2009

Vibrational control of chaos in artificial neural networks

Ralph Bean

Vibrational Control of Chaos in Artificial Neural Networks

Ralph Bean
Department of Computer Science
Rochester Institute of Technology
ralph.bean@gmail.com

August 18, 2009

Abstract

Neural networks with chaotic baseline behavior are interesting for their experimental bases in both biological relevancy and engineering applicability. In the engineering case, the literature still lacks a robust study of the interrelationship between particular chaotic baseline network dynamics and "online" or "driving" inputs. We ask the question, for a particular neural network with chaotic baseline behaviour, what periodic inputs of minimal magnitude have a stabilizing effect on network dynamics? A genetic algorithm is developed for the task. A systematic comparison of different genetic operators is carried out where each operator-combination is ranked by the optimality of solutions found. The algorithm reaches acceptable results and finds input sequences with largest elements on the order of 10^{-3}. Lastly, an illustration of the complexity of the fitness space is produced by brute-force sampling period-2 inputs and plotting a fitness map of their stabilizing effect on the network.

Thesis Committee
Adviser: Dr. Roger Gaborski
Observer: Dr. Peter Anderson
Reader: Thomas Borelli

Contents

List of Figures

List of Tables

1 Introduction

The computational tasks of nonlinear classification and control are of high importance to those interested in vision and acoustics. The information of interest in real world signals is most often embedded intrinsically in their temporal structure. Examples include distinguishing between and producing the subtleties of human speech and predicting natural time series such as weather patterns and radioactive decay [23, 8].

For many tasks, humans out-compete state-of-the-art computational techniques hands-down in both accuracy and flexibility.

Consequently, techniques regarding recurrent neural networks are of interest to computing research as a potential solution which could harness the hard work already done for us computer scientists by the process of biological evolution itself.

A large amount of theory has been developed with regard to feed-forward neural networks from how to train them in both supervised and unsupervised paradigms as well as why and how they work. Things are not as nice when we consider networks that have recurrent connections or feed-back connections. Doya [6] provides an excellent description of the problems confronting gradient descent for training recurrent networks.

The initial inspiration for this thesis came in 2001 from the work of Jaeger [22]. The Echo State Network (ESN) described there combined a randomly generated recurrent network with a linear readout mechanism trained to classify the state of the recurrent network. It was observed that systems that had randomly generated reservoir networks with dynamics at the 'edge of chaos' performed better at temporal discrimination tasks. This thesis treats itself as a part of a process attempting to theoretically explain the experimental success of Jaeger and others' work [17].

D. Verstraeten [23] was the first to use Lyapunov exponents as a measurement of chaos in the context of Jaeger's work. It must be stated that Verstraeten's application was built on a misunderstanding of Lyapunov exponents. In [23], only the first iteration of the map defined by the network was considered in calculating Lyapunov exponents. Lyapunov exponents involve taking the limit of infinite compositions of the map with itself which, for transcendental activation functions like the hyperbolic tangent function most often considered for artificial neural networks, is incalculable but can be approximated

using numeric methods [21].

The point of this thesis is to clarify notions of the edge of chaos as they apply to artificial neural networks. Sections 2 and 3 briefly describe some background in artificial neural networks and dynamical systems theory. Section 4 describes some examples of chaotic networks with no online input. Section 5 relays some of the concepts and methods from control theory. Section 6 details a genetic algorithm developed to investigate stabilizing input streams for the network in section 4.2 along with some preliminary results. Section 7 reports results from the experiments devised in section 6. Section 8 is a discussion of the results and section 9 contains concluding remarks.

2 Artificial Neural Networks (ANNs)

Artificial Neural Networks are a computational model of natural brains that consist of a network of artificial neurons sometimes also called perceptrons. These artificial neuron models activate more or less in response to each other and a series of inputs as governed by a system of equations [22, 6, 10]. They are perhaps best contrasted with the Biological Neural Network or Spike/Pulsed Neural Network computing models which are more biologically realistic but more computationally expensive and more theoretically unwieldy [16, 11, 13].

ANNs are defined by a choice of a weight matrix and activation function. During simulation, the state of the ith neuron is calculated from the weighted sum of the state of all connected neurons in the previous timestep under composition with the activation function.

$$W = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,N} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N,1} & w_{N,2} & \cdots & w_{N,N} \end{pmatrix}$$

$$x_{i,t+1} = f(\Sigma_{j=0}^{N} w_{i,j} x_{j,t})$$

$$\vec{x}_{t+1} = f(W\vec{x}_t)$$

Although in some research, a variety of activation functions are chosen, the two studied here will be the hyperbolic tangent function

$$f(x) = tanh(\mu x)$$

and the exponential sigmoidal function.

$$f(x) = (1 + e^{-\mu x})^{-1}$$

The hyperbolic tangent function has been proposed to best model the 'firing rate' of more biologically plausible neural models. Furthermore, the universal approximation theorem states that a feed-forward ANN with the hyperbolic tangent as the activation function and with a sufficiently large hidden layer is able to approximate any continuous function arbitrarily closely [4] which will be necessary for a construction in section 4. The exponential sigmoidal function is relevant in its ability to elicit chaotic behavior in networks with simple weight matrices

3 Chaos in Nonlinear Systems

A function $f : I \to I$ where $I \subset \mathbb{S}$ is said to be chaotic [12] iff:

1. Periodic points of f are dense in I;
2. f is transitive on I; that is, given any two subintervals U_1 and U_2 in I, there is a point $x_0 \in U_1$ and an $n > 0$ such that $f^n(x_0) \in U_2$;
3. f has sensitive dependence in i; that is, there is a *sensitivity constant* β such that, for any $x_0 \in I$ and any open interval U about x_0, there is some seed $y_0 \in U$ and $n > 0$ such that

$$|f^n(x_0) - f^n(y_0)| > \beta$$

The set of periodic points of f being dense in I amounts to saying that there are many such periodic points. More formally, a set S is dense in an interval I if any open subinterval of I contains a point of S. In this case, any subinterval of I you choose contains a point on a periodic orbit of f.

The second condition, the transitivity of f, amounts to a *mixing* requirement. Points from U_1 at some time end up in U_2 and points from U_2 at some time end up in U_1. This follows from the above property of the density of periodic points of f.

The third property, sensitive dependence on initial conditions alternatively known as 'the butterfly effect', is the most popularly understood and the more hotly contested aspect of chaos. See [9] for a thorough treatment of the confluence of Lorenz and non-mathematicians on broad cultural misinterpretations of chaos theory.

A reasonable alternative to this condition involves characterizing the *rate* of separation of orbits by calculating the system's spectrum of *Lyapunov exponents*.

For a trajectory through the state space, consider the effect had on the points on a hypersphere of arbitrarily small radius centered on the initial seed of the orbit. Over the evolution of the system, this hypersphere evolves into a hyperellipsoid [1, 18, 23]. In short, numerically approximating the Lyapunov exponents means iterating the system into its future and measuring these axes [21].

Where r_k^n is the kth longest orthogonal axis of the hyperellipsoid at the nth iterative application of f, the kth Lyapunov exponent, h_k, is defined as

$$h_k = \log\left(\lim_{n\to\infty}(r_k^n)^{\frac{1}{n}}\right)$$

Although we are most interested in the value of h_k (is the rate of separation exponential or not?), introducing the kth *Lyapunov number* L_k simplifies the equation to

$$L_k = e^{h_k} = \lim_{n\to\infty}(r_k^n)^{\frac{1}{n}}$$

4 Construction of Chaotic Recurrent Networks

By the universal approximation theorem, a feedforward artificial neural network with three layers of cells and two layers of connections is able to approximate any continuous function arbitrarily closely with a sufficiently large hidden layer [4]. There are known chaotic maps. For instance, the chaotic dynamics of the logistic map $f_4(x) = 4x(1-x)$ are well understood [1, 12, 7, 21].

4.1 Training on the Logistic Map

Consider a feed-forward network with topology $1 \to K \to 1$ that approximates $f_4(x) = 4x(1-x)$ with some error ϵ. Remove the single-neuron output layer and replace all connections from the K hidden neurons to that neuron with identical connections back to the input layer. It is clear that the network is now recurrent and that its continued simulation computes the iterative composition of the function the feed-forward network originally modeled. If the error ϵ between f_4 and

the modeled function is not too great, the iterated map now modeled by the network should be qualitatively equivalent to the iteration of f_4 and orbits of the network should exhibit chaotic motion.

The above was carried out by training a network to approximate f_4 with standard back-propagation. Experimental results are shown below in figures 1, 2, and 3.

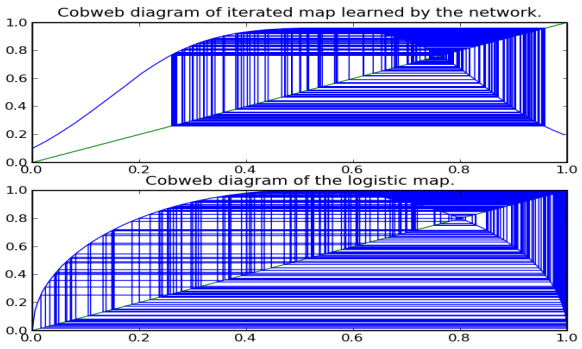

Figure 1: Cobweb diagrams are one way of visualizing the dynamics of iterated one-dimensional maps. To generate the above images, a random seed between 0 and 1 was chosen. The orbit was iterated 2000 times and only the last 200 iterations are plotted to ensure no stabilization from a seed chosen too far from the potentially strange attractor.

Figure 2: Plot of the same orbits shown in the cobweb diagrams but against the timestep.

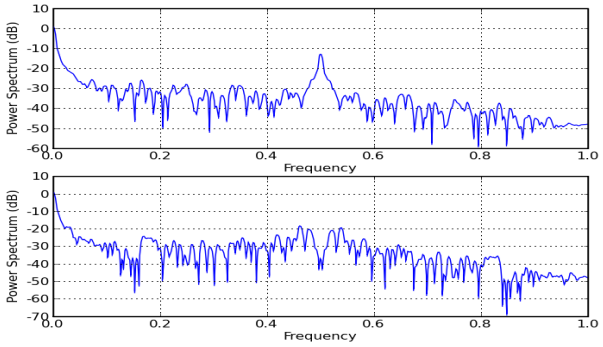

Figure 3: Power spectral density functions of the network and its target map. In each case, the logarithm of the psd is plotted. Note that the functions appear to approximate *log(1/f)*

The same process conceivably applies to other chaotic functions such as $f(x) = \mu x^2(1-x)$ where $\mu = 6.5$ and $f(x) = \mu x^3(1-x)$ where $\mu = 8.1$. Similar results were obtained but are not included in this document.

4.2 Other Known Chaotic Networks

In [20] a network that exhibits chaotic behavior is described. The authors there document a 2-neuron network with weight matrix $W =$

$$\begin{pmatrix} -a & a \\ -b & b \end{pmatrix}$$

where $a = 5$ and $b = 25$ and activation function $f_\mu(x) = (1 + e^{-\mu x})^{-1}$ (see figures 4 and 5).

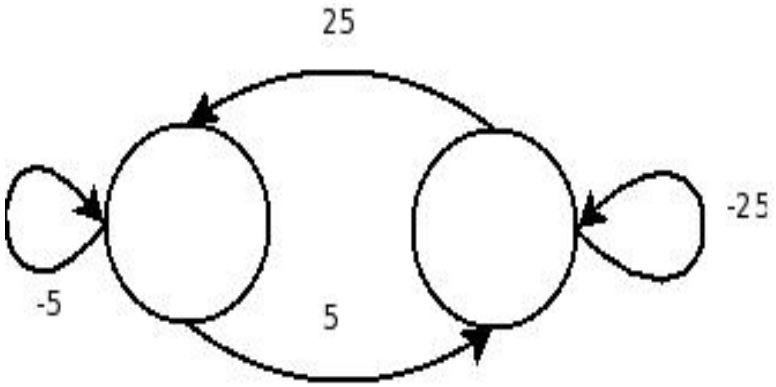

Figure 4: Diagram showing arrangement and weighted connection of neurons in the network above.

For different values of μ, this two-neuron network demonstrates many different classes of dynamical phenomena including the so-called period-doubling route to chaos as can be seen in figure 6. Because of its simplicity, it is the network used in experiments here.

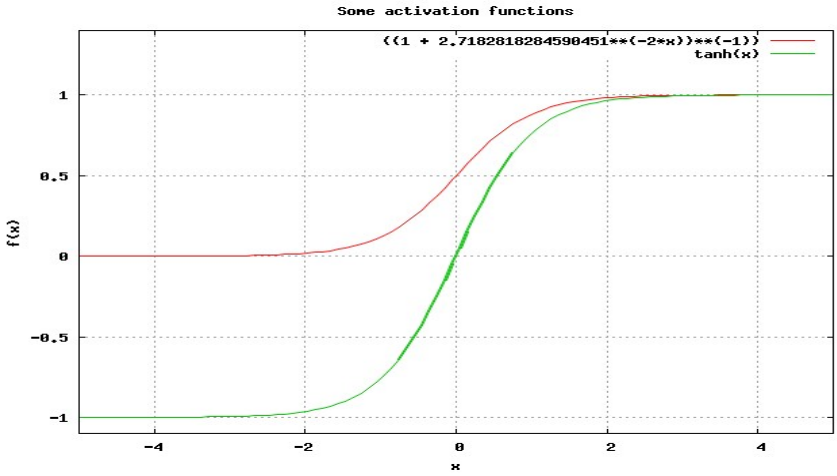

Figure 5: Two example activation functions. One, the hyperbolic tangent. Two, the sigmoidal function parameterized by μ used in the network above.

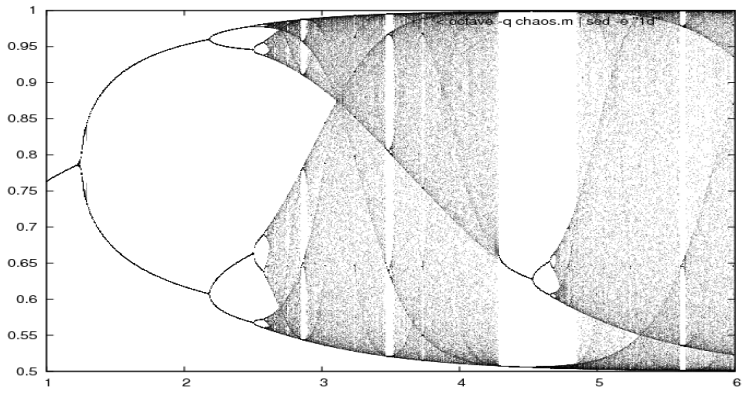

Figure 6: Bifurcation diagram of the network discussed in [20] and used as the system to be controlled in this thesis' experiments. The horizontal axis is the parameter μ and the vertical axis is an approximation of one of the neurons' values along the system's attractor.

5 Control Theory

A body of literature exists documenting theory and methods for the control of chaotic systems. Control of chaos falls into one of two complimentary tasks: stabilization and chaotization[2]. Stabilization is defined as inducing orderly behavior in what would otherwise be an autonomous chaotic system. Chaotization is the other side of the coin: inducing unpredictable or eratic behavior from what would otherwise be an autonomous stable system. Bridging the two tasks are five main methods: Open-loop control, Linear and Nonlinear Control, Adaptive Control, Linearization of the Poincare Map (OGY), Time-Delayed Feedback (Pyragas Method)[2].

The task of this thesis, stabilization of an unknown, chaotic, discrete-time neural network, rules out the use of most of these documented approaches. They are inappropriate either because they apply only to continuous systems and not discrete, or because they require foreknowledge of the system to control, or because they are strictly for the chaotization and not the stabilization of the target system.

Stabilization of chaos using the OGY method has been documented for discrete time neural networks [24]. OGY relies on the dynamical systems theorem stating that there are an infinite number of periodic repelling orbits dense in the chaotic interval. In summary, OGY operates in two phases. First, the system's trajectory is observed. Once a point is observed that is within some ϵ of a point observed in the past, it can be surmised that all points that lay historically between the past observation and the present are quite close to a repelling periodic orbit. Without intervention, the trajectory would spin off and not repeat itself. Knowing that the current state of the system is near a periodic orbit, the second phase of OGY application is to apply a minute perturbation to close the gap between the previously observed and presently observed close points. This 'correction' is applied at each subsequent mapping of the system. The effect is to induce a convergence of the orbit towards an otherwise repelling periodic orbit. A repeated orderly behavior is achieved with a high success rate for small ϵ. The experiments in [24] were duplicated successfully but results are not included in this document.

At its heart, OGY demands that the the controlling method have some knowledge of the baseline dynamics of the system-to-be-controlled. This makes it unsuitable for the purposes of this thesis.

What remains applicable is open-loop control (sometimes called

vibrational control). Although there is a wealth of applications in the literature, few are to discrete-time systems and of those, most are steeped in a mathematical language unfamiliar to the author of this thesis.

The gist of open-loop or vibrational control is to find a finite-length set of perturbations to the controlled system that elicit a periodic behavior in its otherwise aperiodic dynamics, *regardless of whether or not its state at the time of perturbation is near a particular dynamical phenomena* as in OGY. This represents a breaking of the control-observation loop (hence, 'open-loop') and is desirable since, in engineering tasks of visual and acoustic discrimination, the stimuli impinging on a neural network do not adapt themselves to the network's output.

The control theory literature furthermore imposes a requirement on vibrational techniques. The *amplitude* of the perturbing impulses must be sufficiently small that the perturbations do not simply override or 'wash out' the dynamics of the system [2].

6 Genetic Algorithm

The above construction from section 4, as shown, can produce recurrent networks with aperiodic motion under no input. The spirit of neural network application, however, is to perform some task on an input.

The question here asked is "does there exist a finite length input sequence which, when presented under repetition, can stabilize an otherwise chaotic neural network?"

The question is motivated by the reservoir computing application [22, 17]. There, for a readout layer to converge on a decision, there must be some stabilization of the reservoir. Input sequences with no capacity for stabilization will elicit seemingly random patterns from the network. Sequences that correspond to a stabilization of the reservoir will instead produce a sequence of reservoir state vectors amongst which a readout layer can discriminate.

A genetic algorithm is developed to find such sequences for the network from section 4.2. Genetic algorithms are a technique designed to use the principles of natural Darwinian evolution towards solving computational problems. Genetic algorithms (GAs) are distinguished by five components: a population of 'individuals' that are themselves

potential solutions, a fitness metric to rank how 'fit' each individual solution is or is not, a crossover operator to breed and produce 'child solutions' from pairs of individuals, a mutation operator to introduce new genetic material to the pool of evolving solutions, and a selection operator to determine in what way individuals are paired for crossover (reproduction) and in what way individuals are selected for discard (death).

6.1 Genetic Operators

In the application here, individuals in the population are tuples consisting of a finite, variable length numeric sequence S and an amplitude coefficient α.

Fitness of a $\langle S, \alpha \rangle$ tuple is determined by approximating the maximal Lyapunov exponent of the network under perturbation by the sequence scaled by an amplitude coefficient. The maximal Lyapunov exponent is measured as described in [21]. Loosely, it is a measure of 'how stable' a system is. A positive maximal Lyapunov exponent indicates exponential expansion of orbits along at least one dimension of the system. A negative maximal Lyapunov exponent indicates exponential contraction along *all* dimensions. For this genetic algorithm, minimal fitness is 'better' fitness: exponential contraction of neighboring orbits indicates stabilization.

Two major rounds of genetic experiments were run: preliminary and systematic.

6.2 Preliminary Experiments (and results)

In a first batch of experiments, the crossover operator is single-point-crossover where two selected parents have indices randomly selected from their sequences and two complimentary children are created from the splicing together of each pair of subsequences.

The mutation operator alters a single element of a child sequence and replaces it with a randomly chosen valid element.

Below are some preliminary results and observations that motivate the second experiment.

6.2.1 Binary Alphabet

The first experiment run allowed for sequences to be composed only of elements from the alphabet $-1, 1$ and for a fixed amplitude coefficient

$\alpha = 0.00001$. The choice of a value for α was arbitrary but was chosen to be quite small as motivated by the discussion in section 5 on vibrational control.

Two sub experiments were attempted with two different genetic selection methods.

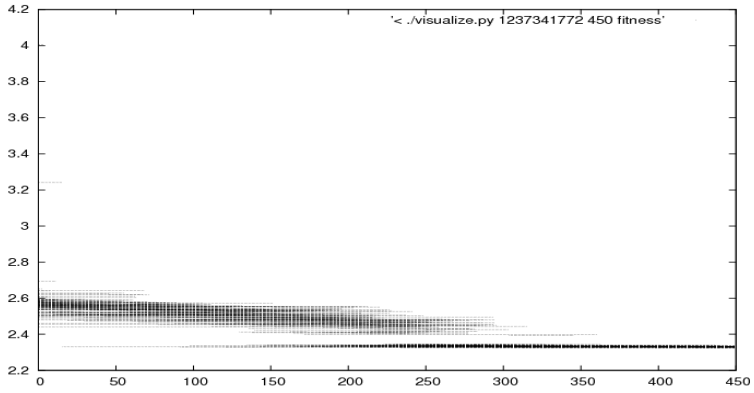

Figure 7: Evolutionary diagram of an early experiment with greedy selection and a binary alphabet for input sequence candidates. The population converges early to an unacceptable solution. Ticks on the horizontal axis are generations of the population and ticks on the vertical axis are fitness values (maximal Lyapunov exponents).

Experiment 1 used the 'greedy' method where a child only replaces its parent if its fitness is superior. As can been seen from figure 7, this apparently lead to premature convergence to a non-optimal solution. The entire population converges to sequences with maximal Lyapunov exponents of about 2.35 which is non-negative indicating a still chaotic network.

Experiment 2 did away with greedy selection and placed children in the population without regard to their fitness. From figure 8, it appears this method effected an increase in diversity and developed a sequence that could stabilize the network, a sequence with a negative maximal Lyapunov exponent.

If the hero sequence did in fact stabilize the network, then we ought to see a dramatic difference in the power spectral densities of the system with and without the sequence present. The power spectral

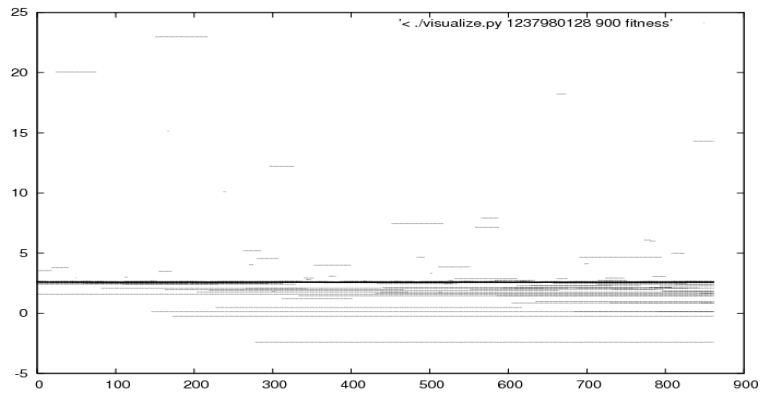

Figure 8: Evolutionary diagram of another early experiment with non-greedy selection and a binary alphabet for input sequence candidates. What appears to be an acceptable sequence is found early on but the population doesn't converge to it.

density function (PSD) of the system running autonomously ought to approximate $1/f$ where f is the horizontal frequency axis while the PSD of the perturbed system ought to have one or more distinct peaks. Experimental comparison did not align with this hypothesis as shown in figure 9. How can this be explained?

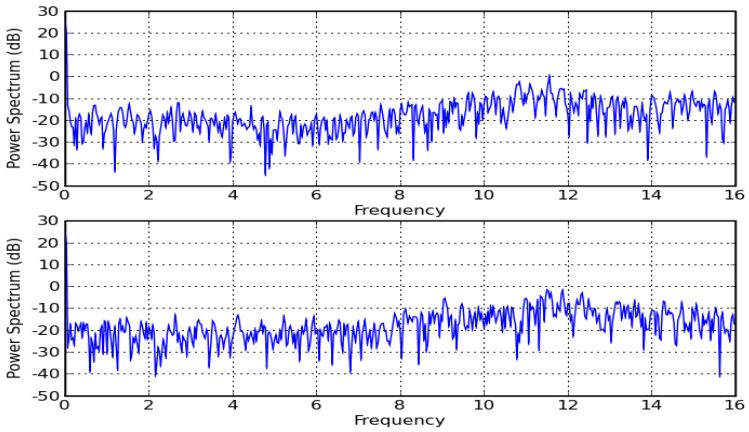

Figure 9: Comparison of the power spectral density functions (PSD) of the network with (above) and without (below) the 'hero' sequence of experiment 2. Neither have distinguished peaks of periodicity and both indicate chaotic behavior, invalidating experiment 2.

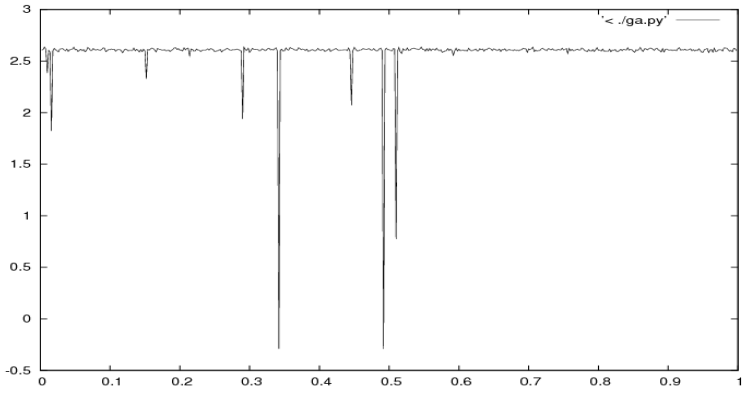

Figure 10: Lyapunov exponents of the network under perturbation by the hero sequence of experiment 2 but with varying initial seeds. Horizontal axis is the chosen initial seed and vertical axis is the maximal Lyapunov exponent. In the original run of the genetic algorithm, the seed was set to 0.34, which maps to a negative-valued valley that represents a numeric error.

In both experiments 1 and 2 discussed so far, approximation of the maximal Lyapunov exponent proceeded using a *fixed seed*. To see what effect this may have had, a test of the exponent approximation function was conducted on a sampling of initial seeds. The result is shown in figure 10 and shows that the algorithm stumbled on a numerical error in the approximation. This was resolved by measuring the Lyapunov exponents at 25 different random seeds and averaging the resultant values.

All further attempts to find a stabilizing input sequence using the above method failed.

6.2.2 Amplitude Coefficient

In all of the above experiments, the amplitude coefficient α was held at 0.00001. To investigate the effect this may have had, α was allowed to vary for simulation of the network under influence of the hero from experiment 2. Bifurcation diagram and Lyapunov exponents are shown in figures 11 and 12, respectively.

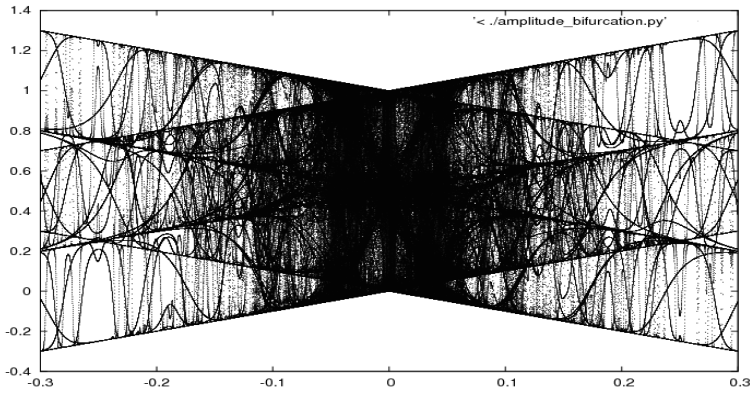

Figure 11: Bifurcation diagram of the network under the influence of the hero from experiment 2 where α is allowed to vary from -0.3 to 0.3. α is represented on the horizontal axis while the vertical axis is the approximate attractor for one of the neurons.

Both figures indicate the appearance and disappearance of orderly and chaotic windows as α varies away from 0 in both directions.

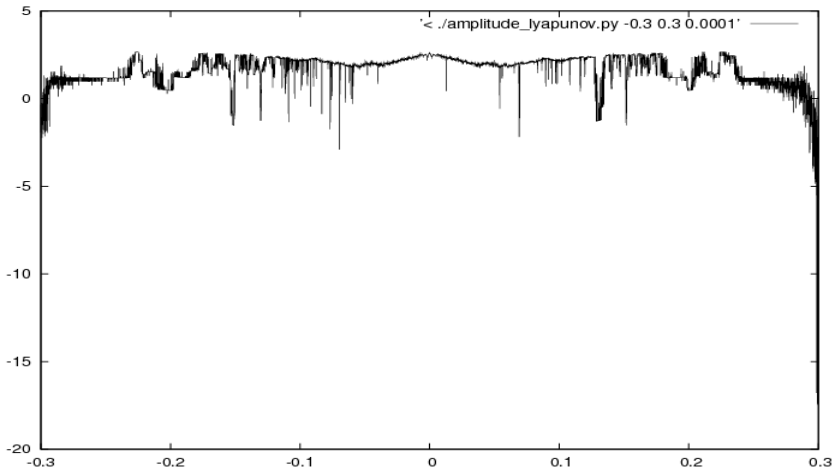

Figure 12: Lyapunov exponents of the network under perturbation by the hero sequence of experiment 2 but with varying α.

This gives us reason to let α vary in genetic simulations and to reformulate the task of the genetic algorithm as the combined minimization of both the maximal Lyapunov exponent and the parameter α. We can see from both figures 11 and 12 that with a sufficiently large value of α, the system can be forcibly stabilized, but ostensibly *any* sequence with a large enough α could accomplish such a task. Again, the gist is to apply as minimal an effect as possible while achieving the greatest possible stabilization.

This has been implemented and first experiments show difficulty achieving values of α below 0.15 which is unacceptably large.

The fitness function chosen was $f(s, \alpha) = tanh(L(s)) + tanh(10(\alpha - 0.12))$. The function was chosen to maximize the slope of the fitness surface around an area that appeared to be a 'barrier' to further evolutionary progress using other guesses at fitness functions. A more systematic comparison of fitness functions is in order.

6.2.3 Real-valued Sequences

A last set of preliminary experiments considered input sequences with real-valued elements not restricted to a binary alphabet. Between any two distinct binary strings exists a continuum of real-valued strings along which the bifurcation structure and Lyapunov surface varies

smoothly. Some of the most optimal α-sequence pairs found so far have α values approaching 0.05 and negative maximal Lyapunov exponents, apparently indicating that more optimal solutions exist in-between sequences with strictly binary elements.

6.3 Systematic Comparison

In preliminary experiments, choices of fitness function and crossover and selection operators were arbitrary. Without a strong theoretical understanding of the problem, all three GA components deserve a systematic comparison of their effects on run-time behavior and best solutions found.

In all, 4 fitness functions, 10 crossover operators, and 5 selection operators are devised. Of each combination, 4 trials were run. Each trial was terminated after 3500 generations and took about 1.2 hours to complete which sums to about 40 days of computing time.

6.3.1 Crossover operators

- "Random Single Point" Subsequences of each parent sequence are selected by a random midpoint and exchanged to constitute complementary children (Figure 13).
- "Fixed Single Point" Subsequences of each parent sequence are selected by a fixed midpoint and exchanged to constitute complementary children (Figure 14).
- "Gaussian Merge" Gaussian merge crossover. The value of each element of two parents are taken as standard deviations from a mean and used to construct a Gaussian (normal) distribution. A random value is drawn from the distribution. Its value is used as the value of the corresponding element in the first child's sequence and its complimentary value is used for the second child (Figure 15).
- "Binary Random Single Point", and "Binary Fixed Single Point" These two methods are identical to the random and fixed single point methods described above except that sequence elements are first converted to a binary representation and treated together as a then much longer sequence.
- "DCT Random Single Point", "DCT Fixed Single Point", "DCT Gaussian Merge", "DCT Binary Random Single Point", "DCT

Binary Fixed Single Point" These five operators employ the same methods described above except that sequences are first passed through the discrete cosine transform and placed in the frequency domain before crossover is applied. Children are then passed through the inverse discrete cosine transform to return them to the phase domain (Figure 16).

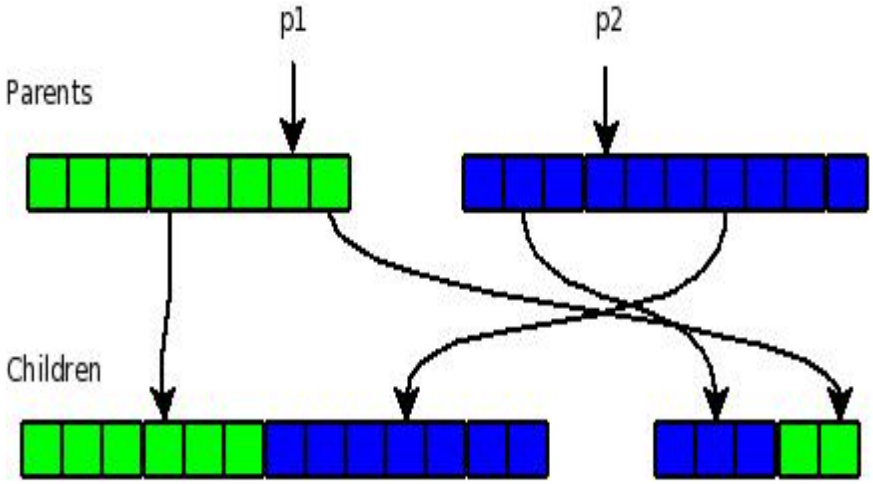

Figure 13: Random single point crossover. Subsequences of each parent sequence are selected by a random midpoint and exchanged to constitute complementary children.

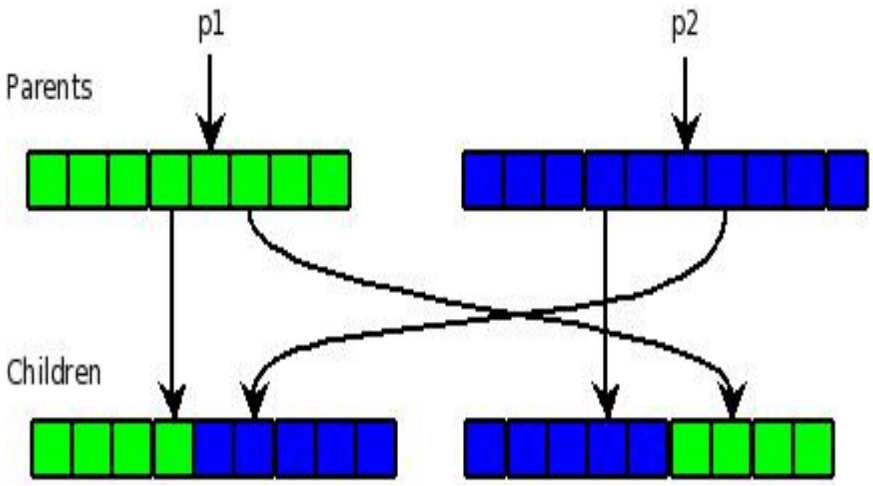

Figure 14: Fixed single point crossover. Subsequences of each parent sequence are selected by a fixed midpoint and exchanged to constitute complementary children.

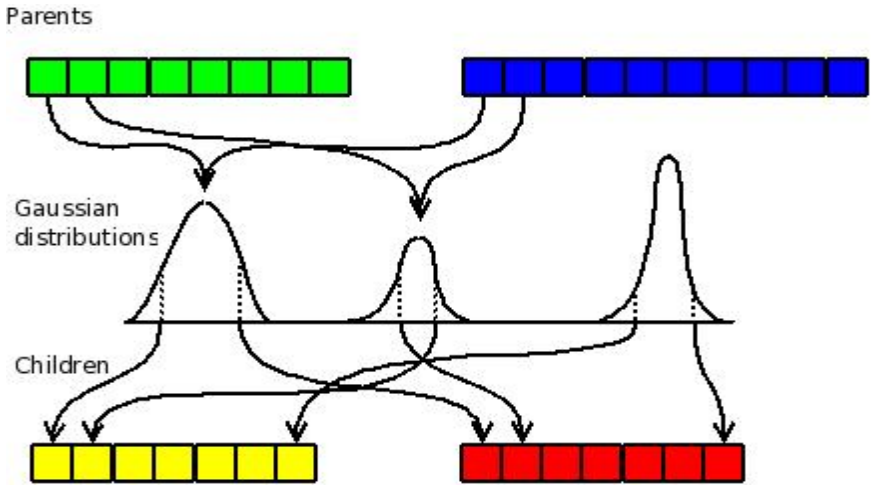

Figure 15: Gaussian merge crossover. The value of each element of two parents are taken as standard deviations from a mean and used to construct a Gaussian distribution. A random value is drawn from the distribution. Its value is used as the value of the corresponding element in the first child's sequence and its complimentary value is used for the second child.

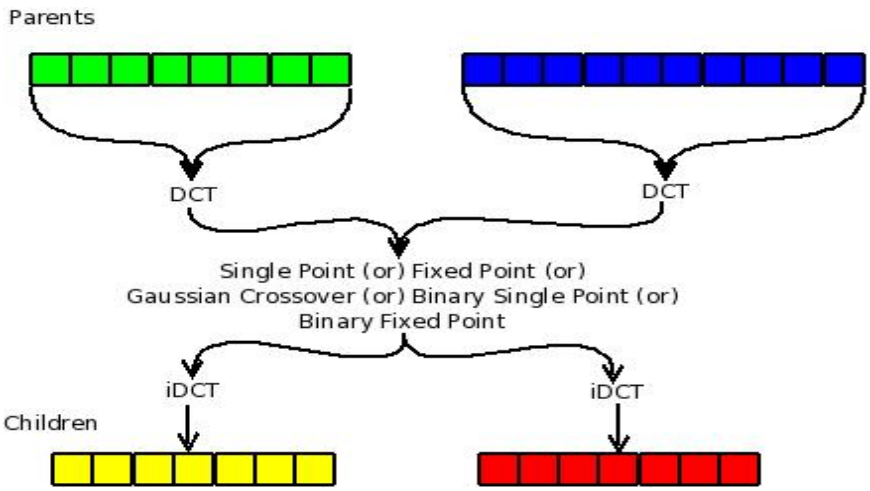

Figure 16: Discrete Cosine Transform operators. This concept is applied to all previously described crossover operators. Input sequences are transformed to the frequency domain, crossover is performed and the produced children are then inverted back into the phase domain.

6.3.2 Fitness functions

- "Tanh Combination Metric" (Figure 17)

$$f(\langle S, \alpha \rangle) = tanh(L(\langle S, \alpha \rangle)) + tanh(10 * (\alpha - 0.12))$$

- "Division Metric" (Figure 18)

$$f(\langle S, \alpha \rangle) = \frac{L(\langle S, \alpha \rangle)}{\alpha}$$

- "Addition Metric" (Figure 19)

$$f(\langle S, \alpha \rangle) = L(\langle S, \alpha \rangle) + \alpha$$

- "Piecewise Metric"

$$f(\langle S, \alpha \rangle) = (L > 0 ? L : \alpha)$$

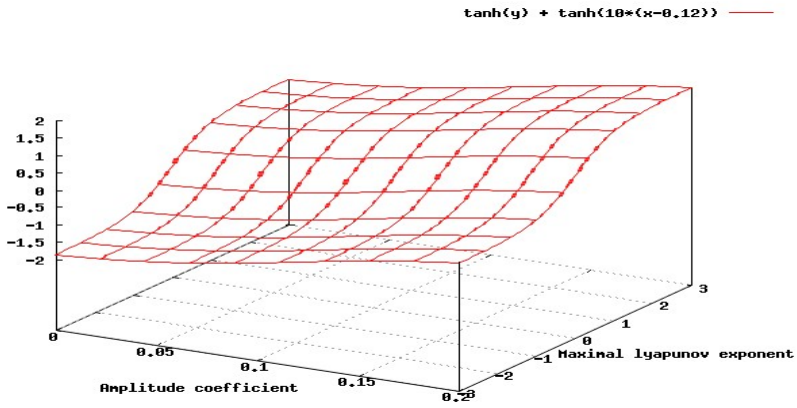

Figure 17: Surface of the "Tanh Combination Metric" fitness function where y represents the maximal Lyapunov number and x represents α

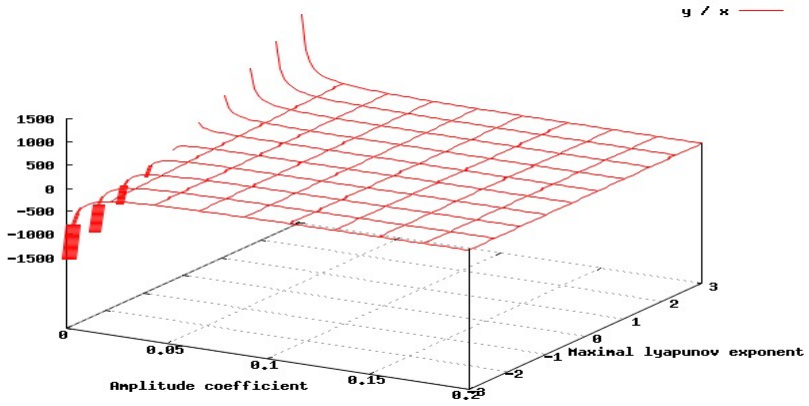

Figure 18: Surface of the "Division Metric" fitness function where y represents the maximal Lyapunov number and x represents α

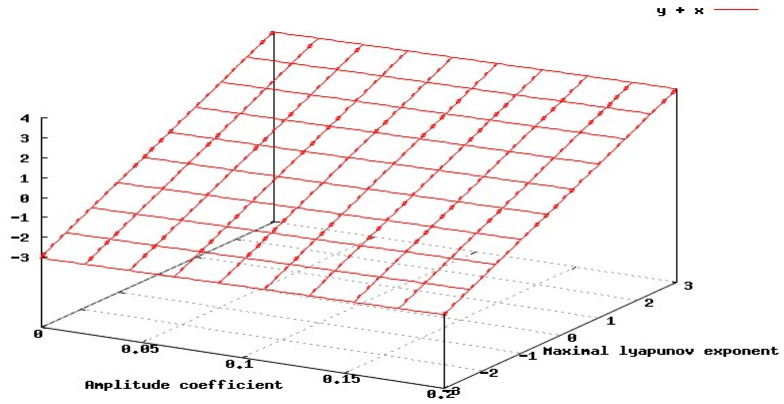

Figure 19: Surface of the "Addition Metric" fitness function where y represents the maximal Lyapunov number and x represents α

6.3.3 Selection operators

- "Non Greedy Selection" Four individuals are selected at random from the population. The best two breed to produce children and the worst two are replaced by the children (Figure 20).
- "Greedy Selection" Two individuals are selected at random for breeding. Their children replace the worst two individuals in the population (Figure 21).
- "Mutant Hero" No crossover occurs. The best individual in the population is selected and two mutant children are produced. They are inserted back in the population in place of the worst two individuals.
- "Greedy Mutant Hero" A combination of the above selection methods. Greedy selection is applied as above followed by an application of the mutant hero method.
- "Non Greedy Mutant Hero" A combination of the above selection methods. Non greedy selection is applied as above followed by an application of the mutant hero method.

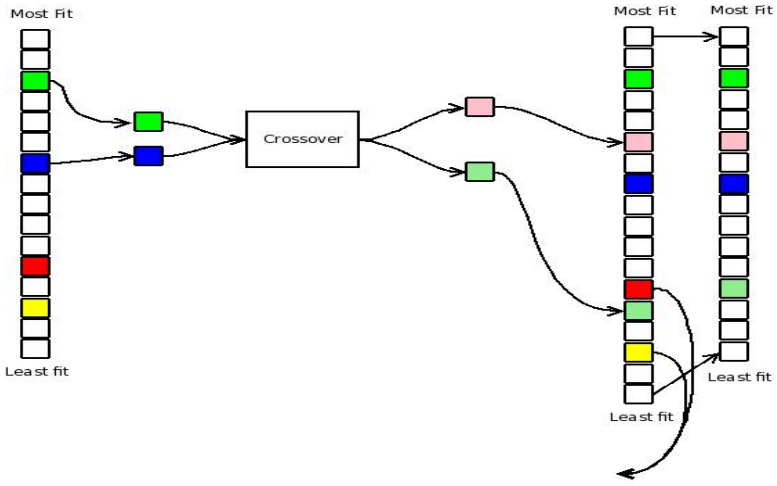

Figure 20: Illustration of the non greedy selection method.

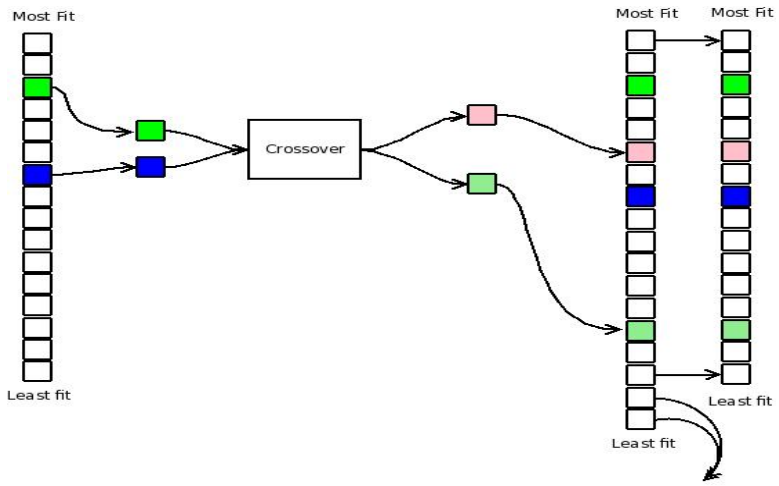

Figure 21: Illustration of the greedy selection method.

6.4 Sampling the Problem Space

Part of the motivation for such a robust comparison of genetic operators is a genuine lack of theory about the relationship between driving inputs network dynamics and without theory we cannot make intelligent choices.

The fitness landscape of the genetic algorithm formulated above is of quite high dimension. For input sequences of length n, a fitness landscape is a n-dimensional manifold immersed in a $n+1$-dimensional space. For $n > 2$ this cannot be visualized. To give some idea of what solutions the GA is searching through, a brute-force visualization of the Lyapunov surface is carried out for the small toy-class of inputs of period 2.

Images are generated by mapping points on the plane $p = (x, y)$ to input sequences $S_{x,y} = \{x, y, x, y, x, ...\}$, literally the concatenation of the pair over and over. A fitness map is generated by measuring the Lyapunov number of each sequence $S_{x,y}$ and plotting the Lyapunov number as an intensity value of the pixel associated with $p = (x, y)$.

7 Experimental Results

The fitness map of the toy class of period-2 inputs is quite complex and appears in figures 22, 23, 24, and 25. Its box-counting fractal dimension was measured at 1.914.

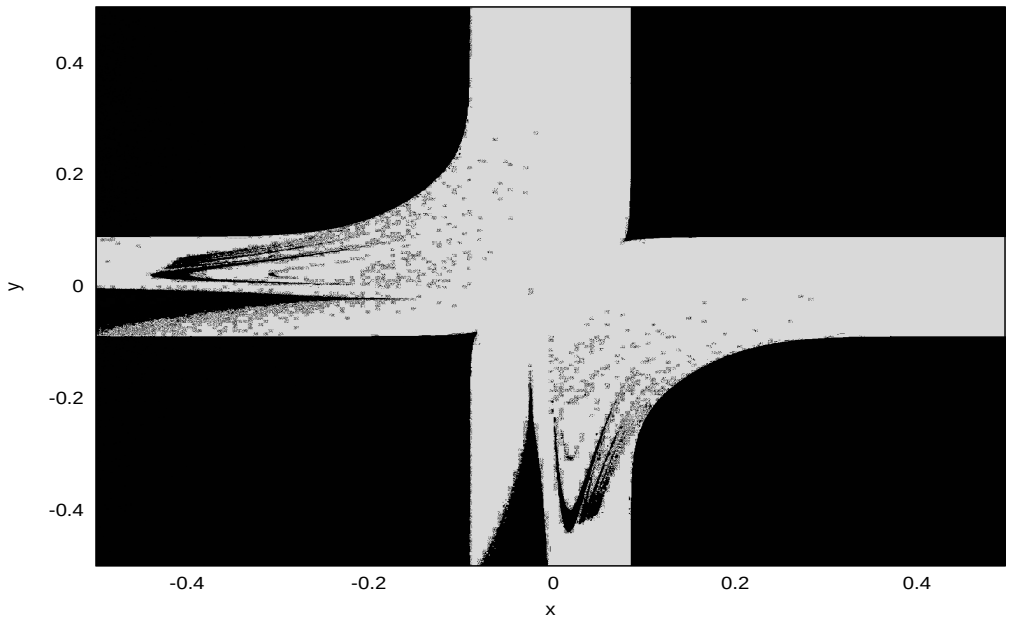

Figure 22: Lyapunov map for the toy class of period-2 solutions shown from $x \in [-0.5 : 0.5], y \in [-0.5 : 0.5]$. Dark indicates a negative maximal lyapunov exponent (stabilized) while light is positive (chaotic).

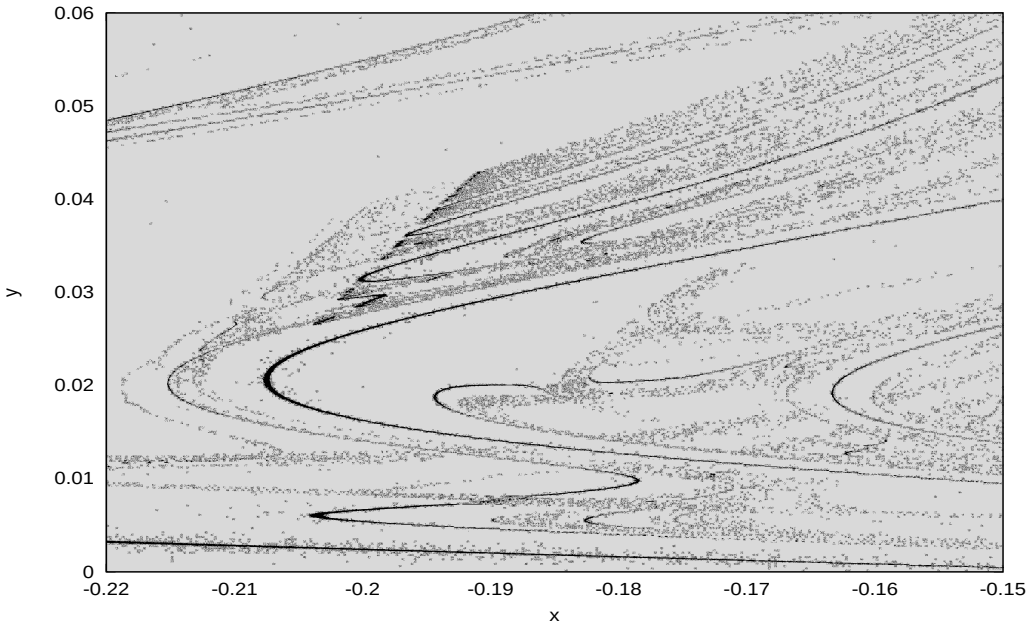

Figure 23: A subset of figure 22 enlarged to show detail at $x \in [-0.22 : -0.15], y \in [0 : 0.06]$. Dark indicates a negative maximal lyapunov exponent (stabilized) while light is positive (chaotic).

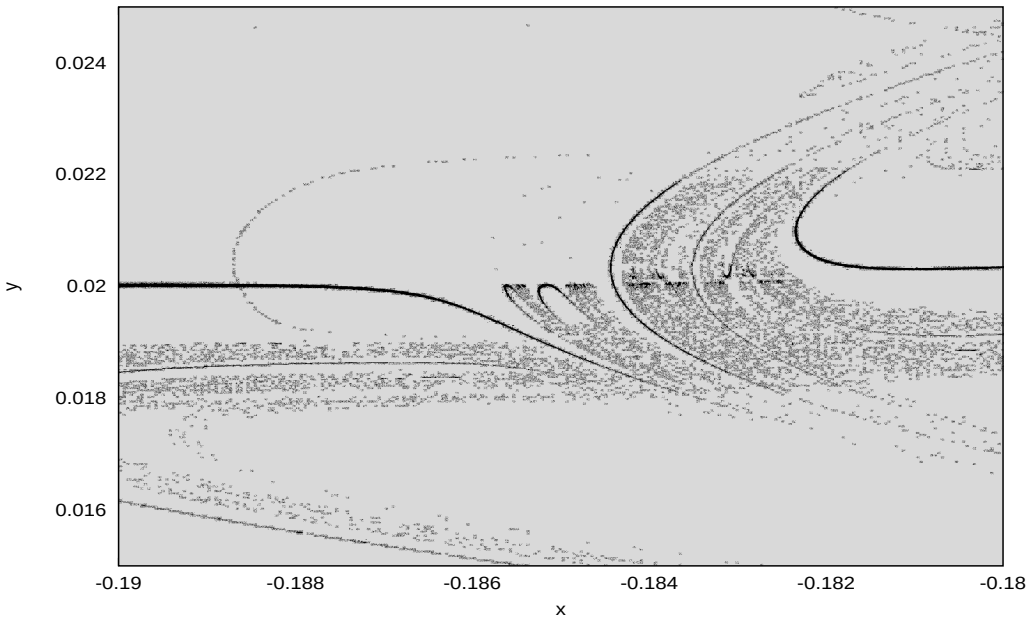

Figure 24: A subset of figure 23 enlarged to show detail at $x \in [-0.19 : -0.18], y \in [0.015 : 0.025]$. Dark indicates a negative maximal lyapunov exponent (stabilized) while light is positive (chaotic).

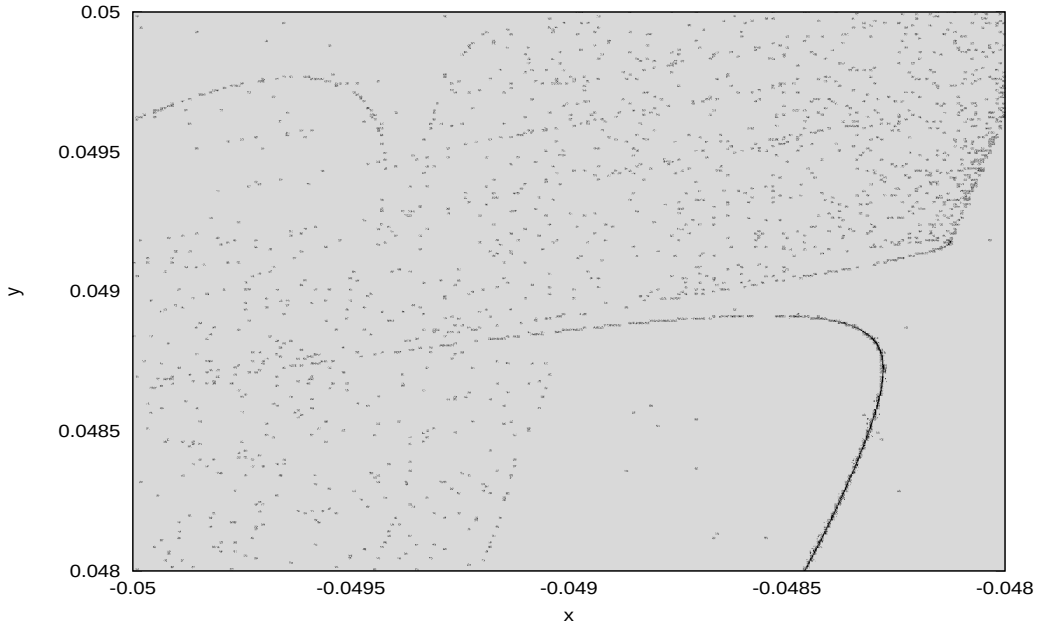

Figure 25: Another subset of figure 22 enlarged to show detail. Chosen to be close to the origin at $x \in [-0.05 : -0.048], y \in [0.048 : 0.5]$. Dark indicates a negative maximal lyapunov exponent (stabilized) while light is positive (chaotic).

The results of the systematic comparison are reported below. Almost all trials found solutions with negative (stabilizing) Lyapunov numbers. Those that did not were thrown out.

Table 1: Smallest and average α found organized by fitness function (200 experiments for each)

Metric	Minimal α	Average α
Tanh Combination	0.001078	0.199405
Division	0.021248	0.259698
Addition	0.045094	0.283053
Piecewise	0.021002	0.092758

Table 2: Smallest and average α found organized by selection method (200 experiments for each)

Selection Method	Minimal α	Average α
Non Greedy	0.002770	0.276603
Greedy	0.001078	0.221014
Greedy Mutant Hero	0.021655	0.184394
Non Greedy Mutant Hero	0.009510	0.150492
Mutant Hero	0.015576	0.211140

Table 3: Smallest and average α found organized by crossover operator (80 experiments for each)

Crossover Operator	Minimal α	Average α
Random Single Point	0.016584	0.173428
Fixed Single Point	0.024325	0.245391
Gaussian Merge	0.021655	0.222666
Binary Random Single Point	0.021248	0.184123
Binary Random Fixed Point	0.015576	0.213345
DCT Random Single Point	0.010714	0.174865
DCT Fixed Single Point	0.023137	0.253200
DCT Gaussian Merge	0.002770	0.184745
DCT Binary Random Single Point	0.001078	0.186835
DCT Binary Fixed Single Point	0.022892	0.248685

Following are the 10 solutions found out of all 800 trials with the overall smallest amplitude coefficients.

Table 4: Top 10 Trials (by smallest α)

α	Fitness	Crossover	Selection
0.001078	Tanh Combo	DCT Binary Random Point	Greedy
0.002770	Tanh Combo	DCT Gaussian Merge	Non Greedy
0.009510	Tanh Combo	DCT Binary Random Point	Non Greedy Mutant
0.010714	Tanh Combo	DCT Random Point	Non Greedy Mutant
0.015576	Tanh Combo	- -	Mutant Hero
0.016584	Tanh Combo	Random Point	Non Greedy Mutant
0.018824	Tanh Combo	DCT Random Point	Non Greedy Mutant
0.021002	Piecewise	- -	Mutant Hero
0.021248	Division	- -	Mutant Hero
0.021655	Tanh Combo	Gaussian Merge	Greedy Mutant

8 Discussion

We can see from tables 1 and 4 that introduction of the hyperbolic tangent fitness metric increased the probability of finding solutions with extraordinarily small amplitude coefficients. This metric was constructed during preliminary experiments to address what appeared to be 'invisible barriers' in the fitness space. Populations tended to either clump at small α but with $L > 0$ (unacceptable; still chaotic) or at $\alpha > 0.12$ with $L < 0$ (acceptably stabilized but with unacceptably large α). The metric was designed to maximize the slope of the fitness surface along areas where populations were stagnating. In retrospect, the slope of the fitness surface really plays no role at all as a selection pressure, but iso-fitness-curves do. It appears to be the case that the curves of equivalently-fit individuals through the fitness surface of the 'tanh combination metric' provide routes to small-α solutions not otherwise accessible.

Take note also that while the piecewise metric did not develop many outstanding solutions, among fitness metrics it produced the best solutions on average. It was designed with much the same intent as the tanh combination metric (to maximize the gradient in the direction most desired) but did not produce any optimal solutions.

Results of the selection methods in Table 2 fail to demonstrate a clearly superior method. All five selection methods appear in the Top 10 table. Greedy and Non Greedy standard selection eke ahead with the top best two solutions found but comparison of their mean α values found with those methods that take advantage of aggressive mutation demonstrates an inconsistency. It is worth pointing out that the mutant hero method with *no* crossover is outperformed by its peer methods.

The performance of crossover methods is a bit more difficult to take in simply because so many were implemented. All methods that involved the Fixed Single Point process performed poorly. This is perhaps due to an imposed rigidity of selective pressure that did not conform to the problem. The Gaussian Merge method performed surprisingly poorly in isolation but stands out as one of the best methods when composed with the discrete cosine transformation. As a group, methods that operated on binary representations of input strings did not perform well, the exception being the DCT Binary Random Single Point method. The only logical grouping of operators that performed well, as a group, were methods that operated in the frequency domain,

methods under composition with the discrete cosine transformation.

9 Conclusion

We can now answer positively to the central question posed in this thesis, but the problems approached here raise more questions. We have found concretely that there do exist finite length sequences of quite small amplitude that can, under repetition as driving inputs, stabilize an otherwise chaotic neural network. Furthermore, subjective inspection of the brute-force sampling of the toy class indicates that arbitrarily small sequences are included in the set of stabilizing inputs (Figure 26).

With regard to the application of genetic algorithms to stabilizing neural networks, performing crossover in the frequency domain appears to have a positive effect on algorithm convergence (Table 3).

Some open questions for future research: does there exist a lower bound on the amplitude coefficient of stabilizing sequences? We have here found sequences with amplitude on the order of 10^{-3} but yet smaller patterns may exist.

While composition with discrete cosine transform in crossover provided quite a boost to algorithm performance, the choice of transform was somewhat arbitrary. Does there exist a transform other than DCT that reduces the complexity of the fitness space. Future research might be directed at the development of a genetic algorithm that searches through the space of transforms for ones that minimize the complexity of the fractal in figure 22.

Other research might include consideration of networks that have been shown to be useful in computational tasks; this as opposed to the toy network studied here. What is the relationship between their usefulness in reservoir computing techniques and their sets of stabilizing inputs? Can we classify chaotic networks in terms of their stabilizing sets? Solutions to these questions are beyond the scope of this document but in the author's estimation point the way toward more fruitful applications of recurrent neural networks in the pursuit of both scientific mastery and the unlocking of nature's mystery.

References

[1] K. Alligood, T. Sauer, J. A. Yorke *Chaos: An Introduction to Dynamical Systems*, Springer-Verlag, 1997.

[2] B. R. Andrievskii, A. L. Fradkov, *Control of Chaos: Methods and Applications*, Automation and Remote Control, vol. 64, pgs 673-713, 2003.

[3] D. Arrowsmith and C. M. Place, *An Introduction to Dynamical Systems*, Cambridge University Press, 1990.

[4] Balzs Csand Csji, *Approximation with Artificial Neural Networks*, Faculty of Sciences, Etvs Lornd University, Hungary, 2001.

[5] J. Clegg, J. A. Walker, and J. F. Miller, *A new crossover technique for Cartesian genetic programming*, Proceedings of the 9th annual conference on Genetic and evolutionary computation, 1580-1587, 2007.

[6] K. Doya, *Bifurcations in the Learning of Recurrent Neural Networks*, Proceedings of the 1992 IEEE International Symposium on Circuits and Systems, 2777-2780, 1992.

[7] R. L. Devaney, *An Introduction to Chaotic Dynamical Systems*, Westview Press, 2003.

[8] A. Weigend and N. Gershenfeld, *Time Series Prediction. Forecasting the Future and Understanding the Past*, Addison-Wesley, 1994.

[9] D. Aubin and A. Dahan Dalmedico, *Writing the History of Dynamical Systems and Chaos: Longue Durée and Revolution, Disciplines, and Cultures*, Historia Mathematica, vol. 29, pgs 273-339, 2002.

[10] P. Dayan and L. F. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*, The MIT Press, 2001.

[11] C. Rocsoreanu, A. Georgescu, and N. Giurgiteanu, *The Fitzhugh-nagumo model: Bifurcations and Dynamics*, Kluwer, 2000.

[12] M. W. Hirsch, S. Smale, R. L. Devaney, *Differential Equations, Dynamical System, and an Introduction to Chaos*, Elsevier, 2004.

[13] E. Izhikevich, *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting*, The MIT Press, 2007.

[14] R. Legenstein and W. Maass, *What makes a dynamical system computationally powerful?* Haykin, Principe, Sejnowski, and McWhirter: New Directions in Statistical Signal Processing: From Systems to Brain 2005.

[15] T. Natschläger, W. Maass, H. Markram, *The "Liquid Computer": A Novel Strategy for Real-Time Computing on Time Series* Special Issue on Foundations of Information Processing of TELEMATIK, vol. 8, num. 1, p. 39-43, 2002

[16] W. Maass and C. M. Bishop, *Pulsed Neural Networks*, The MIT Press, 1999.

[17] W. Maass, T. Natschläger, and H. Markram, *A Model for Real-Time Computation in Generic Neural Microcircuits*, Proc. of NIPS 2002, Advances in Neural Information Processing Systems, volume 15, pages 229-236. MIT Press, 2003.

[18] S. Nikolov, and B. Bozhkov, *Bifurcations and chaotic behavior on the Lanford system*, Chaos, Solitons and Fractals vol. 21, pgs 803-808, 2004.

[19] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 1995.

[20] R. V. Sole, L. Menendez, D. L. Prida, *Controlling Chaos in Discrete Neural Networks*, Phys Lett A, vol. 199, pgs 65-69, 1995.

[21] J. C. Sprott, *Chaos and Time Series Analysis*, Oxford University Press, 2003.

[22] H. Jaeger, *The "echo state" approach to analysing and training recurrent neural networks.* GMD Report 148, GMD - German National Research Institute for Computer Science, 2001.

[23] D. Verstraeten, B. Scharuwen, M. D'Haene, D. Stroobandt, *An experimental unification of reservoir computing methods*, Neural Networks, vol. 20, pgs 391-403, 2007.

[24] H. Yu, Y. Liu, J. Peng, *Control of chaotic neural networks based on contraction mappings*, Chaos, Solitons, and Fractals, vol. 22, pgs 787-792, 2004.