Invited Paper

Spatially Distributed Cellular Neural Networks

Varsha Bhambhani, Luis A. Valbuena, and Herbert G. Tanner*

Mechanical Engineering Department, University of Delaware, 126 Spencer Lab Newark, DE 19716, U.S.A

Received (13 May 2011)
Revised (22 May 2011)
Accepted (1 June 2011)

Abstract

Purpose – The objective of this work is to develop a methodology for the design of cellular neural networks with interconnection topologies optimized and suitable for spatially distributed implementation.
Design/Methodology/Approach – We perform combinatorial optimization on the neural network's topology to obtain a sparser network, in which the links between the components of the network that reside in different physical locations are minimized. The approach builds on existing computationally efficient tools for the design of cellular neural networks and uses the concept of the network's stability parameters to assess the performance of the network prior to testing.
Findings – It turns out that the sparser cellular neural networks thus produced exhibit performance that can be on par with that of networks with full connectivity, and that for implementations of modest size, communication delays are not that significant to affect the stability of the dynamical system.
Originality/Value – The novelty of the proposed approach lies in the formulation of the combinatorial optimization problem in a way that trades-off network performance for communication overhead, and the use of this method for the physical implementation of associative memories across different interconnected processors.

Keywords: Neural networks, pattern recognition.

Paper type: **Research paper**

1. Introduction

A cellular neural network is a system of locally interconnected non-linear processing units, implementing an associative memory (Chua & Yang 1988). Research on the design and synthesis of cellular neural networks (CNN) generated significant interest

*This work is supported by the US National Science Foundation, under Grant #0822845.

in recent years because of CNN interconnection topology is relatively sparse, a feature that makes this type of neural networks a promising choice for very-large-scale integration implementations. Our interest in CNNs is motivated by their successful implementations in applications of pattern recognition (see Itoh & Chua (2010), Alekseev (2008), Korbel (2006), Aydogan et al. (2005) for recent examples).

Several synthesis procedures have been developed for designing symmetric and non-symmetric neural network (Michel & Farrell 1988, Liu & Michel 1993, Park & Park 2000). Besides synthesis methods, increased attention is also being paid lately on the issue of stability in the dynamic behavior of these systems in the presence of time delays (see for example (Arik & Tavsanoglu 2005, Park 2008, 2006, Kao & Gao 2008, Zeng et al. 2008)), typically addressed in the context of Lyapunov-Krasovksii functionals, and linear matrix inequalities (LMIs) formulated based on bounds related to these delays. This particular research direction, however, falls outside the scope of this paper, since the patterns we are interested in identifying are static. In addition in our discrete-time software implementation of our networks, the update time step of the difference equations can be stretched sufficiently to encompass these delays (see section 5).

Several well-known network design techniques including the eigenstructure method are discussed in (Michel & Farrell 1988). Analysis results (Liu & Michel 1993) allow one to locate in a systematic manner all equilibrium points of the neural network, and to determine the stability properties of these equilibrium points. Aforementioned methods have been generalized in (Liu & Michel 1994) to include a design procedure for neural networks with sparse connectivity. The results of (Liu & Michel 1994) 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. It is shown that a sufficient condition for the existence of such a sparse neural network design is the availability of self feedback for every neuron in the network. Critical network issues like improved performance, robustness and invariance to perturbation are addressed in (Liu & Michel 1996), where upper bounds for the perturbations of parameters under which desired memories stored in a neural network are preserved, are given. This type of information is of great practical interest during the implementation process of such networks. In (Park & Park 2000), 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. The optimization problem is transformed to a generalized eigenvalue problem (GEVP), which can be efficiently solved using semidefinite programming methods (Nesterov & Nemirovskii 1994). Networks designed using this approach are expected to exhibit fewer spurious patterns and higher recall probabilities (Park & Park 2000).

The approach to the *design* problem for CNNs (without time delays) matured toward the end of the nineties, although some interesting variations still appear (Giaquinto & Fornarelli 2009). In many of these reported approaches, the network

topology is given *a priori*. With the physical platform on which the network is implemented typically given, this starting point is justified. In this paper, however, we try to answer this question of how the design process can incorporate the network topology as part of the design. In scenarios where implementing and maintaining certain neuron connections is more expensive than others, one may be forced to strike a balance between connectivity cost and network performance. Simple stability analysis of neural networks implementing associative memories, suggests that there is a similar relation between the network connectivity and the rate of convergence of the dynamical system in the case of networks where distributed consensus algorithms run. However, in the case of neural networks, 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. However if neurons communicate at a nontrivial cost, there is cost benefit in designing networks that perform just as well, but are sparser.

The *average recall probability* 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 and 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 metric is used (Liu & Michel 1996, Park & Park 2000) to measure the quality of memorization in associative memories. Although this metric quantifies the network's performance accurately and unambiguously, one shortcoming is that it can only be computed *after* a significant number of test inputs is applied on the network which is tested. In other words, it does not allow the designer to *predict* the network's performance without experimentation. This motivates us to use a different performance metric, one that can be utilized earlier in the design process: the network's *stability parameters*. The concept of stability parameters appears in some earlier work on associative memory networks (Amari 1971, Krauth & Mézard 1987, Gardner 1988), as a measure of quality of memorization. Specifically, it has been demonstrated (Forrest 1988, Kepler & Abbott 1988) that the sign and the magnitude of these numbers are related to the size of the attraction regions of the desired memory patters. It is known that for CNNs with n nodes, these attraction regions have boundaries which are $n-1$ dimensional manifolds, but it is recognized that it is quite difficult to precisely describe them (Lu et al. 2011). Although the applicability of the concept of the stability parameters 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 (Coolen 1991), these parameters are generally accepted as a reasonably good metric for the network's performance (Kurten 1992).

This paper suggests a method to implement spatially distributed associative

memories, by optimizing the network topology of a CNN in which communication links between neurons may incur variable cost. This is done by selectively "trimming" network links in an effort to trade network performance for smaller communication cost. Building on existing design tools, we perform combinatorial optimization on a portion of a CNN that includes the *most expensive* interconnection links, using the network nodes' stability parameters as an indication of memorization performance. The proposed process produces a sparser CNN, the performance of which can be comparable to the original network.

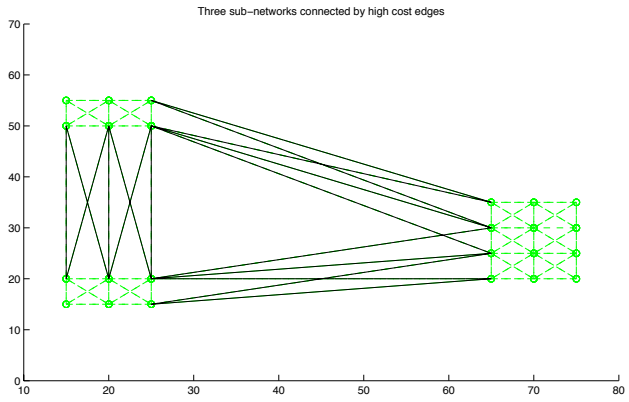


Fig. 1. A CNN interconnection structure where high cost edges are stretched and marked by solid lines. The subnetworks separated by the solid edges are assumed to be implemented in different physical locations.

We tested two different approaches to performing the combinatorial optimization, and verified that the results of the optimization are comparable; in other words, the applicability of the suggested approach does not depend on the optimization algorithm chosen.[a] The first combinatorial optimization method is a sequential process in which a single link is removed each time, specifically the one the removal of which incurs the least performance cost compared to all other existing high-cost links. Although the search in each step of this process is exhaustive, it has to be noted that this method ignores the possibility that a non-obvious combination of links may produce better results through the link's simultaneous removal. This phenomenon has been observed in simulation studies which validated the proposed

[a]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.

method. Motivated by the complexity of the optimization task we also implemented an alternative approach which is based on randomized optimization (Vidyasagar 2001). In this approach a random sample of a sufficient number of independent identically distributed (i.i.d) possible topology choices is generated, and then the optimum is approximated by the element of this sample that performs best. Randomized methods provide probabilistic guarantees of performance and accuracy in terms of the probability that the best solution is not found within a certain subset of the solution space. In our numerical tests, the two methods yield comparable results, which do not differ significantly in terms of resulting network performance, the latter quantified in terms of the network's recall probability. Experiments are performed for a case where a CNN is implemented across three different mobile robots that communicate with each other wirelessly.

The paper is organized as follows: Section 2 provides an introduction to 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. Then section 3.2 presents the randomized topology optimization approach, and compares the results with those obtained in section 3. As an illustrating numerical example, we use a 24-node CNN to compare this sparser topology to the original design, and the numerical results are presented and discussed in section 4. Section 5 presents the real-time hardware implementation of a spatially distributed associative memory, using a small group of mobile robot platforms communicating wirelessly. Finally section 6 concludes the paper by summarizing our findings, including notes on the future research work.

2. Preliminaries

Consider a two-dimensional continuous time zero-input $M \times N$ cellular neural network, as in (Chua & Yang 1988). The dynamics of this system can be represented mathematically as

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

where $1 \leq i \leq M$, $1 \leq j \leq N$, and $\mathrm{sat}\left(x_{ij}\right) \triangleq \frac{1}{2}\left(|x_{ij}+1| - |x_{ij}-1|\right)$. 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 $N_r(i,j) \triangleq \left\{(k,l) : \max\left\{|k-i|, |l-j|\right\} \leq r\right\}$, for $1 \leq i \leq M, 1 \leq j \leq N$. The CNN can be expressed as

$$\dot{x} = -x + T\,\mathrm{sat}\left(x\right) + b \tag{1a}$$

$$y = \mathrm{sat}\left(x\right) \tag{1b}$$

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 , $y = [y_{11}, y_{12}, \ldots, y_{21}, \ldots, y_{MN}]^T \in \mathbb{H}^n$ is the output vector ($\mathbb{H}^n$ 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.

If $\mathbb{B}^n$ represents the set of bipolar vectors in $\mathbb{H}^n$, namely those whose elements are either $+1$ or -1, then for $i = 1, \ldots, n$, the initial condition vectors of (1) should always satisfy $|x_i(0)| \leq 1$. The interconnection topology information 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 $\alpha \in \mathbb{H}^n$ is a *memory vector* for (1) if the latter has an asymptotically stable equilibrium point $\beta \in \mathbb{R}^n$ such that $\alpha = \text{sat}(\beta)$ (Chua & Yang 1988).

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

Problem 1 (Synthesis) *: Given a* CNN *interconnection structure* (1)*, expressed by an adjacency matrix S, and the set of desired bipolar memory vectors $\alpha^1, \ldots \alpha^m \in \mathbb{B}^n$, find the network 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}, typically encodes a lattice structure and can be selected arbitrarily as long as certain conditions are satisfied (Chua & Yang 1988, Liu & Michel 1993, Park & Park 2000): If $\alpha \in \mathbb{B}^n$ and $\beta = T\alpha + b$ are such that

$$\alpha_i \beta_i = \alpha_i \big(\sum_{j=1}^{n} T_{ij} \alpha_j + b_i \big) > 1, \quad \forall i = 1, \ldots, n, \tag{2}$$

then (α, β) is a pair of a memory vector and an asymptotically stable equilibrium point of system (1). Also if $\alpha \in \mathbb{B}^n$ and $\beta = T\alpha + b$ are such that for any $i = 1, \ldots, n$, $\alpha_i \beta_i = \alpha_i \big(\sum_{j=1}^{n} T_{ij} \alpha_j + b_i \big) < 1$, then $\alpha \in \mathbb{B}^n$ cannot be a memory vector. System (1) 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 (Chua & Yang 1988, Liu & Michel 1993, Park & Park 2000):

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

$$\alpha_i \left(\sum_{j=1}^{n} (\widetilde{T}_{ij} \alpha_j + b_i) \right) > 2(k-1) \max_{1 \leq j \leq n} \left| \widetilde{T}_{ij} \right| \tag{3}$$

for $i = 1, \ldots, n$, then any binary vector $\alpha^* \in \mathbb{B}^n$ 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 (1)).

(2) if $x(0) = \alpha^*$ and $\alpha_i^* \neq \alpha_i$, then $x_i(t)$ converges to α_i.

In (3), increasing k results in an increase of both the attractivity and the robustness of the stored memory vector $\alpha \in B^n$ and decrease of the probability of existence of spurious patterns in vertices near α (Park & Park 2000).

The synthesis problem can be formulated as a generalized eigenvector problem (GEVP) (Park & Park 2000). For $i, j = 1, \ldots, n$, and $k = 1, \ldots, m$,

$$\min(-\delta) \quad \text{s.t.} \tag{4a}$$

$$(-\delta)\text{diag}[2q_1, \ldots, 2q_n] - \text{diag}[-p_1, \ldots, -p_n] > 0 \tag{4b}$$

$$\alpha_i^{(k)} \left(\sum_{j=1}^{n} \widetilde{T}_{ij} \alpha_j^{(k)} + b_i \right) - p_i > 0 \tag{4c}$$

$$q_i - \widetilde{T}_{ij} > 0 \tag{4d}$$

$$\widetilde{T}_{ij} + q_i > 0 \tag{4e}$$

$$\widetilde{T}_{ii} = 0 \tag{4f}$$

$$\widetilde{T}_{ij} = \widetilde{T}_{ij}^T = \widetilde{T}_{ij}|_S \tag{4g}$$

$$L < q_i < U, \tag{4h}$$

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) (Boyd et al. 1994), and L and U are the lower and upper bounds for the design variables in the GEVP.

In this paper, we use the magnitude of the stability parameters $K_{i\mu}$ to assess the quality of memorization in the network. The stability parameter for network node i and with respect to memory vector μ is given mathematically as

$$K_{i\mu} = \alpha_i^{(\mu)} h_i = \frac{\alpha_i^{(\mu)} \sum_j c_{ij} \alpha_j^{(\mu)}}{||c_i||_2} \tag{5}$$

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 $K_{i\mu}$ 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 (Kurten 1992, Gardner 1988, Coolen 1991). The stability parameters and the degree of symmetry of connection matrix control the system dynamics and the domain sizes. Positive values of stability parameters indicate that a memory pattern is an attractor.

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

Problem 2 (Topology optimization) *Given* (1)*, implemented on network expressed by a* weighted *graph with adjacency matrix* $\hat{S}$*, with the set of desired bipolar memory vectors* $\alpha^1, \ldots \alpha^m \in \mathbb{B}^n$*, determine the connection weights* T_{ij} *and bias*

parameters b_i of a subnetwork of S, so that this subnetwork stores all desired memory patterns as reachable memory vectors, and recalls them (almost) as well as the complete network.

The performance metric to be used in the optimization algorithm will be formulated using the values of the neural network's stability parameters. 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. Making the Network Sparser: Dilution of Network Connectivity

A CNN is now considered as a collection of sub-networks, which interact with each other over "long-distance" communication links. Communication over these "long-distance" links is assumed more expensive compared to communication between nodes inside each subnetwork. We are interested in minimizing communication cost, while maintaining the functionality and performance of the whole network above a certain threshold.

Given the (unweighted) adjacency matrix S of the CNN, along with the set of desired bipolar memory vectors $\alpha^1, ...\alpha^m \in \mathbb{B}^n$, the first step is to determine the network parameters T_{ij} and b_i through the solution of the GEVP (4). 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 wisely removing some of the high cost links identified in $\widehat{S}$, in a way so that the quality of memorization is not severely affected. The communication costs are captured by the weights of the weighted adjacency matrix $\widehat{S}$. Balancing performance against communication cost is achieved through (combinatorial) optimization over the network links, subject to the stability constraints stated in section 2.

3.1. *Sequential optimization*

This optimization process is iterative. For a given (intermediate) topology S, the algorithm determines the neural network weights T_{ij} and biases b_i for $i, j = 1 \ldots, n$, and based on the selected patterns $\alpha_i^{(\mu)}$ for $\mu = 1, \ldots, m$ to be memorized, an $n \times m$ stability parameter matrix is formed by repeated application of (5): $K = [K_{i\mu}]_{i=1,\ldots,n;\mu=1,\ldots,m}$. For the given patterns $\alpha^1 ... \alpha^m$ 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:

$$\bar{K} = \sum_{i=1}^{n} \sum_{\mu=1}^{m} K_{i\mu} \ . \tag{6}$$

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 3.1. *Several different performance metrics based on different norms of the stability parameter matrix K, such as the minimum row (or column) sums, the (absolute) minimum element of K,* etc*, have been tested as alternatives to* (6)*. 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.*

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

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

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 (3) 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 pseudo-code in algorithm 1 gives an outline of the process for evaluating the edges which are candidate for deletion.

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 (2), or results in an unacceptable performance metric. Although, in general, performance deteriorates as more neuron links are removed, however one may actually increase the objective function using fewer network connections.

3.2. *Randomized topology optimization*

This section suggests another way of finding solutions to the topology optimization problem, in the form of *probable near maxima* of the objective function (6).

Definition 3.2 (cf. (Vidyasagar 2001)) *Let $f : X \to \mathbb{R}$ be a scalar function on a multi-dimensional (parameter) space X, that $\mathbb{P}_X$ is a probability measure on X, and that $\alpha > 0$ is a given real number. A number $f_0 \in \mathbb{R}$ is a probable near maximum of $f(\cdot)$ to level α if $f_0 \geq f^*$, where f^* is the (absolute) maximum of f on X and in addition, $\mathbb{P}_X[x \in X : f(x) > f_0] \leq \alpha$.*

The parameter α essentially quantifies the size of the solution space X on which $f(x) \leq f_0$, in other words, where even better solutions than f_0 may exist. Put differently, if one removes a set of measure α containing the best solutions out of

Algorithm 1 Sequential topology optimization

Require: Matrices α, S, $\widehat{S}$, constants ν, κ.
Ensure: Matrix A of $\bar{K}$ values for each edge that may be removed.

$R \leftarrow \frac{1}{2}\Big(\text{sign}(\widehat{S} - \nu S) + S\Big)$.
$n \leftarrow \text{rowlength}(\alpha)$
$m \leftarrow \text{columnlength}(\alpha)$
$A \leftarrow [0]_{n\times m}$
$C \leftarrow [0]_{n\times m}$
while $\max(R) \neq 0$ **do**
  For each (i, j) such that $R(i, j) \neq 0$, do
  $S' \leftarrow \{S : S(i, j) \leftarrow 0, S(j, i) \leftarrow 0\}$
  Compute T and b, given S' and $\alpha^{(\mu)}$
  $\bar{K} \leftarrow \sum_{i=1}^{n}\sum_{\mu=1}^{m} K_{i\mu}$
  for $\mu = 1$ to m **do**
    for $t = 1$ to n **do**
      $C(t, \mu) \leftarrow \alpha_t^{(\mu)}\big(\sum_{j=1}^{n} T_{tj}\alpha_j^{(\mu)} + b_t\big)$
    end for
  end for
  if $\bar{K} > \kappa \wedge \min C > 1$ **then**
    $A(i, j) \leftarrow \bar{K}$; $A(j, i) \leftarrow \bar{K}$
  end if
  $R(i, j) \leftarrow 0$; $R(j, i) \leftarrow 0$
end while

all of X, the remaining space does not contain a solution that gives a better value than f_0.

Probable near optima can be found using randomized algorithms (Vidyasagar 2001), an approach to optimization that is particularly attractive especially when the functions to be optimized are highly nonconvex, or —as it is the case here— when the problem is of combinatorial nature. A randomized algorithm has always a nonzero probability to give an erroneous solution; in this context, the "erroneous" solution would be one which is not the absolute best out of any remaining solution subset of measure $1 - \alpha$. The probability that the reported solution is not going to be as good is the equal to δ, with $1 - \delta$ being the *confidence* of the randomized algorithm. In other words (Vidyasagar 2001),

$$\mathbb{P}_X[\inf_{x\in X} f(x) \leq f_0 \leq \inf_{x\in X\setminus M} f(x)] = 1 - \delta,$$

where M is any subset of X of measure α. In this context, a randomized algorithm would select a sample $\{x_1, \ldots, x_{N_s}\}$ of N_s i.i.d. possible solutions drawn from some

Algorithm 2 Randomized topology optimization

Require: Matrices α, S, $\widehat{S}$, p,$\acute{\delta}$,$\acute{\alpha}$, ν, $\kappa = 0$.

Ensure: Near maxima of $\bar{K}$ and corresponding topology

1: $R \leftarrow \frac{1}{2}\Big(\text{sign}(\widehat{S} - \nu S) + S\Big)$.
2: $\xi \leftarrow \frac{lg(1/\acute{\delta})}{lg[1/(1-\acute{\alpha})]}$
3: $Row \leftarrow \text{rowlength}(S)$
4: $Column \leftarrow \text{columnlength}(S)$
5: $n \leftarrow \text{rowlength}(\alpha)$
6: $m \leftarrow \text{columnlength}(\alpha)$
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: $\breve{S}(countI, countJ) \leftarrow RandMat(countI, countJ)$
14: **else**
15: $\breve{S}(countI, countJ) \leftarrow S(countI, countJ)$
16: **end if**
17: **end for**
18: **end for**
19: Compute T and b, given $\breve{S}$ 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: $\kappa \leftarrow \bar{K}$; $Topology \leftarrow \breve{S}$.
31: **end if**
32: **end while**

given probability distribution over X, and if

$$N_s \geq \xi \triangleq \left\lfloor \frac{\log(1/\acute{\delta})}{\log[1/(1-\acute{\alpha})]} \right\rfloor,$$

where $\acute{\delta}$, $\acute{\alpha}$ are the confidence and level parameters, respectively, then it is guaran-

teed that the best solution out of the sample of size N_s will be an probable near optimizer of f to level $\acute{\alpha}$ and confidence $\acute{\delta}$.

Given the unweighted adjacency matrix S, the weighted adjacency matrix $\hat{S}$ and the residual adjacency matrix R, we can create random subnetworks $\breve{S}_1, \breve{S}_2, \ldots, \breve{S}_\xi$ where some of the edges in R do not appear. The subnetworks sampled are i.i.d., with the probability of an edge in R linking nodes i and j, to appear in any given one be (an arbitrary) p. In each $\breve{S}_i$ all "low cost" connections in S are included, whereas "high cost" edges appear randomly. Before finalizing the random sample, we test to ensure that each sample topology $\breve{S}_i$ can hold the desired patterns by verifying the stability condition (3). If some network topology fails the stability condition, it is replaced by a new sample, until a complete set of ξ sample topologies with stable CNN dynamics. Then the approximate near optimizer of (6) is given by

$$\breve{\breve{K}} = \max_{1 \le i \le \xi} \breve{K}_i, \tag{7}$$

The process of random generation of ξ networks and determination of approximate near maxima of $\bar{K}$ is pseudo-coded in algorithm 2.

4. A design example and simulations

As an example, we use a CNN that is made up of $n = 24$ nodes (cells) interconnected in a nearest-neighbor lattice structure, with a topology represented by the index matrix S of (8) given here as

$$S = \begin{bmatrix}
0&1&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&1&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0\\
1&1&1&0&0&0&1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0\\
0&1&1&1&0&0&0&1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0\\
0&0&1&1&1&0&0&0&1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0\\
0&0&0&1&1&1&0&0&0&1&0&1&0&0&0&1&1&1&0&0&0&0&0&0\\
0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&0&0\\
0&0&0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1&0&0&0&1&1&1&0&0\\
0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1&0&0&0&1&1&1&0\\
0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1&0&0&0&1&1&1\\
0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&1&0
\end{bmatrix}. \tag{8}$$

The graph associated with this network topology is shown in figure 1. As the figure suggests, the network is divided into three small sub-networks which are linked by expensive connections marked by solid lines. The cost of each connection

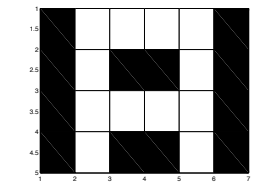
is represented in form of an element of a weighted adjacency matrix $\widehat{S}$ in (9),

$$\widehat{S} = \begin{bmatrix}
0&1&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&2&0&0&0&1&1&2&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&2&0&1&0&0&0&2&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&1&0&0&0&0&2&2&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&0&0&0&1&0&1&0&0&0&2&2&2&0&0&0&0&0&0&0&0&0&0\\
0&1&1&2&0&0&0&1&0&1&0&0&0&2&2&2&0&0&0&0&0&0&0&0&0\\
0&0&2&1&1&0&0&0&2&0&1&0&0&0&2&1&1&0&0&0&0&0&0&0&0\\
0&0&0&1&1&1&0&0&0&1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0\\
0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&2&2&0&0&0&0&1&0&0&0&0&1&1&0&0&0&0&0&0\\
0&0&0&0&0&0&2&2&2&0&0&0&1&0&1&0&0&0&1&1&1&0&0&0&0\\
0&0&0&0&0&0&0&2&2&2&0&0&0&1&0&1&0&0&0&1&1&2&0&0&0\\
0&0&0&0&0&0&0&0&2&1&1&0&0&0&2&0&1&0&0&0&2&1&1&0&0\\
0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1&0&0&0&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&0&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&2&0&0&0&1&0&2&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&2&1&1&0&0&0&2&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&0&1&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&1&0&1&0
\end{bmatrix} . \quad (9)$$

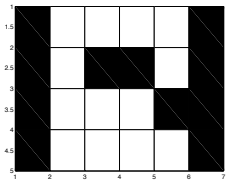
Table 1. Cumulative stability parameter $\bar{K}$ as a result of successive deletion of high cost edges using the sequential combinatorial optimization algorithm.

No. of deleted edges	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

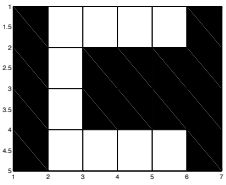
The binary patterns chosen as memory vectors are shown in figure 2. In the initial trial, only the top four memory vectors are considered as associative memory vectors for storage. Each memory vector corresponds to a 6×4 array with black and white boxes represented numerically by -1 and 1 respectively, and is read left to right from top left corner to produce a single column, stack vector of binary



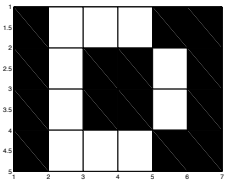
(a) Pattern 1



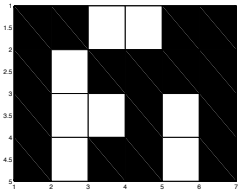
(b) Pattern 2



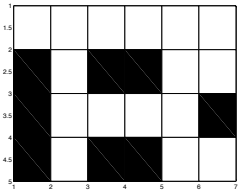
(c) Pattern 3



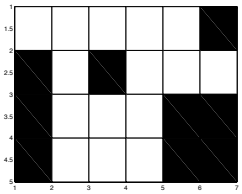
(d) Pattern 4



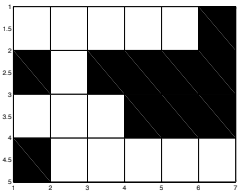
(e) Pattern 5



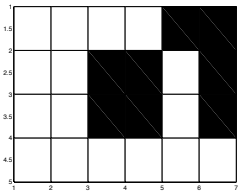
(f) Pattern 6



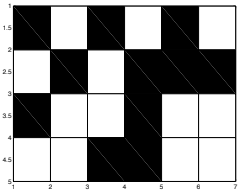
(g) Pattern 7



(h) Pattern 8



(i) Pattern 9



(j) Pattern 10

Fig. 2. Patterns memorized by the CNN. A first implementation of the network included only patterns 1 through 4. A second included all of them, and the two implementations were compared in terms of performance.

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 (4) set to $L = 1$ and $U = 10$.

4.1. *Sequential dilution of network connectivity*

To reduce the communication cost, connection edges from the given set of expensive edges marked by solid lines in figure 1 are considered for possible removal. These edges can be numerically identified by setting a cost threshold ν (below which a connection is thought to be cheap) and evaluating the residual matrix $R = \frac{1}{2}\left(\text{sign}(\widehat{S} - \nu S) + S\right)$. Successive deletion of costly edges results in a different $\bar{K}$ value. Table 1 lists the objective function value resulting from the removal of different sets of expensive edges. For example, deletion of nine costly edges in the sequence shown in table 1 results in a $\bar{K} = 108.3511$ in the example considered.

The performance of proposed design method is quantified in terms of average recall probability of the neural network, as a function of noise contaminating its input. We estimate the recall probability for each noise level by generating a set of 50 different 24×1 arrays of randomly generated elements within a given interval. The elements of the noise array are drawn uniformly from the interval $[-k, k]$, and the level of noise is characterized by the positive constant k (noise factor). We then contaminate each memory vector with each one of these arrays to produce 50 different perturbed versions of each memory vector, at each different noise level quantified by k. Such a perturbed memory vector is denoted $\vec{\alpha}^{(\mu)}$, where μ ranges in $1, \ldots, 4$ (only four memory vectors initially stored). Each element of a perturbed input vector $\vec{\alpha}^{(\mu)}$, is then saturated within the $[-1, 1]$ interval, to ensure that the input vector belongs to $\mathbb{H}^n$. For each memory vector $\alpha^{(\mu)}$, all 50 perturbed version of it are fed to the network, and the number of times for which the network converged to $\alpha^{(\mu)}$ is recorded. From these numbers, and after repeating the process for all $\mu = 1, \ldots, 4$, the recall probability of the network for the particular level of noise is calculated.

Figure 3 shows the recall probability of the CNN with the top four memory patterns of figure 2 stored in the network, as increasingly more neural connections are severed. There is always a gradual decrease in performance as the amount of noise injected in the pattern increases. For moderate to severe network dilution, it is seen that the recall probability of the sparse network remains close to that of the network with the original topology, and only after 16 out of the 17 high cost links are deleted we see a noticeable change in the performance.

4.2. *Randomized dilution of network connectivity*

Our numerical tests suggest that the type of algorithm used for the combinatorial optimization of network connectivity has no effect on the outcome of the design process. In this section, we compare the original (not optimized) network model, with the neural network having its topology improved sequentially, and the and the one derived by randomized algorithm approach. The comparison is made in terms of the average recall probability, and results of the comparative study are shown in figure 4.

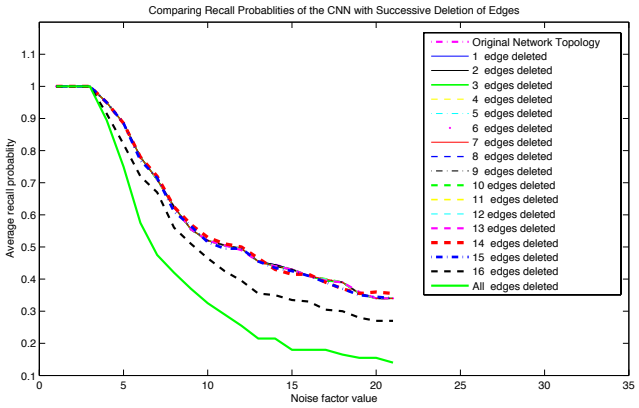


Fig. 3. Network performance, parameterized by the number of links removed. The horizontal axis marks the level of noise injected to the input of the network, and the vertical axis is the average recall probability of the network for that particular level of noise.

It can be seen that the original network topology has roughly same recall probability graph as the sparser topology derived by the two proposed optimization approaches. As said before, the sequential trimming optimization algorithm will trim as many expensive links as possible while keeping the performance metric above the threshold. For example, in the present case the network is trimmed till it reaches the maximum cumulative stability parameter value for the sparer network so obtained ($\bar{K}$=108.3511). If this threshold is set of a relatively small value, however, then the algorithm will progressively remove a much longer set of links and there will be a noticeable decrease in network performance as shown in figure 3. The objective of optimization is thus to balance communication cost versus average recall probability.

For the case of randomized optimization, the specified level $\acute{\alpha}$ and the confidence parameter $\acute{\delta}$ are set to 0.01 and 0.005 respectively. The probability of occurrence of links in the random matrix "RandMat" of algorithm 2, p, is set to 0.9 and the simulation is run ξ number of times. The probable near maximum of $\bar{K}$ obtained using this method is 108.9599, with deletion of 5 costly edges $[(14,9),(10,9),(13,7),(15,9),(10,3)]$. As shown in the figure 4, both optimization methods result in similar recall probability.

4.3. *More memorized information: less accurate recollection*

Intuitively, when more memory patterns are stored in an associative memory, and given that all memories have to share the same state space, the regions of attraction of each individual memory vector is reduced. The implication of this fact is that the associative memory with more stored patterns is less robust to noise (is more likely to fail to recollect accurately) compared to the one that stores less information.

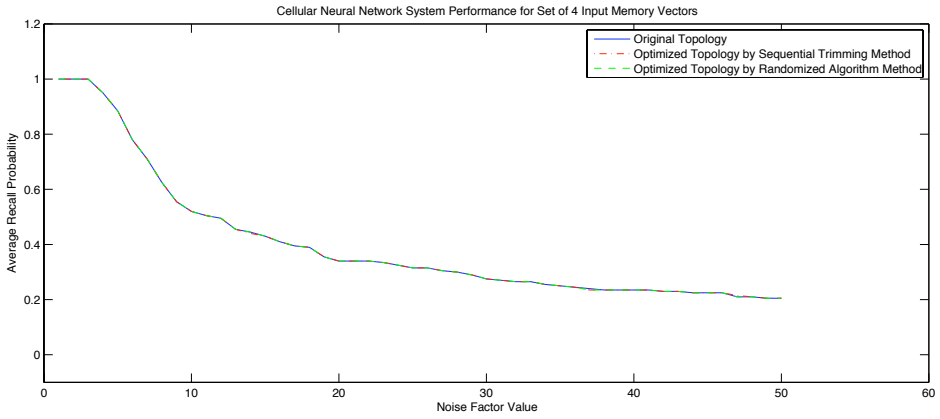


Fig. 4. Network recall probability for a set of four memory vectors. The overlapping curves suggest that in terms of performance, the original network and the ones with diluted connectivity are essentially indistinguishable, irrespectively of the method used for optimization.

Toward this end, we now consider all ten input patterns shown in figure 2.

Figure 5 depicts the effect of the additional requirements for memorization on the ability of the network to reject noise in its input patterns. It is seen that increasing the number of stable attractors in the system reduces the average recall probability, primarily due to the decrease of the attraction regions of each stable equilibrium.

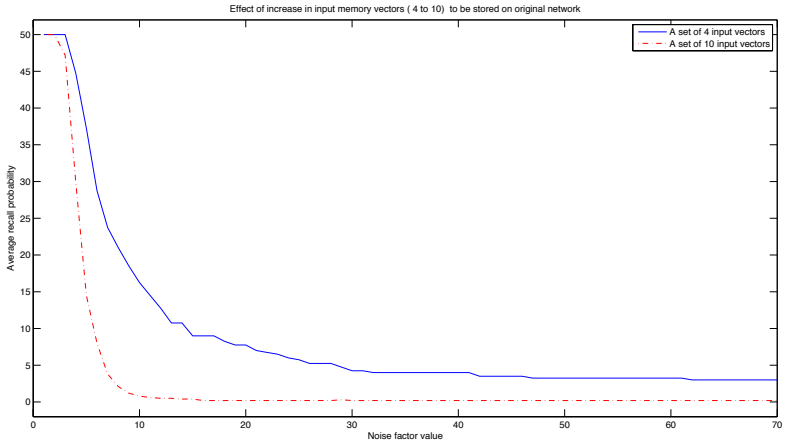


Fig. 5. Comparison between the recall probability of an optimized network with four memories (solid), and an optimized network with ten memories (dashed). As the stable attractors multiply, their individual attraction regions shrink and it is easier for a noisy pattern to be mistaken for another.

The numerical tests reported in this section also verify that the process of dilution of network connectivity has the same overall effect on the network, irrespectively of how much information is engraved in the memory of the system. The objective is to test the ability of the network to store additional patterns with lean interconnection topologies. We compare the recall probability of the original (not optimized) network, the neural network obtained from the application of the sequential optimization algorithm 1, and the one designed using the randomized algorithm 2. Results indicate that while the recall probability of the networks where ten memory vectors are stored are clearly lower compared to that of the networks with only four memory vectors (figure 4), for the same number of stored memories, the performance of the neural network is not adversely affected by a moderate dilution of connectivity. All three CNN topologies, where ten memories are stored, behave equally well.

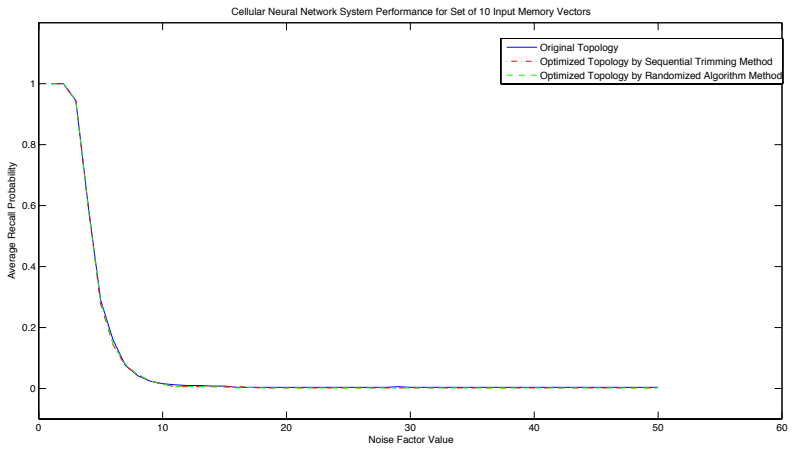


Fig. 6. Network's Recall Probability for set of 10 memory vectors: Comparison among the performance of original network, one optimized using a sequential and a randomized optimization algorithm

5. Experiments with a CNN implemented across different robots

This section describes the physical implementation of a CNN in a spatial distributed way, on a group of three mobile robots interconnected through wi-fi. The mobile robot platform used in this experiment is the Corobot®, manufactured by Coroware Inc. The Corobot is a mobile robot platform equipped with a PC class CPU (1.5 GHz Via C-7) which communicates either wirelessly (wi-fi) or through Ethernet. In our implementation, each Corobot is assigned a fixed IP address, and serves both as a client as well as a server at the same time. When it is transmitting

information to other Corobots on the network, it behaves as a client, whereas when receiving information (nodes, states, status flags) from other Corobots, it behaves as a server that creates a new thread per each incoming communication request from other Corobots. The communication channel is not continuous; the channel is closed after each transmission and is re-established before a new information has to be transmitted.

Fig. 7. The experimental set-up, where a CNN is implemented over a wireless network of three mobile robots.

The network of figure 1 is thus implemented in a way that each of the three subnetworks shown is realized on a different Corobot, and the high cost links between the subnetworks are implemented over wireless channels (figure 7). The network topology is same as represented by connectivity shown in (8) and the cost of connections are represented by the elements of the weighted adjacency matrix $\widehat{S}$ (9). The objective is to experimentally test the hypothesis that a CNN can be implemented in a spatially distributed way, and that the unavoidable communication delays related to wireless communication do not necessarily destroy the stability properties of the dynamical system (1).

To test this hypothesis, we adopted the discretized form of the CNN dynamics (1) proposed in (Liu & Michel 1994) as a system of difference equations

$$x_i\big((k+1)h\big) = \left[x_i(kh) + h\sum_{j=1}^{n} T_{ij}y_j(kh) + \frac{b_i}{a_i}(e^{a_i h} - 1)\right] e^{-a_i h} \tag{10a}$$

$$y_i(kh) = \mathrm{sat}\big(x_i(kh)\big) \tag{10b}$$

where h is the step step duration, k indexes the current time step and i marks the particular component of the state vector. System (10) represents a time-discretization of the continuous-time dynamics (1) under the assumption of a small time step. Indeed, it is verified in our simulations that relatively small values for h result in trajectories for the state of the CNN which are almost indistinguishable from those of the continuous-time equations. In addition, and for the time step duration that was found sufficient for our experimental implementation ($h = 0.05$ sec), the error between simulation of the discrete-time dynamics (10) and experimental

results practically coincide, with a maximum error difference of the order of 10^{-4}.

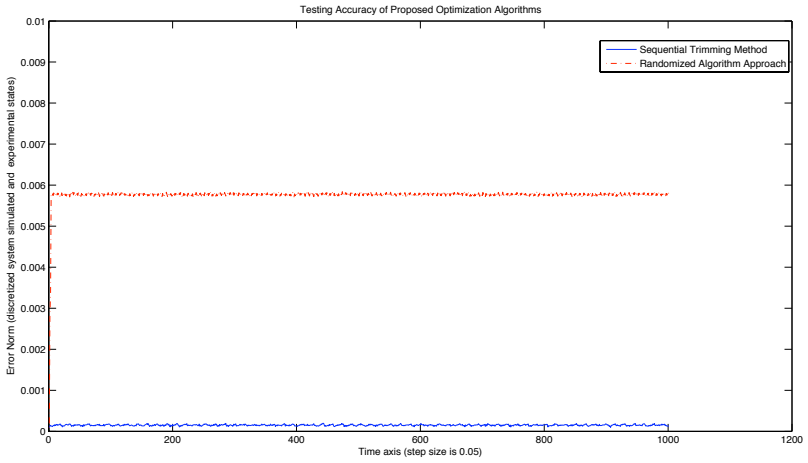


Fig. 8. The norm of the difference between the simulated trajectories of the continuous-time equations (1) and the time-discretized difference equations (10) as implemented on the experimental set-up, for the cases of a network optimized using the sequential combinatorial optimization algorithm (lower blue curve) and the randomized algorithm (upper red curve).

Obviously, as h grows, the trajectories of (1) and (10) diverge, and eventually the CNN as implemented physically becomes unstable. In the case tested here, the time step required was sufficiently small for the delays related to communication be accommodated. If an application imposes larger communication delays which cannot be accommodated by the length of the discrete-time step, one possible solution would be to "stretch" the time axis of the discrete-time system, "freezing" the evolution of the discrete-time system between steps for a period that allows the dissemination of information between subnetworks. In this way, convergence speed is traded-off for increased robustness to communication delays, and stability of the network (in terms of convergence to memory vectors) is not affected.

6. Conclusions

It is possible to selectively sever links between neurons of a cellular neural network without significantly affecting its ability to recall the patterns engraved in its memory. This can be done by the application of a combinatorial optimization algorithm, in conjunction with existing design tools for cellular neural networks, where the objective function is built using the network's stability parameters. The type of optimization algorithm used is immaterial. This type of network optimization yields sparser interconnection topologies, which in turn allows for physical implementations of these networks which are spatially distributed, and in which the links that

connect neurons that reside in different physical locations are as small as possible. Numerical and experimental testing has shown that such spatially distributed implementations are indeed feasible, and that for some cases, the communication delays related to the communication between the different components of the network are not significant enough to affect the performance and stability properties of the dynamical system.

References

Alekseev, S. A. (2008), 'Polarization method for recognizing the shape of a surface from the shading', *Journal of Optical Technology* **75**(2), 90–93.

Amari, S.-I. (1971), 'Characteristics of randomly connected threshold-element networks and network systems', *Proceedings of the IEEE* **59**(1), 35–47.

Arik, S. & Tavsanoglu, V. (2005), 'Global asymptotic stability analysis of bidirectional associative memory neural networks with constant time delays', *Neurocomputing* **68**, 161–176.

Aydogan, D., Elmas, A., Albora, A. M. & Ucan, O. N. (2005), 'A new approach to the structural features of the Aegean sea: Cellular neural network', *Marine Geophysical Researches* **26**, 1–15.

Boyd, S., Elghaoui, L., Feron, E. & Balakrishnan, V. (1994), *Linear Matrix Inequalities in Systems and Control Theory*, Vol. 15, Studies in Applied and Numerical Mathematics, Society for Industrial and Applied Mathematics.

Chua, L. O. & Yang, L. (1988), 'Cellular neural networks: theory and applications', *IEEE Transactions on Circuits and Systems* **35**, 1257–1290.

Coolen, A. C. C. (1991), 'On the relation between stability parameters and sizes of domains of attraction in attractor neural networks', *Europhysics Letters* **16**, 73–78.

Forrest, B. M. (1988), 'Content-addressability and learning in neural networks', *Journal of Physics A: Mathematical and General* **21**(1), 245–255.

Gardner, E. (1988), 'The space of interactions in neural network models', *Journal of Physics A: Mathematical and General* **21**(1), 257–270.

Giaquinto, A. & Fornarelli, G. (2009), 'Pso-based cloning template design for CNN associative memories', *IEEE Transactions on Neural Networks* **20**(11), 1837–1841.

Itoh, M. & Chua, L. O. (2010), 'Autoassociative memory cellular neural networks', *International Journal of Bifurcation and Chaos* **20**(10), 3225–3266.

Kao, Y. & Gao, C. (2008), 'Global exponential stability of cellular neural networks with variable coefficients and delays', *Neural Computation & Applications* **17**, 291–295.

Kepler, T. B. & Abbott, L. (1988), 'Domains of attraction in neural networks', *Journal de Physique* **49**(10), 1657–1662.

Korbel, P. (2006), 'Cellular neural network-based object recognition with deformable grids', *Machine Graphics and Vision* **15**(3-4), 461–469.

Krauth, W. & Mézard, M. (1987), 'Learning algorithms with optimal stability in neural networks', *Journal of Physics A: Mathematical and General* **20**(11), L745–L752.

Kurten, K. E. (1992), 'Adaptive architectures for hebbian network models', *Journal de Physique I* **2**, 615–624.

Liu, D. & Michel, A. N. (1993), 'Cellular neural networks for associative memories', *IEEE Transactions on Circuits and Systems* **40**, 119–121.

Liu, D. & Michel, A. N. (1994), '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* **41**, 295–307.

Liu, D. & Michel, A. N. (1996), 'Robustness analysis and design of a class of neural networks with sparse interconnecting structure', *Neurocomputing* **12**, 59–76.

Lu, W., Wang, L. & Chen, T. (2011), 'On attracting basins of multiple equilibira of a class of cellular neural networks', *IEEE Transactions on Neural Networks* **22**(3), 381–394.

Michel, A. N. & Farrell, J. A. (1988), 'Associative memories via artificial neural networks', *IEEE Control Systems Magazine* **1990**, 6–17.

Nesterov, Y. & Nemirovskii, A. (1994), *Interior Point Polynomial Algorithms in Convex Programming*, Vol. 13, Studies in Applied and Numerical Mathematics.

Park, J. H. (2006), 'Global exponential stability of cellular neural networks with variable delays', *Applied Mathematics and Computation* **183**, 1214–1219.

Park, J. H. (2008), 'Robust stability of bidirectional associative memory neural networks with time delays', *Physics Letters A* **349**(6).

Park, J. & Park, Y. (2000), 'An optimization approach to design of cellular neural networks', *International Journal of Systems Science* **31**, 1585–1591.

Vidyasagar, M. (2001), 'Randomized algorithms for robust controller synthesis using statistical learning theory', *Automatica* **37**, 1515–1528.

Zeng, Z., Huang, D.-S. & Wang, Z. (2008), 'Pattern memory analysis based on stability theory of cellular neural networks', *Applied Mathematical Modeling* **32**, 112–121.