leq \xi} \ddot{K}_i, \tag{13}$$

Here, the respective network $\ddot{S}_i$ having stability parameter $\ddot{\bar{K}}$ is supposed to outperform its counterparts in terms of stability. The process of random generation of ξ networks and determination of near maxima of $\bar{K}$ is summarized using Algorithm 2.

5 A Design Example and Simulations

As an example, and proof of efficiency of suggested method, we use a CNN that is made up of $n = 24$ cells interconnected locally and the topology represented by the index matrix S is given by (14) as:

$$S = \begin{vmatrix}
0&1&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&1&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0\\
1&1&1&0&0&0&1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0\\
0&1&1&1&0&0&0&1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0\\
0&0&1&1&1&0&0&0&1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0\\
0&0&0&1&1&1&0&0&0&1&1&1&0&0&0&1&1&1&0&0&0&0&0&0\\
0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&0&0\\
0&0&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0&1&1&0&0&0&0\\
0&0&0&0&0&0&1&1&1&0&0&0&1&0&1&0&0&0&1&1&1&0&0&0\\
0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1&0&0&0&1&1&1&0&0\\
0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1&0&0&0&1&1&1&0\\
0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1&0&0&0&1&1&1\\
0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&1&0
\end{vmatrix}, \tag{14}$$

The schematic of the network topology is shown in Fig. 1. As can be seen, the network is divided into three small sub-networks which are linked by costly edges marked by solid lines. These costs are represented in form of a

Algorithm 2 Randomized Algorithm

```
Require: Matrices $\alpha$, $S$, $\widehat{S}$, $p$, $\acute{\delta}$, $\acute{\alpha}$, $\nu$, $\kappa = 0$.
Ensure: Near maxima of $\bar{K}$ and corresponding topology
 1: $R \leftarrow \frac{1}{2}\Big(\mathrm{sign}(\widehat{S} - \nu S) + S\Big)$.
 2: $\xi \leftarrow \frac{lg(1/\acute{\delta})}{lg[1/(1-\acute{\alpha})]}$
 3: $Row \leftarrow$ rowlength($S$)
 4: $Column \leftarrow$ columnlength($S$)
 5: $n \leftarrow$ rowlength($\alpha$)
 6: $m \leftarrow$ columnlength($\alpha$)
 7: $C \leftarrow [0]_{n\times m}$
 8: while $limit < \xi + 1$ do
 9:   $RandMat \leftarrow randsrc(n, m, [0, 1; (1-p), p])$
10:   for $countI = 1$ to $Row$ do
11:     for $countJ = 1$ to $Column$ do
12:       if $R(countI, countJ) = 1$ then
13:         $\ddot{S}(countI, countJ) \leftarrow RandMat(countI, countJ)$
14:       else
15:         $\ddot{S}(countI, countJ) \leftarrow S(countI, countJ)$
16:       end if
17:     end for
18:   end for
19:   Compute $T$ and $b$, given $\ddot{S}$ and $\alpha^{(\mu)}$
20:   $\bar{K} \leftarrow \sum_{i=1}^{n}\sum_{\mu=1}^{m} K_{i\mu}$
21:   for $\mu = 1$ to $m$ do
22:     for $t = 1$ to $n$ do
23:       $C(t,\mu) \leftarrow \alpha_t^{(\mu)}\big(\sum_{j=1}^{n} T_{tj}\alpha_j^{(\mu)} + b_t\big)$
24:     end for
25:   end for
26:   if $\min C > 1$ then
27:     $limit = limit + 1$
28:   end if
29:   if $\kappa \leq \bar{K}$ then
30:     $\kappa \leftarrow \bar{K}$; $Topology \leftarrow \ddot{S}$.
31:   end if
32: end while
```

weighted adjacency matrix $\widehat{S}$ in eq. (15) such that the costly edges have a higher weight.

$$\widehat{S} = \begin{vmatrix}
0&1&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&2&0&0&0&1&1&2&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&2&0&1&0&0&0&2&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&1&0&0&0&0&2&2&0&0&0&0&0&0&0&0&0&0\\
1&1&1&0&0&0&1&0&1&0&0&0&2&2&2&0&0&0&0&0&0&0&0&0\\
0&1&1&2&0&0&0&1&0&1&0&0&0&2&2&2&0&0&0&0&0&0&0&0\\
0&0&2&1&1&0&0&0&2&0&1&0&0&0&2&1&1&0&0&0&0&0&0&0\\
0&0&0&1&1&1&0&0&0&1&1&1&0&0&0&1&1&1&0&0&0&0&0&0\\
0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&0&0\\
0&0&0&0&0&0&2&2&0&0&0&0&0&1&0&0&0&0&1&1&0&0&0&0\\
0&0&0&0&0&0&2&2&2&0&0&0&1&0&1&0&0&0&1&1&1&0&0&0\\
0&0&0&0&0&0&0&2&2&2&0&0&0&1&0&1&0&0&0&1&1&2&0&0\\
0&0&0&0&0&0&0&0&2&1&1&0&0&0&2&0&1&0&0&0&2&1&1&0\\
0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1&0&0&0&1&1&1\\
0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&2&0&0&0&1&0&2&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&2&1&1&0&0&0&2&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&1&0
\end{vmatrix}, \qquad (15)$$

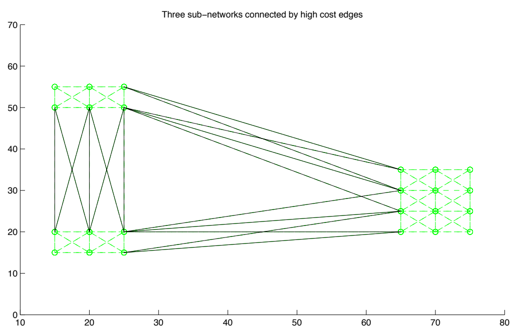


Figure 1: A CNN interconnection structure where high cost edges are stretched and highlighted

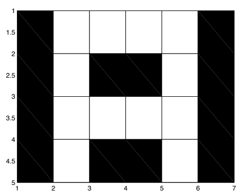
The binary patterns chosen as memory vectors are shown in Fig. 2. Each memory vector corresponds to a 6×4 array with black and white boxes represented numerically by -1 and 1 respectively, and is read left to right from top left corner to produce a single column, stack vector of binary values. The network parameters namely the bias vector b and the weight matrix $T = \widetilde{T} + I_n$, are obtained by solving the GEVP with the bounds in system (9) set to $L = 1$ and $U = 10$.

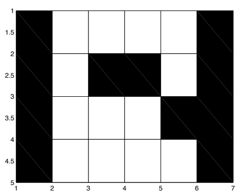
To reduce the communication cost, connection edges from the given set of costly edges marked by solid lines in Fig. 1 are considered for possible removal. These edges can be numerically identified by setting a cost threshold ν and evaluating the residual matrix $R = \frac{1}{2}\left(\operatorname{sign}(\widehat{S} - \nu S) + S\right)$. Table 1 lists which costly edge when trimmed results in what $\bar{K}$ value. For example, deletion of nine costly edges in the sequence shown in table will result in maximum stability parameter sum of 108.3511 in the example considered. Also Fig. 3 shows the variation in cumulative stability parameter $\bar{K}$ with the number of costly edges deleted. As shown in the figure, the dashed line is the original network cumulative stability parameter value and the solid line represents the evaluation of the cumulative stability parameter for the network being optimized, during the optimization process.

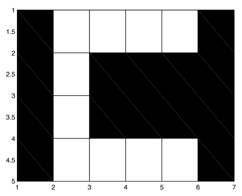
The performance of proposed design method is quantified in terms of average recall probability of the neural net-

Table 1: Cumulative stability parameter $\bar{K}$ as a result of successive deletion of high cost edges.

Edge pair	Node A	Node B	$\bar{K}$
1	8	13	108.3267
2	10	9	108.2702
3	9	14	108.2706
4	8	14	108.2682
5	9	15	108.2867
6	7	13	108.2611
7	3	10	108.2774
8	3	4	108.2660
9	7	14	108.2672
10	10	15	108.3511
11	4	9	108.2609
12	9	16	108.2572
13	8	15	108.2511
14	16	21	107.8300
15	22	15	107.2706
16	21	22	103.7681
17	15	16	100.7607


(a) Pattern 1


(b) Pattern 2


(c) Pattern 3

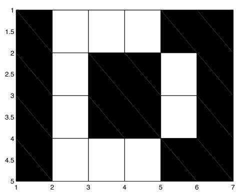

(d) Pattern 4

Figure 2: Four patterns to be stored as memory vectors

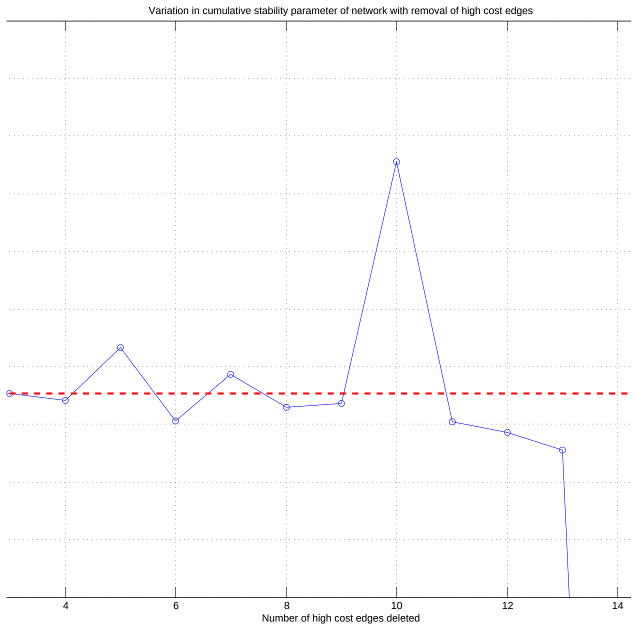


Figure 3: Effect on arithmetic sum of stability parameter $\bar{K}$ with trimming of high cost edges

work. We estimate the recall probabilities by generating a set of 50 different arrays of random noise and contaminating to each memory vector with these arrays to produce 50 different perturbed input memory vectors for each given memory pattern α^{μ}. Also for $i = 1, \ldots, n$, the perturbed input memory vectors always satisfy $|x_i(0)| \leq 1$. For each initial condition vector α^{μ}, the corresponding set of perturbed condition vectors are fed to the network and those perturbed condition vectors which result in same output as the initial condition vector are recorded. A comparison is made be-

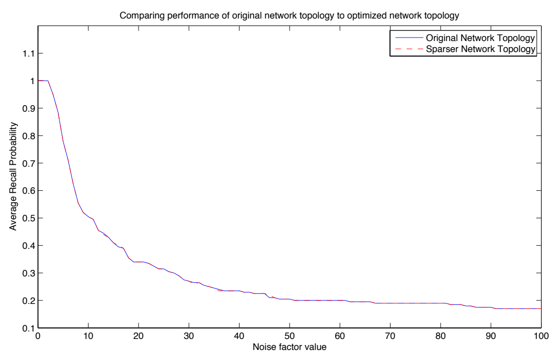


Figure 4: Network performance vs. selective deletion of edges for the original and the optimized network. The curves are initially indistinguishable.

tween the original network model and the network with the topology determined by the optimization algorithm (shown in Fig. 4). It can be seen that the original network topology has same recall probability graph as the sparser topology derived with selective deletion of costly edges. However with a relatively large number of deleted edges, there is a noticeable decrease in network performance. The objective of optimization is thus to balance communication cost versus average recall probability.

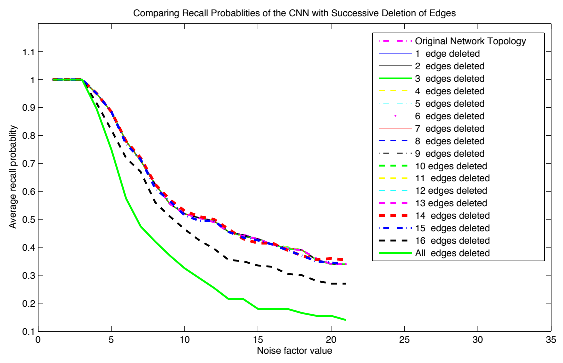


Figure 5: Network performance, parameterized by the number of edges removed.

Figure 5 shows the effect on recall probability of the CNN parameterized by the number of edges deleted. The vertical axis represents the average of the recall probabilities for the four given initial condition vectors and the horizontal axis represents the noise factor or the amplification factor by which the set of noise is multiplied. There is always a gradual decrease in performance as the amount of noise injected in the pattern increases. The recall probability of the original network design evolves with the amount of noise very similar to that of the sparser network produced through the topology optimization process. The additional curves correspond to even sparser networks, and demonstrate that reducing the connectivity further, has an observable impact on network performance. Also if the new $\bar{K}$ value is close to the original $\bar{K}$, performance is still comparable. But when $\bar{K}$ drops significantly compared to its original value, we see a notable performance degradation.

Also we compared the two methods i.e the combinatorial optimization and the random algorithmic method in terms of the stability parameters and number of costly edges deleted. In the simulation, the specified level $\acute{\alpha}$ and the confidence parameter $\acute{\delta}$ values are set to 0.01 and 0.005 respectively. The probability of occurrence of 1s in random matrix, p is set to 0.9 and the simulation is run several number of times. The best value of near maxima of $\bar{K}$ obtained through this method was 108.9599 with deletion of 5 costly edges $[(14,9),(10,9),(13,7),(15,9) and (10,3)]$. Thus it is observed that our designed method of dilution of connectivity performs as good as the randomized algorithm suggested in [17].Further the two methods are compared in terms of recall probability as shown in Fig. 6. As shown in the figure, both methods result in more or less similar recall probability.

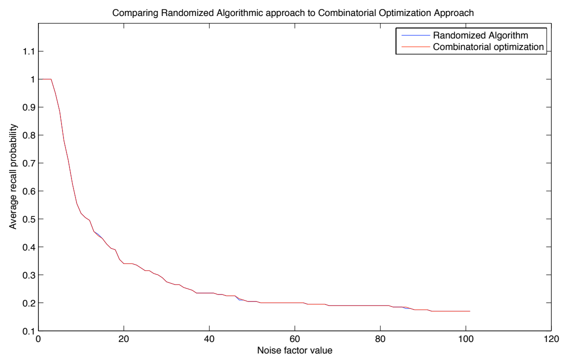


Figure 6: Network's Recall Probability: Randomized Algorithm vs. Combinatorial optimization

6 Conclusions

This paper suggests a topology optimization approach to cellular neural network design, as a method for realizing associative memories using sparser (thus more efficient, or less expensive) networks. The optimization criterion utilizes the stability parameters of the network as a quantitative measure of its ability to recall patterns contaminated with different levels of noise. These stability parameters have been associated with the size of the attraction regions of the neural network. Simulations and comparisons show that a sparser topology can be achieved without significantly degrading the performance of the network, by selectively deleting those weights from the optimized network which contribute the least (as measured by the arithmetic sum of the stability parameters) to ability of the network to recall the desired patterns. Also the efficiency of the cellular network designed by an alternative approach, Randomized algorithm, is on par with the efficiency of the network optimized by combinatorial optimization.

Acknowledgments

This work is supported by NSF-IIS award #0822845.

References

[1] L. O. Chua and L. Yang, "Cellular neural networks: theory and applications," *IEEE Transactions on Circuits and Systems*, vol. 35, pp. 1257–1290, 1988.

[2] A. N. Michel and J. A. Farrell, "Associative memories via artificial neural networks," *IEEE Control Systems Magazine*, vol. 1990, pp. 6–17, 1988.

[3] D. Liu and A. N. Michel, "Cellular neural networks for associative memories," *IEEE Transactions on Circuits and Systems*, vol. 40, pp. 119–121, 1993.

[4] J. Park and Y. Park, "An optimization approach to design of cellular neural networks," *International Journal of Systems Science*, vol. 31, pp. 1585–1591, 2000.

[5] S. Arik and V. Tavsanoglu, "Global asymptotic stability analysis of bidirectional associative memory neural networks with constant time delays," *Neurocomputing*, vol. 68, pp. 161–176, 2005.

[6] J. H. Park, "Robust stability of bidirectional associative memory neural networks with time delays," *Physics Letters A*, vol. 349, no. 6, p. 494499, 2008.

[7] D. Liu and A. N. Michel, "Sparsely interconnected neural networks for associative memories with applications to cellular neural networks," *IEEE Transactions on Circuits and Systems -II, Analog and Digital Signal Processing*, vol. 41, pp. 295–307, 1994.

[8] ——, “Robustness analysis and design of a class of neural networks with sparse interconnecting structure,” *Neurocomputing*, vol. 12, pp. 59–76, 1996.

[9] Y. Nesterov and A. Nemirovskii, *Interior Point Polynomial Algorithms in Convex Programming*. Studies in Applied and Numerical Mathematics, 1994, vol. 13.

[10] S.-I. Amari, “Characteristics of randomly connected threshold-element networks and network systems,” *Proceedings of the IEEE*, vol. 59, no. 1, pp. 35–47, 1971.

[11] W. Krauth and M. Mézard, “Learning algorithms with optimal stability in neural networks,” *Journal of Physics A: Mathematical and General*, vol. 20, no. 11, pp. L745–L752, 1987.

[12] E. Gardner, “The space of interactions in neural network models,” *Journal of Physics A: Mathematical and General*, vol. 21, no. 1, pp. 257–270, 1988.

[13] B. M. Forrest, “Content-addressability and learning in neural networks,” *Journal of Physics A: Mathematical and General*, vol. 21, no. 1, pp. 245–255, 1988.

[14] T. B. Kepler and L. Abbott, “Domains of attraction in neural networks,” *Journal de Physique*, vol. 49, no. 10, pp. 1657–1662, 1988.

[15] A. C. C. Coolen, “On the relation between stability parameters and sizes of domains of attraction in attractor neural networks,” *Europhysics Letters*, vol. 16, pp. 73–78, 1991.

[16] K. E. Kurten, “Adaptive architectures for hebbian network models,” *Journal de Physique I*, vol. 2, pp. 615–624, 1992.

[17] M. Vidyasagar, “Randomized algoriyhms for robust controller synthesis using statistical learning theory,” *Automatica*, vol. 37, pp. 1515–1528, 2001.

[18] S. Boyd, L. Elghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in Systems and Control Theory*. Studies in Applied and Numerical Mathematics, 1994, vol. 15.