1-1-2011

On the Effect of Topology on Learning and Generalization in Random Automata Networks

Alireza Goudarzi
Portland State University

10.15760/etd.193

On the Effect of Topology on Learning and Generalization in Random Automata Networks

by

Alireza Goudarzi

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science

Portland State University
© 2011

ABSTRACT

We extend the study of learning and generalization in feedforward Boolean networks [70, 93] to random Boolean networks (RBNs). We explore the relationship between the learning capability and the network topology, the system size, the training sample size, and the complexity of the computational tasks. We show experimentally that there exists a critical connectivity K_c that improves the generalization and adaptation in networks. In addition, we show that in finite size networks, the critical K is a power-law function of the system size N and the fraction of inputs used during the training. We explain why adaptation improves at this critical connectivity by showing that the network ensemble manifests maximal topological diversity near K_c. Our work is partly motivated by self-assembled molecular and nanoscale electronics. Our findings allow to determine an automata network topology class for efficient and robust information processing.

ACKNOWLEDGMENTS

I am indebted to my advisor Christof Teuscher for his support and guidance without which none of this would have been possible. The author is also grateful for invaluable contributions from Natali Gulbahce and Thimo Rolf. Many indispensable discussions with Lukas Svec and Guy Feldman have helped to shape this work. This study was supported in part by the National Science Foundation under grant number 1028120.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1

INTRODUCTION

The goal of this thesis is to study find optimal topologies for unstructured networks of random automata performing robust information processing in noisy environment. We extend our understanding of the task solving in feedforward random networks of 2-input Boolean gates to random networks of Boolean gates with various number inputs. We study the learning capability of these systems in a noise free environment and calculate their intrinsic functional capacity. We then study the behavior of these random networks in noisy environment and explain the evolution of their connectivity and other topological properties. Our result is applicable in application-specific hardware systems and future self-assembled nanoelectronics.

1.1 GOAL AND MOTIVATION

We introduce a general purpose network information processing system based on Random Boolean Networks (RBN) [40]. Our long term aim is to gather viable heuristics that we can use to develop a self-organizing algorithm to configure this network for a specific computational task. This algorithm should be robust both during the assembly process and during the operation of the network. Teuscher *et al.* [86–88] introduced RBNs as a candidate solution to build the future nanoelectronic architectures.

For practical purposes, our conceptual model of computing today is based on the von Neumann architecture in which data and instructions are stored in memory and processed by a processing unit. The explicit sequential nature of the processing is a major drawback in speed of computation. The traditional way of building processors employs integrating well-designed transistor-based circuits that are built using semiconductor technology. To improve the performance of this type of processor, we have to build circuits with higher integration, i.e., pack more transistors in less space. For many years this has been the standard way for the microprocessor industry to build faster processors. In the past decade, as the level of integration has dropped deep into the nanometer scale, the fabrication of processors has faced new challenges.

The nanoscale fabrication of microprocessors introduced higher energy consumption, risk of failure because of high operational temperature, and low yield [34]. A common solution around the performance limitation of a single processor is concurrency. We can distribute the computational load of a program between many processors to achieve a speed-up. But this solution brings many challenges along with it. Asanovic *et al.* [9] gives an overview of different techniques that are proposed for distributed concurrent programming and the drawbacks that each technique faces. However, in the perspective of the author, the major limitation of all parallel programming techniques today is scalability. Except for a few "embarrassingly parallel" computations — such as matrix multiplication — we do not have a general way of achieving speed-up for a system with more than eight processors; the cost of inter-processor communication to guarantee memory consistency is higher than we can afford. However, biological organisms appear to have evolved to solve the distributed parallel processing problem effectively. We seem to be immersed in concurrent operations all around us. From the laws of physics that govern the interactions between the objects of various scales in our universe to the biology of living organisms, we have observed, studied and modeled these systems for centuries. Yet we are unable to harness the power of concurrency in the architectures we so carefully design and build specifically to achieve it. Both the founder of theoretical computer science, Turing, and the creator of the stored program architecture, von Neumann, had envisioned the limitations that the sequential nature of their designs entailed. Their fascination with the way biological organisms and our brain work eventually led to the creation of Turing's "unorganized machine" [85, 90] and von Neumann's "self-reproducing automata" [97]. Unfortunately both of these ideas remained mainly in the world of academia. However, challenges we face in the world of computer architecture and advances in nanotechnology has produced a new wave of research in the theory and the application of simple automata networks, an underpinning element of both the unorganized machine and the cellular automata.

1.2 CHALLENGES

Experimental physicists and material scientists have devised practical ways to self-assemble nanoscale wires and switches very cheaply in comparison to conventional electronics fabrication [34]. However, one issue with using these self-assembled circuits is the lack of knowledge about their final structure. All the methods of programming a general purpose computer today rely on the knowledge of the underlying system architecture, i.e., the memory structure, the input-output ports, etc. The high level questions we attempt to study in this thesis are:

1. How would we create a general purpose programming system to configure computers with a random or unstructured architecture?
2. What is the optimal structure of a reprogrammable computer based on unstructured devices?

This study aims to create a foundation for creating self-organization algorithms for a random network of random automata. Despite the similarities between our proposed model of automata networks with various classical neural networks, we cannot use traditional learning algorithms to train generic random networks for a computational task. Learning in classical neural nets often rely on the assumptions about the processing elements of the network and the full connectivity between elements [35]. It is through these assumptions that convergence theorems could be proven. However, we do not like to make any assumption for the processing elements and their connectivity to reason about their programming. This allows our result to be general and independent of underlying technology.

1.3 CONTRIBUTIONS

Our contributions in this work are as follows:

1. We proposed an augmented RBN model to perform computational tasks with inputs and outputs.
2. We developed necessary software frameworks to use RBNs and feedforward networks in a task solving context (see Section 3.5).
3. We integrated our C++ RBN framework into the ParadisEO [17] framework for evolutionary computation.
4. We reproduced the learning capability results for feedforward random networks with exact connectivity of 2 (see Section **??**).
5. We extended the learning capability theory to RBNs with exact and mean connectivities (see Sections **??** and **??**).
6. We calculated phase volumes for RBNs (see Section 3.3.2).
7. We calculated the functional entropy of the RBNs as a function of $\langle K \rangle$ (see Section 3.3.2).

8. We introduced the cumulative learning probability, the cumulative perfect training likelihood, the cumulative generalization score, and the cumulative training score to study the performance of task solving in RBNs with fixed $\langle K \rangle$ evolution, independently from training sample size (see Sections **??** and **??**).

9. We published the first results from the learning probability of the RBNs in [88].

10. We published the results for RBN learning probability with various $\langle K \rangle$ in [33].

11. We studied and established optimal connectivity K_c in RBNs for information processing (see Section 4.1).

12. We showed that computation and robustness are optimal in RBNs with critical connectivity K_c (see Section 4.1.6).

13. We showed that K_c scales as a power-law of the system size N (see Section 4.1.3).

14. We showed that the optimal critical connectivity K_c for robust computation corresponds to the "edge of stability" (see Sections 4.1.7 and 4.1.8).

15. We showed that the degree distribution of RBNs changes from a Poissonian distribution to an exponential distribution (see Section 4.2).

16. We explained the evolution of exponential degree distribution using the maximization of entropy during the Evolutionary Steady State (ESS) (see Section 4.2.7).

17. We showed that the population at K_c has maximum fitness diversity and, using Fisher's fundamental theory of natural selection, we explained why this connectivity ensures optimal robustness and computation (see Section 4.2.7).

18. We studied the behavior of graph topological measures for Erdös-Rényi random graphs, exponential random graphs, and evolved RBNs as a function of $\langle K \rangle$ (see Section 4.3.2).

19. We showed that the graph topological measures show maximum variance near K_c and postulated this to be a possible source of diversity in the fitness of the networks (see Sections 4.3.2 and 4.3.3).

2

RELATED WORK

2.1 ARTIFICIAL NEURAL NETWORKS

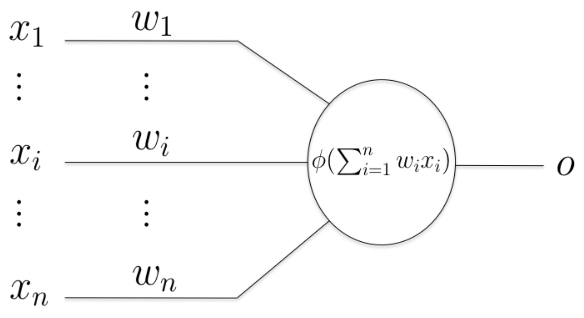
Artificial Neural Networks (ANN) are forms of parallel distributed models of computation that are inspired by the structure of the brain. ANN research started with the first synthesis of neuron model by the pioneering work of McCulloch and Pitts [59]. Rosenblatt [77] incorporated this model into his *perceptron* (Figure 2.1). Given an input vector of n elements and the corresponding set of presynaptic weights $\{w_i, 1 \leq i \leq n\}$, the output of the perceptron is calculated as follows:

$$o = \begin{cases} 1, & if \sum_{i=1}^{n} w_i x_i + b > 0 \\ 0, & otherwise \end{cases} \tag{2.1}$$

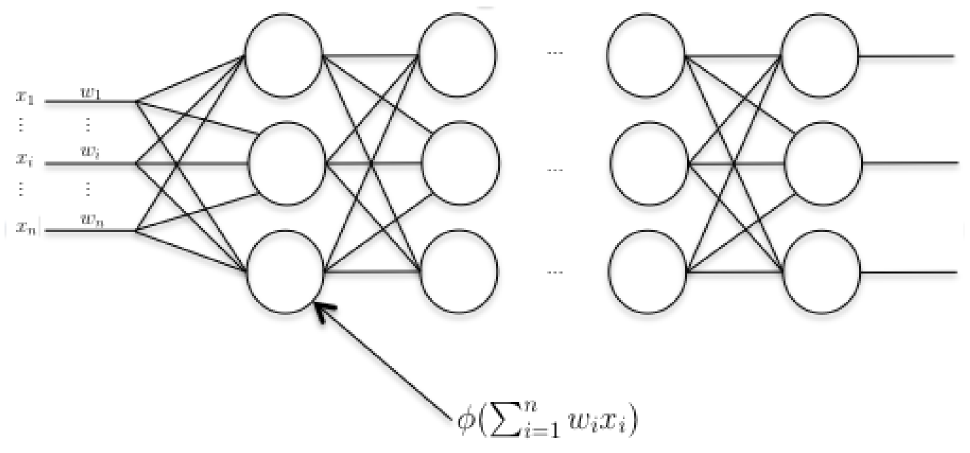
Here, each x_i is one of n inputs with their corresponding weight w_i. b is an inductive bias which adjusts the firing threshold of the perceptron.

Later, Widrow [98] proposed the Madaline rule to train networks of mutliple adaptive elements. The hallmark of ANN developement was Rumelhart's *et al.* [79] error backpropagation algorithm. This algorithm extends the Madaline rule and the steepest descent optimization to multi-layer ANNs.

The introduction of the McCulloch-Pitts neuron independently coincided with Alan Turing's proposal for an "unorganized machine" [85, 90]. The "unorganized machine" consists of networks of logical NAND gates with two inputs. Each input is connected to a randomly selected node in the network. The links between the nodes in this network have a switch that can be turned on or off to close or open



(a) Perceptron



(b) Madaline

Figure 2.1: Architecture of a single layer perceptron Artificial Neural Network (ANN).

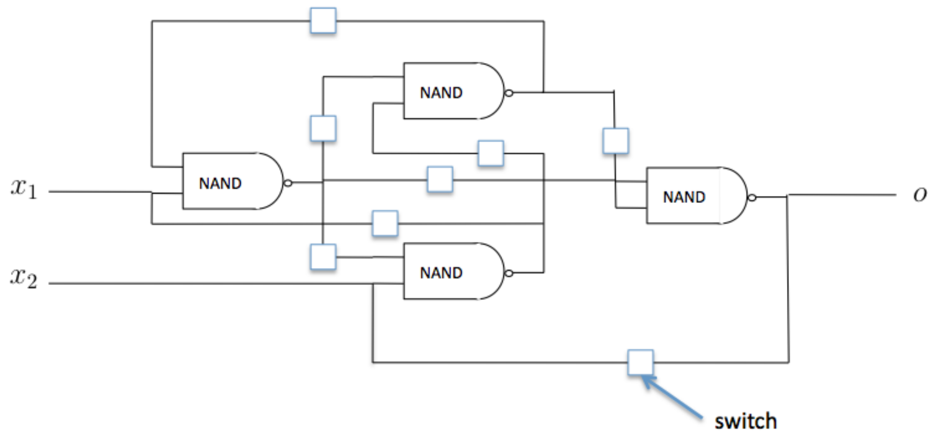


Figure 2.2: Turing's B-type unorganized machine made up of randomly connected NAND gates with switches that may turn the links on or off.

the link. This architecture is called a B-type network. Turing proposed that by configuring the switches on the links, we can configure the network to perform any computation. Figure 2.2 illustrates the Turing B-type network. Turing's model is of particular importance for this thesis because despite its simplicity, it lays out a first computing model using unstructured circuits. Aleksander, Martland, and many others used random logical networks to implement various computations, such as pattern classification or associative pattern storage and retrieval [2–4, 56–58].

2.2 HOPFIELD MODEL AND SPARSE NETWORKS

In 1982, J. J. Hopfield proposed a biologically plausible ANN model for storing and retrieving information [37]. The importance of this model is due to the presence of feedback connections in the network. The model therefore constitues a dynamical system with different dynamical regimes and attractors; hence this model is sometimes called the attractor network. The Hopfield model consists of N identical threshold neurons. Each neuron i receives a link from every other neuron. The coupling weight J_{ij} characterizes the coupling strenght from the neuron j to the neuron i. We identify the state of *i-th* neuron with σ_i and it can assume the values "0" or "1." The neurons update their state asynchronously according to:

$$\sigma_i(t+1) = \sum_{1<j<N} J_{ij}\sigma_j(t). \tag{2.2}$$

This model is capable of storing a pattern $\xi = \sigma_1\sigma_2\ldots\sigma_s\ldots\sigma_N$ if the connection weights between the neurons follow:

$$J_{ij} = \begin{cases} \sum_{s=1}^{N}(2\sigma_i^s - 1)(2\sigma_j^s - 1) & i \neq j \\ 0 & i = j \end{cases} \tag{2.3}$$

After setting the weights, the network may retrieve the complete pattern ξ from any partial pattern ξ' set as initial condition. A later paper [38] extends this results to multi-valued neurons. Amit *et al.* [5] studied the thermodynamics of the Hopfield model using the spin-glass formalism. Amit *et al.* [6] and Gardner [29] showed the Hopfield model may store an infinite number of patterns p if $N \to \infty$ due to:

$$p = \alpha N. \tag{2.4}$$

α is the signal-to-noise ratio that defines how much distortion is acceptable in the retrieved patterns [48]. For binary synaptic weights, the optimal capacity is at $\alpha_c = 0.83$. Near saturation, i.e., for low values of α, the system has a spin-glass phase and a ferromagnetic phase with $2p$ stable states. The latter shows at least 98.5% accuracy of retrieved patterns for $\alpha < \alpha_c = 0.14$ [8]. However, these results hold for uncorrelated patterns ξ with arbitrary activity level (fraction of firing neurons in the pattern). For low activation patterns due to finite correlation, the number of retrievable patterns from the network will be $p < 1 + a^{-2}$ [7]. Here a is the activity of the pattern. Gardner and Derrida [31] calculated the optimal number of storable patterns in the network with the maximum retrieval error of ϵ_c. Fontanari [26] showed how to extend the simple pattern storage and retrieval of the Hopfield model to learning and generalization. In this approach, imprinting partial or noisy examples of a pattern to the network makes the network deduce the pattern itself. The generalization error in this method approaches the retrieval error of the standard Hopfield model, i.e., $\epsilon = 0.0165$, in the limit of a large number of training samples.

Canning and Gardner [18] show that a partially connected Hopfield model has better information storage capacity per connection. The size of the basin of attraction for the optimal network capacity was calculated in [30]. In addition, it has also been shown through simulation that the storage capacity of Hopfield networks becomes maximal in a small-world [1] topology with long-range connections of probability $p = 0.1$ [13, 49].

2.3 LEARNING THEORY

Learning in ANNs includes two broad categories: *supervised* and *unsupervised* learning. The difference between the two is how the ANN treats the training data. Learning is also closely related to the concept of generalization. Generalization characterizes the behavior of the network in response to novel input data. The relationship between learning, generalization, error, and the complexity of the ANN has been subject of much research as part of a unifying theme referred to as Vapnik-Chervonenkis (VC) dimension [12, 91, 94, 95]. In this section, we will explore in more depth the different learning types and the VC dimension theory.

In supervised learning, the training examples are organized into input patterns and corresponding expected output patterns from the network. The training process consist of feeding the inputs to the ANN and calculating the error between the ANN output and the expected output. The parameters (weights or functions) of the ANN are then tuned to minimize the error of the network. All gradient descent and back-propagation-based training algorithms fall within the class of supervised learning.

Unsupervised learning, on the other hand, does not comprise the notion of input and output. The data is fed into the network and the network adapts to represent the features of the data in some sense. For example, the Hopfield learning rule is a type of unsupervised learning in which the weights of the networks evolve to represent the aggregate local field of all the stored patterns in the network.

In his seminal work "A theory of learnable" [91], Valiant directly studied the phenomenology of *learning* in its generic form. In his words "[. . .] a program for performing a task has been acquired by learning if it has been acquired by any means other than explicit programming [91]." Valiant shows that it is possible to design learning machines that can learn whole classes of nontrivial general purpose concepts in polynomial time. This learning machine consists of a *learning protocol* and a *deduction procedure*. The learning protocol specifies how to obtain information from the outside world. The deduction procedure is the method to deduce the a correct recognition algorithm for the concepts to be learnt. In this context "[. . .] concept Q has been learnt if a program for recognizing it has been deduced. . . [91]." Closely related to the issue of learning a concept is generalization. Generalization is a measure of how well the concept is learnt. The learning machine deduces the concept using a few example patterns. After the learning process is completed, if the machine can associate novel patterns to the learnt concept correctly, then we say the machine can generalize well. Learning and generalization has, in turn, become the method of comparing various learning algorithms.

Vapnik [94] achieved the next crucial milestone in the theory of learning and generalization. Let us first define a prerequisite quantity, the Vapnik-Chervonenkis (VC) dimension of a learning machine. Here we refer to this as d_{VC}. Consider a set of p patterns. There are 2^p possible classifications over these patterns (assuming binary classification). That is, there are 2^p different ways to assign either of the classes "0" or "1" to each of the p patterns. If a classifier C can implement all of these classifications then the VC theory states that C *shatters* the space of patterns. Moreover, there exist a critical number of patterns $p \geq d_{VC}$ for which not a single set of p patterns can be completely classified by C. In this case the VC dimension of C is d_{VC}. d_{VC} defines the capacity of a learning machine in purely statistical terms. The Vapnik-Chervonenkis theorem (in combination with Sauer's lemma [80]) may express the bound for the probability of the maximum difference between test and generalization error of the class C as a function of d_{VC} due to [92, 96]:

$$P(max_{c \in C}|E_c(p) - \epsilon_c| > \mu) \leq e^{d_{VC}[ln(2e\alpha) - \mu^2\alpha]}. \tag{2.5}$$

Here, $E_c(p)$ is the error of the classifier $c \in C$ on p different test patterns, ϵ_c is the generalization error of the c, and $\alpha = \frac{p}{d_{VC}}$. In the limit of $d_{VC} \to \infty$, the r.h.s of this equation approaches zero for $ln(e\alpha) < \mu^2\alpha$ [69]. Hence, we will have a critical accuracy:

$$\mu_c = \sqrt{\frac{ln(e\alpha)}{\alpha}} \tag{2.6}$$

for which

$$P(max_{c \in C}|E_c(p) - \epsilon_c| \leq \mu_c) = 1. \tag{2.7}$$

2.4 CELLULAR AUTOMATA

In the late 40's, John von Neumann [97] introduced Cellular Automata (CA) as a biologically inspired model of discrete-state discrete-time dynamical system. The simplest type of CA or Elementary CA (ECA) [39] consists of a one dimensional lattice of cells, each with a self-connection and two connections to its immediate right and left neighbors. Each cell may assume either of the two states "0" and "1." All cells change their states at the same time according to a binary function called the CA rule. The dynamical properties of this system depends on the initial configuration of the lattice and the CA rule.

Stephen Wolfram [100] pioneered the investigation in local and global dynamics of the ECA. He classified the space of CA rules into four classes. In finite time, CA rule classes I and II will end in a fixed-point or cyclic attractors from almost any initial configuration. Class III rules are very sensitive to initial configuration and will lead to the strange (chaotic) attractors. IV rules show complex dynamics and one of the rules, i.e., 110, has been proven to be computationally universal.

Li and Packard [52] studied the structure of the ECA rule space. They organized the 256 possible rules into 88 equivalent classes according to the internal symmetries in the rules. Based on the dynamics of the CA, they assigned each equivalent class to five different dynamical classes of (1) null dynamics, (2) fixed-point, (3) periodic, (4) locally chaotic, and (5) chaotic. This assignment is not one-to-one hence the calculation of intra and inter-class transition in dynamics follows [53].

Wootters and Langton [101] studied the sharp transition in the dynamics of ECA rule space as a function of λ (homogeneity) of the CA rules. The phase transition becomes sharp in the limit of infinite-valued CA. Increasing the number of local connections pushes this transition towards $\lambda = 0$ which suggests that the at infinite range neighborhood the transition vanishes. Langton [50] suggested that complex computation in ECA occurs at the region of the rule space that corresponds to the phase transition in λ. This region is called "the edge of chaos." However, Mitchell *et al.* [62, 63] refuted the edge-of-chaos-computation argument by showing that it is possible to find CA rules in other regions of the rule space that perform complex computation.

The first method for the automatic design of CAs for parallel computation was proposed in [61]. This approach used genetic algorithms to evolve CA rules that could solve the density classification task. However, since this task requires global information processing it does not suit ECA dynamics since information transfer in ECA is limited. For finite size CAs, perfect solutions, i.e., CA rules that solve the task from all initial configuration of the CA, were not found. Analytical derivation of the CA rules and the initial configuration for the ECA to perform a desired configuration remains an open problem.

Mesot and Teuscher [60] showed that Random Boolean Networks (RBN) outperform 1-D CA (with neighborhood as large as 7) in density classification. Moreover, they derived an analytical method to deduce the local rules in the network to perform the task. In this experiment, RBN also performs better than small-world CA [89]. This is not so surprising because RBNs have a “global” view of the system’s state.

2.5 RANDOM BOOLEAN NETWORKS

Stuart Kauffman [40] introduced RBNs as a biologically viable model for gene regulatory networks. RBNs may be thought of as a generalized CA in which each cell—out of the total of N—is connected in random to K other cells; this is also called the NK model. Kauffman himself studied many theoretical aspects of RBNs and their applications in the context of biology [41–47, 83, 84].

Kauffman classified the dynamics of RBNs into three classes depending on the values of $\langle K \rangle$. For $\langle K \rangle < 2$, the networks are likely to find a fixed point or periodic attractor quickly. Networks of $\langle K \rangle > 2$ will have the chaotic dynamics in which the networks will not find an attractor in finite time, or the the attractor will not have a finite size. At $\langle K \rangle = 2$, the dynamical regimes of the networks show the maximal variance [47], this dynamical behavior called the “edge of chaos.” The source of this diversity in dynamics is the sensitivity of the dynamics in this regime to initial state of the networks and the structural diversity of the networks (c.f. section 4.3).

The median cycle length (number of states in the cyclic attractor) in ordered RBN scales according to $O(e^{\frac{1}{8}log^2 N})$, in the complex regime according to $O(\sqrt{N})$, and finally in the chaotic regime according to $O(0.5 \times 2^N)$. For small networks, the cycle length in complex regime scales with N [11].

Derrida and Pomeau [21] devised an annealed approximation method for determining the dynamical regime of the RBNs as a function of $\langle K \rangle$ and p, i.e., the fraction of "1"s in the Boolean function of the nodes. Critical connectivity K_c that leads to the complex dynamics of the network is calculated as follows:

$$K_c(p) = \frac{1}{2p(1-p)}. \tag{2.8}$$

If this condition holds, the network is a critical network. However, this result applies to a network at the thermodynamic limit, i.e., $N \to \infty$. For finite size networks, the Derrida criticality [21] is calculated by averaging the spreading of the two states of the network that are one Hamming distance apart after one time step, normalized by the network size [83]. If the result is equal to 1, then the network is in the complex regime, if it is smaller than 1, the network is in ordered regime, and if it is larger than one the network is in the chaotic regime.

Flyvbjerg [25] derived another order parameter for measuring complex dynamics of the network based on the frozen component. In this second method, a network is said to have complex dynamics if 50% of the nodes of the network change their state and the other 50% do not.

In the classical NK model, the probability p of connecting every two nodes is independent. Hence, the resulting network reflects the Erdös-Rényi graph [23] characterized by the binomial degree distribution given by:

$$P(K) = \begin{pmatrix} N \\ K \end{pmatrix} p^K (1-p)^{N-K}, \tag{2.9}$$

where $P(K)$ is the probability of a node having degree K. In the limit of large N, this probability distribution is approximated by the Poissonian degree distribution [1, 64]:

$$P(K_i = K) = \frac{\langle K \rangle^K e^{-\langle K \rangle}}{\langle K \rangle !}. \tag{2.10}$$

Here, $\langle K \rangle$ is the expected degree of the network. However, many natural networks have power-law degree distribution in the form of:

$$P(K) \approx \langle K \rangle^{-\lambda}. \tag{2.11}$$

This makes the Erdös-Rényi's model an implausible model for comparison with real-world data. Consequently, Serra *et al.* [81,82] studied the dynamics of power-law RBNs and found that these RBNs have fewer attractors than the classical RBNs. They also found that the transient length and the cycle periods of the attractors are significantly shorter in power-law RBNs. Darabos *et al.* [20] conducted a comprehensive study of dynamics of RBNs with Poissonian and power-law degree distribution under normal and noisy update rules.

Patarnello and Carnevali [70] conducted the first general study of learning capability of random feedforward networks. In this study, Paternello and Carnevalli used a simulated annealing and genetic algorithms to evolve feedforward networks of logical gates to solve computational tasks such as addition [71]. The ability of these networks to generalize depends on the complexity of the task, the number of gates in the network, and the number of the training examples used during optimization. The ability of the networks to learn from partial inputs and generalize to the entire input space is attributed to the second law of thermodynamics [19]. Later, Van den Broeck and Kawai [93] confirmed the learning in feedforward networks and developed a theoretical framework for analyzing problem complexity and predicting the learning capability of feedforward Boolean networks.

Despite all the discoveries and achievements in the field of ANN, the methods of training general computational networks without a-priori knowledge of their structure and compute nodes remains obscure. All the local learning today depends on the fully connected or otherwise well structured networks with uniform compute elements. Moreover, the compute elements are assumed to calculate some differentiable function of their inputs. In what follows we will explore ways to use RBNs as a basic model of random networks of generic random automata for computational tasks.

3

METHODS AND MEASURES

3.1 UNORGANIZED NETWORKS OF RANDOM AUTOMATA

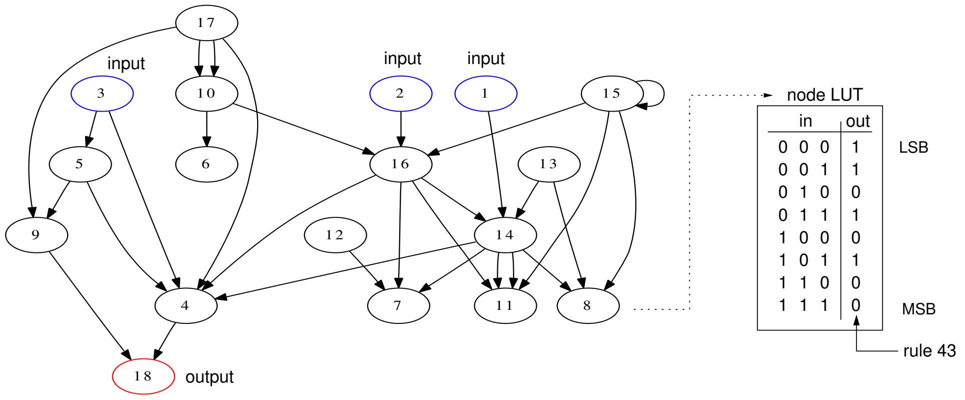
We define the class of networks under investigation in this study as Random Automata Networks (RAN). Although the reader will find shortly that the model is very similar to Kauffman's NK model and to Random Boolean Networks (RBN) [40], the addition of external inputs and outputs to the network introduces subtlties in the definition of $\langle K \rangle$, N, and the implementation of the networks (see Section 3.3.3) that might be confusing, had we used the same naming convention to refer to our model. However, for the purpose of this thesis, we restrict our definition to only binary automata and Boolean functions as follows. We define an RAN as N automata with binary state $\{0, 1\}$. The dynamics of these automata changes according to:

$$F : \{0,1\}^N \mapsto \{0,1\}^N, \tag{3.1}$$

where

$$F = \{f_i | 1 \leq i \leq N\}, \tag{3.2}$$

and each f_i represent a Boolean function of k_i inputs randomly chosen from the set of N automata. We randomly choose the k_i input automata and the Boolean function f_i for each automaton i. Note that the number of inputs may vary from automata to automata. The state of the system is updated synchronously; that is to say each automata σ_i updates its state at time $t + 1$ according to:



in			out
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Figure 3.1: The structure of the Random Boolean Network (RBN).

$$\sigma_i(t+1) = f_i(\sigma_1(t), \sigma_2(t), \ldots, \sigma_{k_i}(t)) \tag{3.3}$$

Figure 3.1 shows the structure of a RBN with inputs and outputs. It is convenient to identify the canonical ensemble of these networks with the number of automata N and the average input per automata

$$\langle K \rangle = \frac{1}{N} \sum_{i=1}^{N} k_i. \tag{3.4}$$

For practical reasons, we limit the maximum number of inputs per automaton to $k_{max} = 8$. Since we will use complex network theory to study the properties of our RAN, we establish the proper analogy between the two here. RAN will map into a directed random graph with N nodes and average connectivity $\langle K \rangle$. Note that there is only one link between two nodes in one direction. We do allow self-loops in this model.

3.1.1 Output Interpretation

The dynamics of the recurrent network that we described depends on three degrees of freedom: the set of Boolean functions of the automata, the connections within the automata, and the initial configuration of the automata (automata states at

$t = 0$). For most of the experiments in the studies we have only changed the connectivity and the functions of the automata. We made this choice merely to reduce the search space. To read the output of the network, we set the state "0" on all the automata and simulate the network for t_1 time steps proportional to N for the dynamics of the network to settle in an attractor. We then run the network for t_2 time steps, also, proportional to N, and observe the activity of the output bits (Figure 3.2). If the state of an output node is "1" for 50% or longer of the observation interval t_2, we say the output of that automata is "1", otherwise we say the output is "0." The inputs of the task we want to solve are wired to random automata in the network. We set the value of the inputs at the beginning of the settlement interval t_1 and keep them fixed until the end of the observation interval t_2.

In [19, 70, 71, 93], the authors conduct their experiment using feedforward random Boolean networks. These are networks that have a layered structure. To construct these networks, we assign each node a random rank value between 0 and $R > 0$. We choose the source of the connection to each node to be strictly from nodes with rank lower than the rank of the destination node. This ensures that there will be no feedback loops in the network. We reserve layer 0 and R for the input and output nodes respectively. In feedforward networks, we will not need to wait for the dynamics of the network to settle down. Yet, we have to wait long enough for the maximum of $R + 1$ time steps to make sure that the network has processed the inputs through all the layers to the output nodes. After this time, the output nodes will have a stable value and will not need any special interpretation.

3.2 LEARNING CAPABILITY MEASURES

Patarnello and Carnevali [70] introduced the notion of *learning probability* as a way of describing the learning and generalization capability of random feedforward networks. They defined the learning probability as the probability of the

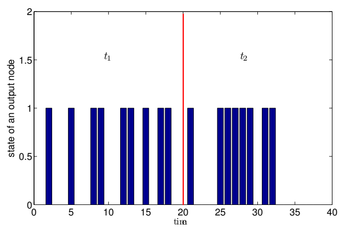


Figure 3.2: Measuring the output of the network. The y-axis shows the state of the output node oscilate between "0" and "1." Because the dynamical nature of the networks, we wait t_1 timesteps for the dynamics of the networks to reach an attractor and measure the activity during t_2 time steps. If the activity is "1" for 50% or more of the time steps during t_2, the outputs will be "1", otherwise "0."

training process yielding a network with perfect generalization, given that the training achieves perfect fitness. Let's first remember the Bayes rule for calculating the posterior probability $P(X = x)$ that a random variable X assumes value x given the evidence $E = e$. Note that the evidence is also described using random variables.

$$P(X = x|E = e) = \frac{P(E = e|X = x)P(X = x)}{P(E = e)}. \tag{3.5}$$

Next, we formalize the learning probability definition as follows:

$$P(g = 1|f = 1) = \frac{P(f = 1|g = 1)P(g = 1)}{P(f = 1)}, \tag{3.6}$$

where f and g refer to the fitness and generalization score of a network respectively. f is calculated by subtracting the error of the output of the network for a subset of all possible inputs from 1, while g is calculated by subtracting the error for all possible inputs from 1 (see Section 3.4). The error is the Hamming distance between the output of the network and the desired output. Note that this definition of the generalization g is different from the popular definition in the machine learning community, which does not include the performance of a system on inputs that have been used during training for generalization. Here, we calculate the generalization over all possible inputs. We argue that our definition of the generalization is better suited for measuring performance of systems that operate under noise. In such systems, the training patterns do not neccessarily produce the same output during testing. Therefore, testing the performance on all the patterns can give us a better estimate of the robustness of the system. Since for systems operating under (internal and external) noise-free conditions $P(f = 1|g = 1) = 1$, we can simplify equation 3.6 and write:

$$P(g = 1|f = 1) = \frac{P(g = 1)}{P(f = 1)}. \tag{3.7}$$

By looking only at the equation 3.7 we will overlook the full statistics since equation 3.7 only depends on perfect cases, i.e, $f = 1$ and $g = 1$. We will therefore define the *perfect training likelihood*:

$$P(f = 1) = \frac{\sum_{r=1}^{R} \lfloor f_r \rfloor}{R}. \tag{3.8}$$

Here, f_r is the fitness of the best network at the end of run r of the experiment and R is the total number of runs. The floor function $\lfloor\rfloor$ only counts the $f_r = 1$ in the summation. The importance of $P(f = 1)$ becomes obvious when we determine how many runs of experiment is enough to gather sufficient statistics. For example for difficult tasks, where we have $P(f = 1) = 0$, the learning probability will be undefined. The learning probability depends on the capacity of the learning machine and the number of training patterns. The latter is, by convention, expressed as the fraction $s = \frac{m}{m'}$, where m is the number of training patterns and m' the number of all possible patterns; for a specific problem [92].

An example of the calculation of learning probability and perfect training liklihood is given in Figure 3.3. The goal is to learn a function of three input variables. The training sample size is 4 out of 8 possible patterns therefore $s = 0.5$. We run the experiment 4 times. At the end of each run we record the fitness and the generalization score of the best networks according to equations 3.17 and 3.18. Note that both fitness and generalization scores are normalized to be between 0.0

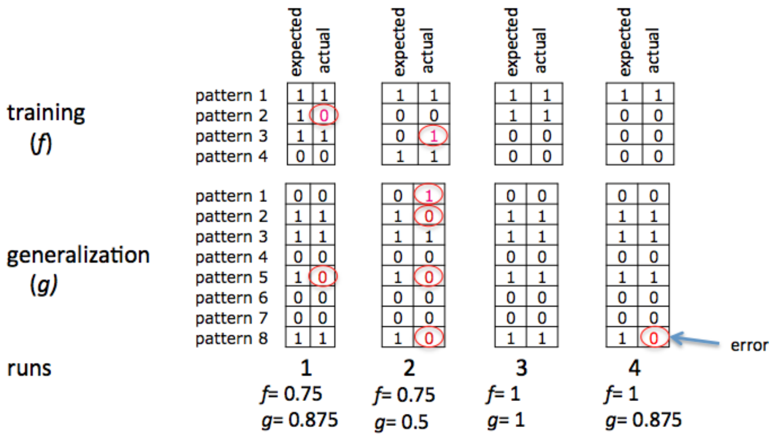


Figure 3.3: Calculating learning probability. The circles mark the erroneous outputs for the even-odd task with 3 inputs. There are 8 possible input combinations. The training sample size in this example is 4. For each generation a new training sample is generated. We repeat the experiment four times to calculate the probability $P(g = 1|f = 1)$.

and 1.0 inclusively. We see from the figure that in 2 runs out of the 4, we find networks with fitness of 1.0 and in one run we have a generalization score of 1.0. Consequently, the perfect training likelihood is 0.5 and the learning probability is 0.5.

The probabilistic measures, such as the learning probability described above, only focus on the perfect cases and hence describe the performance of the training process rather than the effect of the training on the network performance. Thus, we define the *mean training score* as $\beta(s) = \frac{1}{r}\sum_r f_{final}$ and the *mean generalization score* as $\beta'(s) = \frac{1}{r}\sum_r g_{final}$, where f_{final} and g_{final} are the training fitness and the generalization fitness of the best networks respectively at the end of training. For instance, in the example in Figure 3.3, the mean training score is $\frac{1}{4} \times (0.75+0.75+1+1) = 0.875$ and the mean generalization score is $\frac{1}{4} \times (0.875+0.5+1+0.875) = 0.8125$.

To compare the overall network performance across all s values, we introduce a *cumulative measure* for all four measures as defined above. The cumulative measure is obtained by a simple trapezoidal integration [99] to calculate the area under the curve for the learning probability, the perfect training likelihood, the mean generalization score, and mean training score.

3.3 COMPUTATIONAL TASKS

3.3.1 Task Description

We use five computational tasks to evaluate the fitness of the networks over the course of simulated evolution. These tasks are:

1. The bitwise AND task.
2. The mapping task.
3. The even-odd task.
4. The full-adder.
5. The CA rule 85.

We give a description of these tasks in this section. Furthermore, we analyze these tasks in terms of their complexity from a learning and information theoretic perspective.

The **bitwise AND** task is defined over two sets of binary numbers of l bits. To implement that task correctly, a machine would have to calculate the correct AND operation of each respective bit of all possible pairs of numbers in the sets. This means that there are $I = 2^{2l}$ different combinations that the system has to get right.

As for the **permutation** task, the system receives a l-bit long binary input pattern and has to generate the same number of “0”s and “1”s in an l-bit output

A	B	C	*output*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Table 3.1: The truth table for the even-odd task.

in any order. Note that an input pattern with n "1"s may not have a unique solution. In fact there are

$$\binom{l}{n} = \frac{l!}{(l-n)!n!} \tag{3.9}$$

different input patterns with l bits and n "1"s each of which may be also considered a candidate solution to any of these patterns.

The **even-odd** task, sometimes called the parity task or addition modulo 2, computes the summation over the l input bits modulo 2. Algorithmically we can say that the system should output a "1" if there is an odd number of "1"s in the input, otherwise the system outputs "0." This is traditionally known as the XOR task in the ANN literature and is not linearly separable. Table 3.1 depicts the truth table of the 3-bit even-odd task.

The **full-adder** task is the implementation of a one bit full-adder circuit. The inputs of this task are two 1-bit binary numbers A and B and an input carry C_{in}. The output is a 1-bit summation S and an output carry C_{out}. The values of S and

A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 3.2: The truth table for the full-adder.

C_{out} are calculated as follows:

$$\begin{aligned} S &= A \oplus B \oplus C_{in} \\ C_{out} &= (A \cdot B) + (C_{in} \cdot (A \oplus B)) \end{aligned} \tag{3.10}$$

Here, $\oplus$ represents the exclusive OR, $\cdot$ represents the AND, and the $+$ represents the OR Boolean functions. The truth table of the full-adder is given in Table 3.2.

The **CA rule 85** task is an implementation of the rule 85 of elementary cellular automata. Using Wolfram's encoding of CA rules, rule 85 takes three inputs A, B, and C and outputs $\bar{C}$. This task is defined for three inputs, but its value only depends on one input. Table 3.3 shows the truth table of CA rule 85.

In the next two sections, we briefly describe how the complexity of these tasks are different from one another. We first look at the perspective of a classifier to see how likely it is to find a machine that can perform a task. We will then analyze through information theoretic calculations how difficult each of these tasks would be for a classifier.

A	B	C	$output = \bar{C}$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Table 3.3: The truth table for CA rule 85.

3.3.2 Task Complexity: The Learning Machine

Van den Broeck [93] introduced the concept of phase volume V of a Boolean function. V is the number of classifiers that can produce this function correctly. He defined the probability of the Boolean function F of volume V_F, $P(V_F)$, to be the fraction of the class of classifiers that realize F,

$$P(V_F) \propto V_F^{-\alpha}, \tag{3.11}$$

with $\alpha \approx 0.7$. This is calculated via sampling for the class of feed-forward random Boolean networks in which each node receives two inputs. This means that if we keep randomly sampling the space of feedforward Boolean networks, we find networks that realize F with probability $P(V_F)$. This indicates how difficult it is for this class of networks to solve this task using a stochastic optimization technique. This also could be interpreted as the complexity of F.

Yet the concept of the phase volume can become useful in calculating the richness of a class of classifiers. We can then calculate the entropy of realizable

Boolean functions of i variable in this class using:

$$S = -\sum_{f=0}^{2^{2^i}-1} p_f log_2 p_f. \tag{3.12}$$

Here, p_f denotes the probability that a classifier realizes function f. We calculate p_f by creating random networks and simulating each network to measure its output. p_f is the fraction of the networks that realize the function f. The absolute bounds on the value of S is easy to calculate:

$$0 \leq S \leq 2^i. \tag{3.13}$$

The exact value, however, depends on the distribution of the p_f over all possible functions. The maximum value occurs for a uniform distribution and the minimum value occurs if only one function is implementable. For example, for $i = 3$, if the class of networks with $\langle K \rangle = 2.0$ and $N = 20$ implements all the functions with uniform probability distribution, then the entropy will attain its maximum value of $S = 8$ bits. This implies that if we simulate 256,000 networks, we will see that each of the 256 functions is realized by 100 networks. On the other extreme, if the class only implements one function, then $S = 0$ bits.

Figure 3.4 illustrates the entropy of the realizable Boolean functions of three variables using RBNs as the computing model. The sample is taken over 30,000 networks with $N = 20$ for each $\langle K \rangle = 1.0, 1.5, 2.0, 2.5, 3.0, 3.5$, and 4.0. For each network, we sample 300 initial configurations chosen from Independent and Identical Distributions (IID). The entropy starts at 1.5 for $\langle K \rangle = 1.0$ and rises to a maximum of about 4 at $\langle K \rangle = 3.5$. For $\langle K \rangle \geq 3.5$, the entropy declines again suggesting increasing chaoticity in the dynamics of the system beyond a point at which the *computational capacity* (measred by the entropy of the realized functions) declines. For most values for $\langle K \rangle$, the entropy is much lower than the maximum (i.e., 8), possible 8, and even the maximum empirical entropy is only half of the

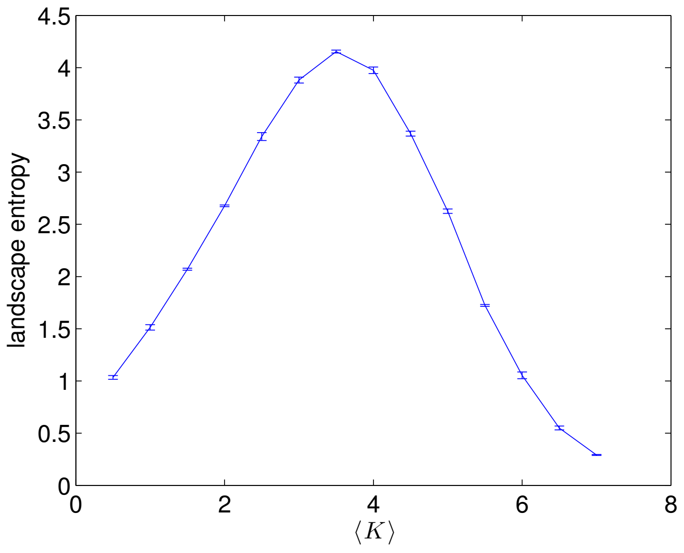


Figure 3.4: Boolean function landscape in the space of RBNs. In-degree distribution of the network is binomial and $N = 20$.

theoretical maximum value. This suggests that the distribution of realization of the 256 possible functions is highly skewed.

The functional landscape of the model depicted in Figure 3.4 does not show how many different functions are realized by a particular network, rather, it focuses on the entire class of networks. However, we could ask the question, "Given a particular network, how many different functions does that network implement if we start from different initial configurations?" This question is of great importance when we analyze the stability and the reliability of computations using RBNs. Thus, we calculate the entropy of realizable functions for individual networks over 300 different initial configurations and average that over 10,000 networks for each connectivity class $\langle K \rangle = 0.5, 1.0, 1.5, ..., 7$. Figure 3.6 illustrates the results. The value of the entropies for various $\langle K \rangle$ are well below 1. For all initial configurations, if a network realizes the same function, the functional entropy of the network will be zero. If each intial configuration causes the network to realize a different

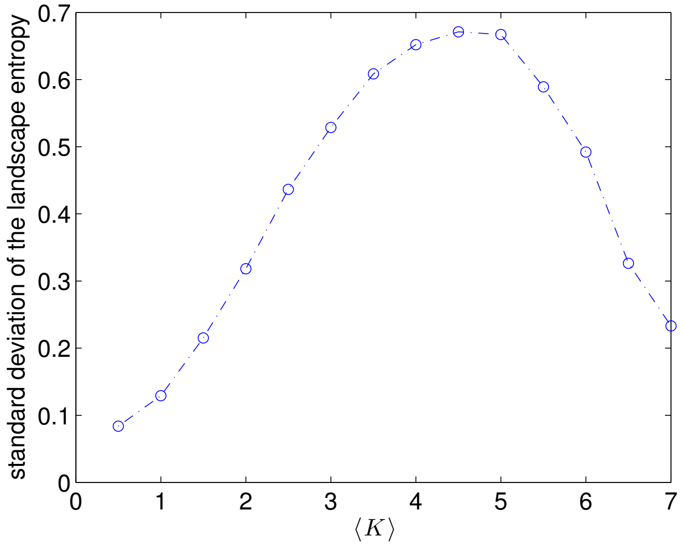


Figure 3.5: Standard deviation of the entropy of the realizable functions for individual networks. K is the average connectivity of the network.

function, the functional entropy of the network will be near 8. This suggests that the dominant factor in the dynamical behavior of the network is the structure of the network and not the initial configuration. Thus, there are only a few sets of distincts dynamical attractors that are dictated by the structure of the networks. At $\langle K \rangle = 4.5$, the attractors of the ensemble are most sensitive to their intial configurations.

In order to get a clearer picture of the landscape of the realizable functions, we plot the frequency distribution of the functions and sort them according to their frequency. Figure 3.7(a) shows the frequency distribution of the realizable functions, sorted according to their p_f, for networks of various $\langle K \rangle$. This distribution is a rapidly decreasing distribution. On the log-linear scale we see regions where the frequency decreases exponentially with stepwise discontinuity. We plotted the same frequencies on the log-log scale (Figure 3.7(b)) to make the discontinuity more visible. The discontinuity suggest a hierarchical landscape or the likelihood

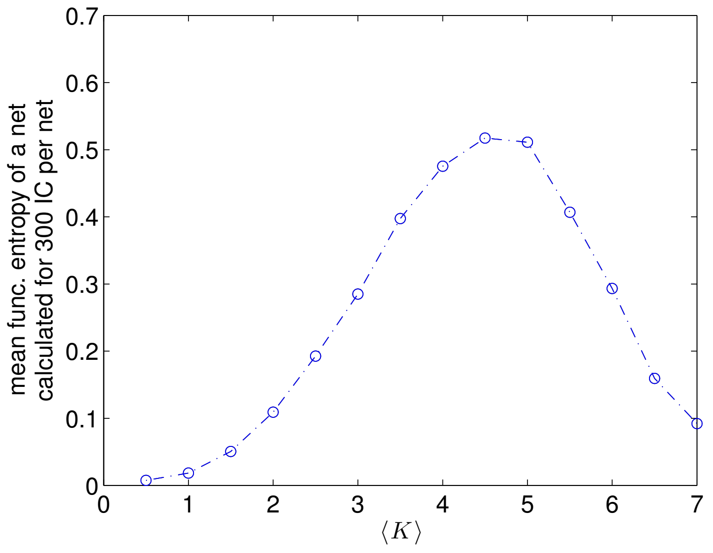
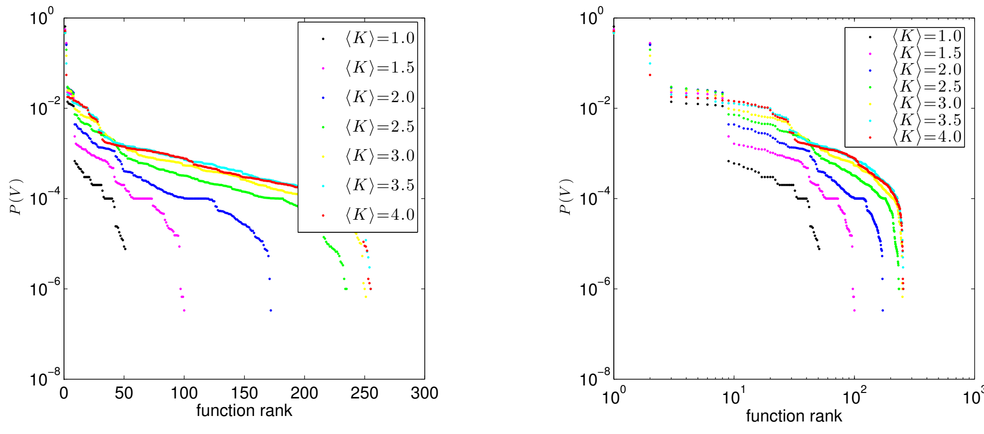


Figure 3.6: Entropy of the realizable functions for one network starting with 300 different initial configurations. The degree distribution of the network is binomial $N = 20$. K is the average connectivity of the network.

of realization of the Boolean functions with plateaus and sharp jumps to neighboring regions. The hierarchical levels belong to the classes of the Boolean functions that have similar probability of realization (p_f) by the networks. This suggests that from the perspective of a learning machine, there are distinct *"difficulty"* classes in the space of 3-input Boolean functions. This finding matches similar observations as presented in [74,93] for feedforward networks with two inputs per node.

3.3.3 Solving Tasks

A computational task is characterized by a static or dynamic mapping of I binary inputs to O binary outputs. To realize this mapping through RAN, we have to connect the input bits to the network. There are four different methods by which this could be done. Throughout our study we only use the second method, but we mention the other three for completeness and explain the implication of using



(a) log-linear (b) log-log

Figure 3.7: Hierarchical structure in the functional landscape of RBNs for different connectivities in log-linear scale (a) and log-log scale (b). The y-axis shows $P(V)$, the probability of function f having phase volume of V. The x-axis shows the functions ranked based on their position in a list sorted from largest to smallest $P(V)$. This suggests the existance of distinct *"difficulty"* classes in the space of 3-input Boolean functions from the perspective of RBNs as learning machines that compute these functions.

each method:

1. One way to connect the inputs is to assume I additional nodes outside of the network and wire them randomly to the N nodes in the network. Consequently, one can either count the links between this input nodes and the automata nodes in the network as part of the automata's $\langle K \rangle$ inputs or we can ignore these links. If we ignore these links, the graph-theoretical definition of $E = K \times N$, where E is the number of edges in the graph, does hold.

2. The second way of connecting the inputs is to count the connections in the in-degree of the receiving compute nodes in the network, since these inputs actually influence the state of those nodes. In this case $E > K \times N$.

3. The third method consists of assuming that the I inputs are part of the network. This new network will have $I+N$ automata. The practical problem with this model is that when we are connecting links to the nodes randomly, some of these links will be assigned to the I nodes and will therefore not be used. This is because the signals in the I nodes are fixed. Although the training algorithm, as we will see, might get rid of these extra links, this will cause a topological bias in the dispersion of the links.

4. Finally, we can include the number of input nodes I are in the system size N, but make sure they have no input links themselves. In this case, we have $E = K \times (I + N)$, there will be some nodes in the graph that will have more than one link with their neighbors.

As a practical matter, we choose not to include the I input nodes in the count of the network nodes. However, we count the links from these nodes to the N network nodes as part of the E graph links. Note that this choice implies $\langle K \rangle \geq \frac{E}{N}$.

We define a computational task by an I-input to O-output mapping. Let the set

$$M' = \{0,1\}^I \mapsto \{0,1\}^O \tag{3.14}$$

represent the entire input-output mappings for a specific task. Therefore the number mappings for the task will be

$$m' = |M'| = 2^I. \tag{3.15}$$

At each generation of the evolution, the GA chooses a random subset $M \in M'$ of size T to calculate the fitness of the individuals in the population. We call M the training sample and $T = |M|$ training sample size.

3.4 EVOLVING NETWORKS TO PERFORM COMPUTATION USING GENETIC ALGORITHMS

The degrees of freedom in RBNs create a complexity catastrophe as the size of the network increases [47]. For every $\langle K \rangle$ and N, the number of possible RBNs, $G(N,K)$, is roughly given by [32]:

$$|G(N,K)| = \left(\frac{2^{2^K} N!}{(N-k)!} \right)^N . \tag{3.16}$$

Any exhaustive or heuristic search in this space with a rugged fitness landscape [47] is clearly hopeless. We use Genetic Algorithms to explore this large space of networks ensemble for networks that can perform specific computational tasks (see Section 3.4).

Genetic Algorithms (GA) are a class of population based metaheuristic optimization techniques [10] inspired by natural evolution. In this study we use GA to evolve networks to perform specific computational tasks. GA has the benefit that it can stochastically explore the search space using a population based optimization and find optimal solutions according to an objective function. This makes the GA well suited for searching in spaces that undergo a complexity catastrophe.

Here, we describe the fitness function, and genetic operators for our population. We define the fitness as follows:

$$f = 1 - \frac{1}{m} \sum_{M} (expected - actual)^2. \tag{3.17}$$

Here, *expected* refers to the expected value of the output and *actual* refers to actual value measured from the output of the network. The summation term in equation 3.17 is the sum of the Hamming distances between expected and actual outputs, normalized by the number of training patterns. To observe how the generalization of the system changes over time, we calculate the generalization score for each individual by

$$g = 1 - \frac{1}{m'} \sum_{M'} (expected - actual)^2. \tag{3.18}$$

This generalization does not in any way affect the operation of the GA, but rather informs us about how well the individuals can perform the required computation.

Before we can use the GA to evolve networks, we need to create a genetic representation of the network. We do this by encoding the directed graph that represents the network in an adjacency list and append a concatenated list of the output column of the Look-Up Table (LUT) of the Boolean function of each node in the network to the end of it (Figure 3.8). This implies that the genomes of the individual networks have different lenghts.

Normally, the evolution takes place from generation to the next by applying four operations:

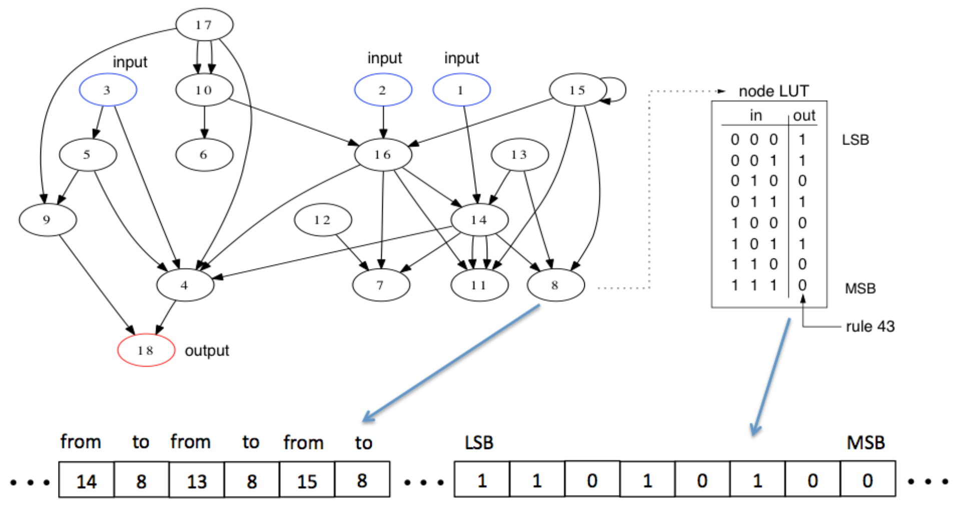


Figure 3.8: Genomic representation of the RBN.

1. Selection: choosing individuals in the population for recombination and mutation.

2. Recombination (crossover): combining two selected individuals to create offsprings for the next generation.

3. Mutation: randomly changing a selected individual to create a new offspring for the next generation.

4. Replacement: selecting individuals from the parent and offspring populations for the next generation.

Our experiments show that using a recombination operator in this study eradicates diversity in the population and hinders fitness maximization. We therefore use pure mutation in our experiments. The mutation takes place on the links and the function part of the genome independently. To mutate the functions, the operator randomly picks a position in the functional part of the genome and flips the binary value at that position. For the mutation of the connections in the network, we define two different operators. The length of the genome of each individual changes over generations due to changes in the number of links and the size of the LUTs. First, we use a rewiring operator that chooses a link and randomly picks a new node either for its destination or for the source. Note that the mutation under this operator preserves the average connectivity of the network. The second operator is the link addition or deletion operator. This operator randomly chooses a location in the adjacency list and either deletes the link at that location or inserts a new links between two randomly chosen nodes at the location. The process of adding and deleting the link may occur $1 + \alpha$ times with probability $p(\alpha) = \frac{0.5^{\alpha}}{2}$. The number of generations and the population size during the evolution depends on the specific experiments and we will clarify them in section **??**.

3.5 SOFTWARE FRAMEWORK

For our simulations we used a C++ network simulator initially developed at the Teuscher-Lab for high performance network simulation. We extended the original framework with the following features:

1. Augmented RBN model to with external input and output signals.
2. Layered structure (necessary to simulate feedforward networks).
3. Output interpretation module (necessary to interpret output of the RBN).
4. ParadisEO [17] GA integration.
5. Genome export module to convert networks into their genetic representation.
6. Recombination operator.
7. Three different structural mutation schemas.
8. Functional mutation scheme.
9. Derrida calculation modules.

10. Network topology export module for graph-theoretical studies.

11. Repair operator for recovering damaged feedforward networks after the mutation and recombination.

12. A flexible task solving module that interfaces the RBN and GA engines, with ability to define mapping or sequential tasks for the networks.

To perform evolutionary optimization on the networks we chose the ParadisEO metaheuristic optimization framework. ParadisEO is a open source software framework for different metaheuristic optimizations. ParadisEO can be accessed at `http://ParadisEO.gforge.inria.fr/`. We used the open source Brain Connectivity Toolbox for our graph-theoretical studies. The Brain Connectivity Toolbox software and documentations can be found at `https://sites.google.com/a/brain-connectivity-toolbox.net/bct/Home`.

4

EVOLUTION OF NETWORKS WITH VARIABLE $\langle K \rangle$

4.1 FINDING THE OPTIMAL NETWORK CONNECTIVITY

In chapter **??** we reviewed the results from the evolution of the networks with the constraint of fixed connectivity $\langle K \rangle$ using a rewiring mutation scheme for the structure of the networks as described in section **??**. In this chapter, we relax this constraint and let the average connectivity change. We observe that this type of evolution results in the convergence of the connectivity to a critical value K_c that depends on the system size N. We argue that for large systems this critical value converges to $K_c \approx 1.87$. This critical connectivity has been hypothesized to be conducive to complex computation while maintaining maximal adaptability and robustness to structural and dynamical perturbation [47].

To let the average connectivity of the network change during the evolution, we let the number of the links in the network change. For the fixed $\langle K \rangle$ evolution, we used the rewiring mutation scheme in which randomly chosen endpoints of a link in the network were attached to another randomly chosen node. We will abandon this mutation scheme in this chapter and use a link addition or deletion scheme instead.

4.1.1 Revised Mutation Scheme

We define a mutation step to include adding a link between two randomly chosen nodes or deleting an randomly chosen link form the network. To let the evolution converge faster, we let mutation repeat the process $1+\alpha$ times with probability

$p(\alpha) = 0.5^{\alpha}$, where $\alpha > 1$. This probabilistic scheme will be particularly helpful as we move to experiments with larger networks. The functional mutation is unchanged and performed as before by flipping the random entry of the look-up table of a randomly chosen node.

Our ultimate goal is to find and study a critical average connectivity that emerges from the evolution of the networks as a result of the fitness maximization. Note that in our evolutionary setup, the population is subject to significant amount of disturbance due to our high mutation rate, i.e., 0.8. Therefore, the existence of a critical connectivity K_c that can maximize the fitness is also an evidence that networks with K_c maintain maximum robustness and adaptability while performing complex computations. Although the critical connectivity of $K_c = 2$ for the networks have been hypothesized and observed by many researchers [11, 47, 51, 54, 55, 67], as far as we know, this is the first time that the critical connectivity has been established for networks in a concrete computational context, i.e., with specific tasks and not in a closed system (such as classical RBNs with no external inputs).

4.1.2 Evolutionary Steady State

As a preliminary stage, we wanted to find a target connectivity that the evolutionary pressure pushes the networks toward. First, we have to observe that the population under mutation and selection reaches the Evolutionary Steady State (ESS). ESS is a state of stable maximum fitness during the evolution. We begins by evolving two populations of networks for the even-odd task. The first population starts from the initial connectivity $\langle K \rangle = 1.0$ and the second population starts from the initial connectivity $\langle K \rangle = 7.0$. We repeat this experiment with different networks of size $N \in \{5, 10, 20, \dots, 60\}$. The networks have to evolve to solve the even-odd task with three inputs. We carry on the training with $T = 1$, $T = 4$, and $T = 8$. For $T = 4$ and $T = 8$, the population dynamics soon reach

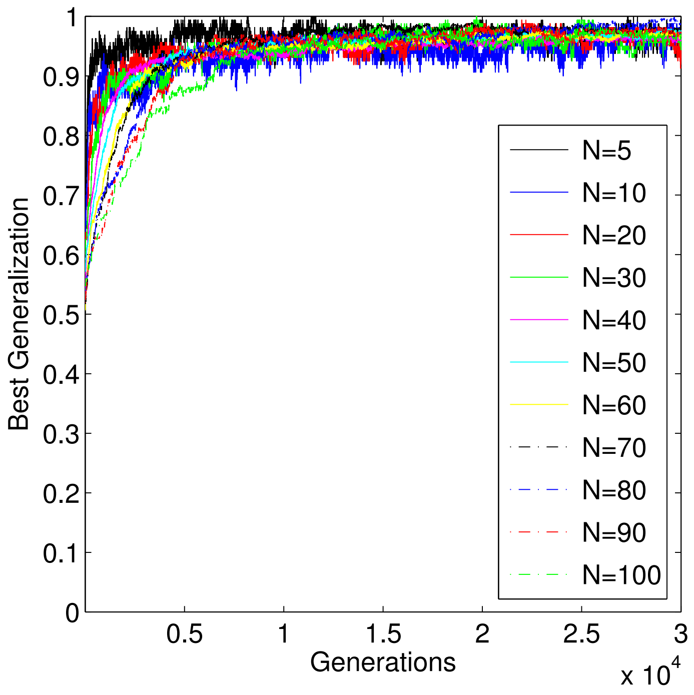


Figure 4.1: Population dynamics of the fitness. The average fitness reaches a stable maximum suggesting the existence of Evolutionary Steady State (ESS) dynamics after 10,000 generations.

the ESS. During the ESS, the connectivity shows convergence toward a specific K_c value. Therefore, for the rest of experiment, we can only use $\langle K \rangle = 1.0$ as a starting connectivity.

Next, we have to study the critical connectivity and its role in the context of learning, generalization, adaptation, and robustness. We evolve networks of size $N \in \{5, 10, 20, 30, \ldots, 100\}$ with an initial connectivity of $\langle K \rangle = 1.0$. We run the experiments for 30,000 generations. For this number of generations, even for the largest network sizes, we see a very strong convergence the maximum fitness (Figure 4.1). The reason that the fitness values in Figure 4.1 are lower than 1 is that they are averaged over 30 runs. Clearly, for generations higher than 5,000, we

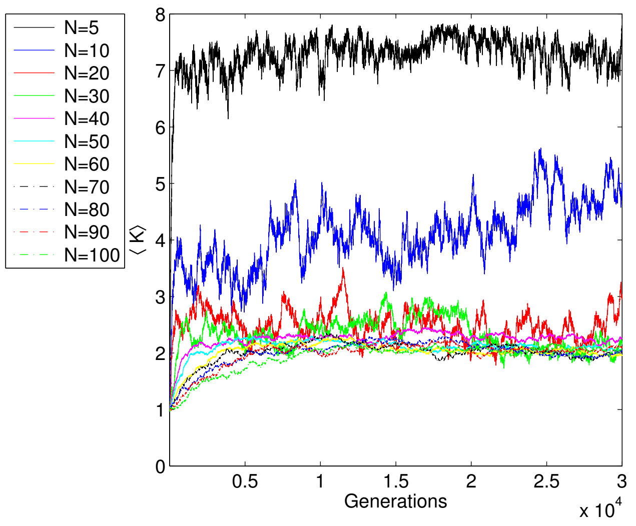


Figure 4.2: Evolution of $\langle K \rangle$ in the population. During the steady state the value of $\langle K \rangle$ is stable (for very large and very small networks) or fluctuates around a central point (for $N = 10$).

observe an ESS. We also observe a convergence of the $\langle K \rangle$ throughout the evolution and in the final population (Figure 4.2). According to these two facts, we have a convergence to a connectivity that is favored by evolution. This connectivity not only improves the computing power of the networks but also its resilience to a fairly high perturbation rate of 0.8. We can now study the scaling properties of the critical K_c with respect to the system size, the training sample size, and the task complexity.

4.1.3 Scaling Property of the Critical Connectivity

There are two different ways to look at this critical connectivity. First, we can take the snapshot of the final population and look at its properties. Figure 4.4 shows the result of the average final $\langle K \rangle$ for all the networks of various sizes for the extreme cases $T = 1$ and $T = 8$. The purpose of the scaling plot is to derive how the critical connectivity changes with the system size. Four natural equations that are candidates to describe our data shown in Figure 4.4 are:

1. Model 1 (M1): Power-law equation with three free parameters a, b, and c of the form:
$$K_c = aN^b + c. \tag{4.1}$$

2. Model 2 (M2): Exponential equation with four free parameters a, b, c, and d of the form:
$$K_c = ae^{bN} + ce^{dN}. \tag{4.2}$$

3. Model 3 (M3): Power-law equation with two free parameters a and b with the form:
$$K_c = aN^b. \tag{4.3}$$

4. Model 4 (M4): Exponential equation with two free parameters a, b of the form:
$$K_c = ae^{bN}. \tag{4.4}$$

After careful analysis of the sum squared error, the R-squared measure for goodness of fit, and the Akaike Information Criteria (AIC), we picked Eq. 4.1 to be the best model for our data (see Section 4.1.4). Luckily, Eq. 4.1 naturally fits out desired description for scaling to the parameter c. The parameter c is the value of K_c in the thermodynamic limit. In other words, this equation represent a power-law decay with the value of K_c when $N \to \infty$ and $b < 0$.

We plot the data and the resulting curves from the fit equation on a log-log scale (Figure 4.4). Two general scaling behaviors are evident. First, the curve for the even-odd and the full-adder tasks with $T = 8$ shows a concave shape with a decreasing rate of decay. Second, the curve for the even-odd and the full-adder tasks with $T = 1$ and the R85 task with $T = 1$ and $T = 8$ show a convex shape with an increasing rate of decay. The latter case asks for a deeper investigation because it goes against what is observed from the average connectivity in the final population.

To further investigate the scaling result, specially for the contradictory cases of the convex curves, we look at the distribution of the average connectivity in the evolved population. We expect that the distribution of $\langle K \rangle$ for the limiting network size in this study ($N = 100$) to be a sharply peaked binomial-like distribution around the calculated parameter c (Table 4.2). For $T = 1$ and $T = 8$ for individuals with generalization score $g > 0.8$, we create frequency distributions from the connectivity of the evolved population as well as their Derrida sensitivity [83] (Figure 4.6). For $T = 8$, both full-adder and even-odd tasks derive the networks toward a peak at the critical connectivity. For the R85 task, however, the distribution is wider. For $T = 1$ there is no individual in the population that can fully realize the even-odd and the full-adder task. The R85 task is realized, but in a very wide distribution around $\langle K \rangle$. We conclude that for $T = 8$, the convergence for the R85 task is very weak, and in the $T = 1$ case, we do not see any convergence at all. The lack of convergence in the connectivity invalidates the scaling plot for a training sample size of $T = 1$ for the even-odd, the full-adder tasks, and all training sample sizes of the R85 task. We conclude that the information in the training sample sizes is not enough to impose evolutionary pressure on the networks.

The lack of convergence of $\langle K \rangle$ during the steady state rises an interesting question about the meaning of the free parameter c and the final average $\langle K \rangle$ in the evolved population. What we know for sure is that the the real value of $\langle K \rangle$ is changing and does not converge for $T = 1$, or for any training size of the task R85. However, the value seems to fluctuate around a certain target $\langle K \rangle$ and does not visit all possible values with the same probability. The evolution of the networks with respect to the objective function calculated on one input pattern ($T = 1$), although this imposes a very low evolutionary pressure, still causes some

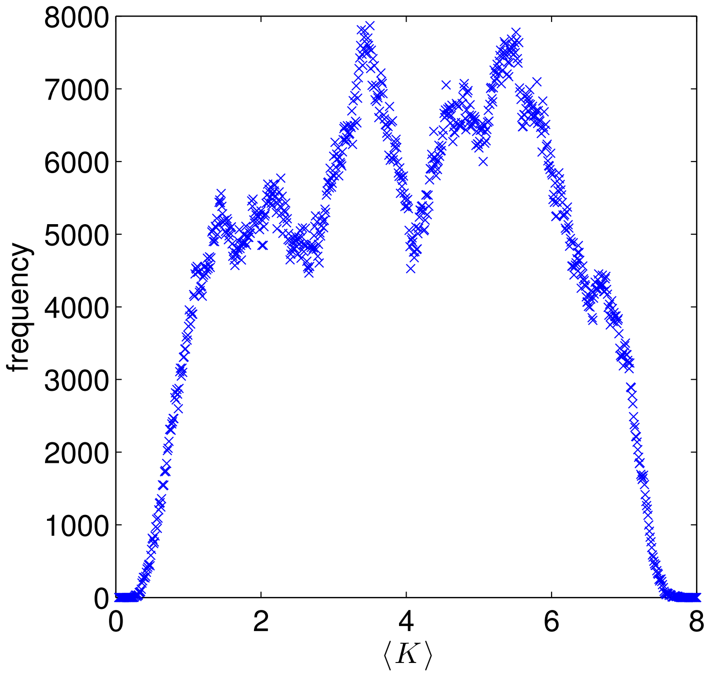


Figure 4.3: Fitness-free evolution of the networks reveals that $\langle K \rangle$ that occurrence of $\langle K \rangle$ is a binomial random variable. The average of the distribution is 4.0, which is in the middle of minimum and maximum possible in-degree per node in our networks.

of the networks to perform worse than the others, and therefore be deleted from the population. But what if all the networks were equally fit? Or in other words, what if we do not subject the networks to a fitness evaluation at all?

4.1.4 Choosing the Right Model for Scaling Data Using AIC

In Section 4.1.3, we picked the model 1 (M1) with three degrees of freedom (free parameters) of the form: $aN^b + c$ to describe the scaling behavior of the K_c as a function of N. In this section, we describe a rigorous statistical method behind our choice for the M1 model. We used Akaike Information Criteria (AIC) [36] to pick the most plausible model that describes our data. AIC is a relative measure of goodness of fit of a statistical model rooted in information entropy. Loosely speaking, AIC describes the trade-off between bias and variance in a statistical model. This can also be thought of as the trade-off between accuracy and the complexity of the model. The method we describe here is well described in [16].

The general form of calculating the AIC measure for a model is as follows:

$$AIC = -2ln(likelihood) + 2DF, \tag{4.5}$$

where the likelihood is the ratio between the prediction of the model and the real data, and DF is the number of free parameters in the model. AIC can also be calculated from the residuals of a fit using:

$$AIC = 2ln(\frac{SSE}{n}) + 2DF. \tag{4.6}$$

Here, SSE is the sum squared of the residuals from the fit and n is the sample size. AIC requires a bias-adjustment for small sample sizes ($\frac{n}{DF} < 40$). In our case, this adjusted AIC is calculated using:

$$AIC_{adj} = nln(\frac{SSE}{n}) + 2DF + \frac{2DF(DF+1)}{(n - DF - 1)}. \tag{4.7}$$

The best model to describe a dataset is selected via the relative distance of the models with the "truth." To calculate this relative distance, we have to first subtract the minimum adjusted AIC $minAIC_{adj}$ from the AIC_{adj} for all the models as follows:

$$\Delta AIC_{adj} = AIC_{adj} - minAIC_{adj} \tag{4.8}$$

To quantify the plausibility of each model, we calculate the relative likelihood of our model given the data. This is given by:

$$\ell = e^{-0.5\Delta AIC_{adj}}. \tag{4.9}$$

For the final comparison, the models are now weighted based on their calculated ℓ as follows:

$$w_{\ell_i} = \frac{\ell_i}{\sum_{i \in models} \ell_i}. \tag{4.10}$$

w_{ℓ_i} is the Akaike weight for model i, ℓ_i is the relative likelihood of model i. The denominator is the sum over the relative likelihood of all the models and ensures the addativity of the weights. Table 4.1 shows the w_{ℓ_i} for models M1, M2, M3, and M4 (see Section 4.1.3) for the same datasets used in calculating final parameter for the power-law in Table 4.3 the plots on Figure 4.5. According to this analysis, model M1 that we chose to be the right scaling model is orders of magnitude more successful in describing the data except for R85 task. R85 task is not described by the power-law because of the lack of convergence during the ESS. However, the exponential law of the model M3 seems to be describing the scaling behavior of the R85 task reasonably well. We have included the results for calculation of AIC_adj, ΔAIC_{adj}, and ℓ for the datasets that we used for fitting in Appendix A.

4.1.5 Fitness-Free Evolution

We repeat the evolution of the networks of size $N = 100$ for 50,000 generations from different initial $K \in \{0.1, 1, 4, 7, 7.9\}$. Since we bypass the fitness evaluation,

task	T	$w_{\ell_{M1}}$	$w_{\ell_{M2}}$	$w_{\ell_{M3}}$	$w_{\ell_{M4}}$
FA	4	0.9827	0.0076	0.0096	0.0000
EO	4	0.9997	0.0003	0.0000	0.0000
R85	4	0.1359	0.0007	0.8634	0.0000
FA	8	0.9577	0.0421	0.0002	0.0000
EO	8	0.9901	0.0099	0.0000	0.0000
R85	8	0.0775	0.0004	0.9221	0.0000

Table 4.1: AIC weights calculated for the four proposed models.

each individual is assigned the fitness value 0 in each generation. Therefore, the selection process is a purely stochastic process with no implicit or explicit bias. As a reminder to the reader, the maximum connectivity per node for our networks is 8, meaning each node in the network can have any connectivity between 0 to 8. We therefore expect a system with unconstrained evolution (through selection and replacement due to evaluated fitness) to visit all possible states in the average connectivity space $0 \leq K \leq 8$. We expect the average of the connectivity to be $\langle K \rangle = 4.0$ during the evolution and the fluctuations to form a binomial distribution. Studying the average $\langle K \rangle$ of the final population does not give us enough information about the distribution because: (1) 30 runs is not enough for gathering statistics for average $\langle K \rangle$ and (2) the final population is just a snapshot of the last state of the evolution and does not give information about the dynamics of the evolution. However, since in one view this evolution is in the steady state for all the experiments, we can look at the connectivity of the networks in the population throughout the evolution and see how it evolves in the $\langle K \rangle$ space. The result of this experiment is shown in Figure 4.3. The evolution of $\langle K \rangle$ of course is oscillatory. We create a histogram from all the experiments combined and we find that the average connectivity forms a binomial distribution with the mean of $\mu = 4$. If we separate the distributions by their initial connectivity value, we find that the distribution still looks like a binomial distribution, but with a slight bias toward the initial $\langle K \rangle$. This works since we can attribute the bias to the few initial generation of the evolution before the connectivity of the individuals in the population fully disperse in the $\langle K \rangle$ space. Moreover, it is obvious that we do not have a situation in which the $\langle K \rangle$ of the population converges to smaller and smaller values with increasing system size increases as predicted by the convex curves. On the contrary, $\langle K \rangle$ should go higher than the c for the $T = 8$ full-adder and the even-odd tasks, and get closer to 4.0, if not exactly 4.0.

The results in Figure 4.4 are calculated for the average connectivity of the final population after the evolution. But since the evolution shows a steady state, we can use the same trick that we used to study the fitness-free evolution. That is, we can detect the steady state of the evolution of the population with respect to the objective function and track the desired property (in this case $\langle K \rangle$) during the evolution. We collect all the $\langle K \rangle$ data during the course of the evolution and create a distribution of $\langle K \rangle$ by binning the data. Between the raw data and the frequency distribution out of the binned data, we have five different ways to pick the right K_c. From the frequency distribution, we could either calculate the position of the peak of the distribution or choose the expected value of $\langle K \rangle$ from the calculated distribution. From the raw data, we could use the mean of the $\langle K \rangle$ or the median of the $\langle K \rangle$. Alternatively, we can use a Gaussian mixture method to model the raw data and calculate the mean and the variance. Unfortunately, decisions based on the frequency distribution depend on the bin size used to create the distribution. If we could use the raw data we could overcome this shortcoming. A few experiments show that using the mixture models is the most robust approach against variations in the data.

We recalculate the scaling of K_c using the mean of the Gaussian mixture model of the raw data (Figure 4.2). Instead of $T = 1$, however, we calculate K_c for $T = 4$. This will allow us to relate K_c to learning and generalization. The calculated free parameters of the scaling model are listed in Table 4.3. To show the statistical convergence in a more visual way, we have plotted the distribution of the $\langle K \rangle$ after the evolution for both $T = 4$ (Figure 4.7(a)) and $T = 8$ (Figure 4.6(a)). The peak of the distribution near $\langle K \rangle = 2$ suggests most networks after the training have the critical connectivity and our assumption of convergence of the connectivity in the population to K_c is valid. The distribution of the Derrida measure for the dynamics of the networks is also calculated for $T = 4$ (Figure 4.7(b)) and $T = 8$ (Figures 4.6(b)). The peak of the distribution around 1 on the Derrida distribution shows that most networks after the training are in the complex dynamical regime. These observations suggest that learning and generalization correlates with complex dynamics on the networks. For $T = 1$, however, the distributions for the final population $\langle K \rangle$ (Figure 4.8(a)) and Derrida (Figure 4.8(b)) measure does not indicate any real convergence due to the lack of information in the training set to put evolutionary pressure on the population. The result of this study could show us the learning, adaptation and robustness capability in the population. Moreover, we could also see how the networks generalize to the novel inputs.

4.1.6 Robustness

We have calculated two different distributions for the fitness and the generalization score for all the tasks. First, the probability that the the fitness or generalization score of the population during each generation in the steady state is higher than its mean value throughout the steady state $p(f \geq \bar{f}; K)$ and second, $p(g \geq \bar{g}; K)$. These two probabilities are a measure of robustness of the computation with respect to learning and generalization. The distributions are shown in Figures 4.9 for the even-odd task, Figure 4.10 for the full-adder task, and Figure 4.11 for the R85 task. Note that the population is only seeing half of the input patterns, yet it generalizes the computation to the novel inputs. We see that the distribution is very sharp near K_c. This means $T = 4$ provides enough information to the networks during the evolution to achieve a convergence in connectivity. For R85, the distribution looks a little wider. This agrees with the earlier observation of weaker convergence in the connectivity during the learning of this task. This robustness measure, however, does not directly describe the quality of the generalization.

We have also calculated the probability distribution that the population achieves perfect fitness and generalization score, namely, $p(f = 1; K)$ and $p(g = 1; K)$. We observe the same sharp peak around K_c. This is a strong evidence that both the learning and the generalization capability of the population is maximal exactly at, or very close to, K_c. Furthermore, K_c is certainly lower than the "edge of chaos," i.e., $\langle K \rangle$=2. Rohlf *et al.* [75] introduced the "edge of stability" $K_c = 1.875$ for RBNs where the damage spreading is independent of system size. The authors argued that this critical connectivity maximizes the robustness in the networks against perturbations. Our experiments also give solid evidence that the the computation at the "edge of stability" is maximally robust against perturbations.

4.1.7 Verifying that $K_c < 2.0$

The observation of $K_c < 2.0$ (Table 4.4) does not exactly match the hypothesis that complex computation is enhanced in the $K_c = 2$ region. Also, if the value is biased by our finite size system, we would expect K_c to be higher than 2. To verify this fact, we evolve networks of size $N = 100$ with the even-odd and the full-adder tasks with eight training patterns and an initial connectivity of $\langle K \rangle = 3.0$. The selection of $\langle K \rangle = 3.0$ is due to its symmetry to $\langle K \rangle = 1.0$ with respect to $\langle K \rangle = 2.0$ the connectivity of the complex dynamical regime. We combine the result of the evolution from the initial $\langle K \rangle = 1.0$ and $\langle K \rangle = 3.0$ and create the distribution of $\langle K \rangle$ as before. The location of the peak of the distribution is $K_c = 1.87$ for the even-odd task and $K_c = 1.9$ for the full-adder task. This implies that for complex computations, K_c is less than 2 for large systems. The

task	T	a	b	c
FA	1	-0.1092 ± 0.4772	0.6606 ± 0.8084	5.3930 ± 1.6510
EO	1	-0.0654 ± 0.4592	0.8244 ± 1.3576	5.3750 ± 2.6230
R85	1	-0.0002 ± 0.0029	1.9380 ± 2.8160	4.0680 ± 0.9620
FA	8	44.6200 ± 24.1300	-1.2570 ± 0.3069	1.8090 ± 0.2130
EO	8	85.4900 ± 219.9100	-1.8660 ± 1.5105	1.9800 ± 0.2970
R85	8	$-1.981E - 12 \pm 2.5208E - 10$	5.6320 ± 27.5080	2.928 ± 0.4620

Table 4.2: Parameters of the finite size scaling using final population mean statistics with 95% confidence interval bounds. The estimation is done with the nonlinear least square method. The estimated parameters are for a power-law equation of the form $ax^b + c$.

result is independent of the initial connectivity of the population. We have to be careful, however, that if the initial connectivity is much higher than K_c for larger system sizes, the evolution should continue for longer than 30,000 generations for the systems to converge to K_c.

4.1.8 Discussion

In this section, we showed that the the evolution of a network population with a computation-dependent fitness will lead to a critical connectivity K_c in which the networks operate in a complex dynamical regime. We showed that these complex dynamics optimize robust learning and generalization in a noisy environment. The exact value of K_c depends on the amount of information that is available to the population via the training samples. For very complex tasks, such as the even-odd and the full-adder tasks, for which the perfect ($f = 1$, $g = 1$) computation requires a great deal of stability and reliability, $K_c \approx 1.87$, "the edge of stability," to achieve maximum robustness against perturbations.

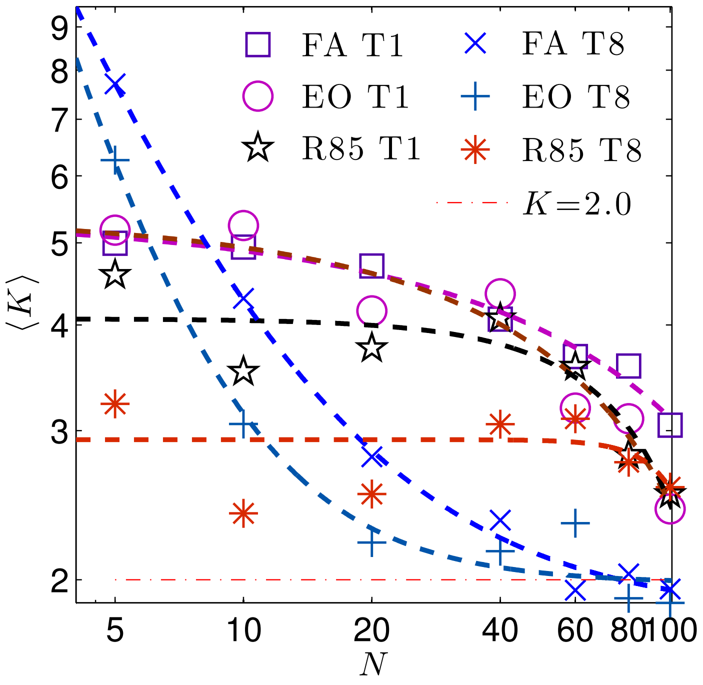
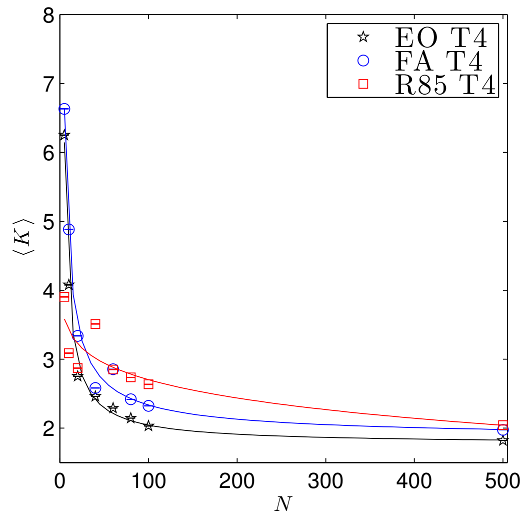
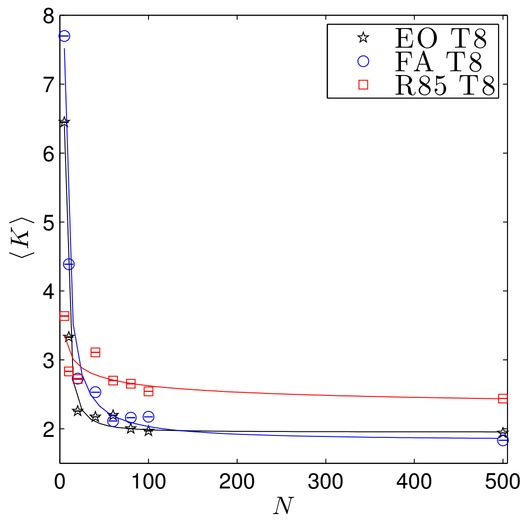


Figure 4.4: Finite size scaling of $\langle K \rangle$ as a function of N for the three tasks (full-adder, even-odd, and rule 85) and the training size using final population statistics. Points represent the data of the evolved networks, lines represent the fits. The finite size scaling for $\langle K \rangle$ shows that it scales with a power-law as a function of the system size N. The dashed lines represent the power-law fit of the form $ax^b + c$. We used a N/N_{max} weighting for the data to emphasize the larger network sizes in estimation.



(a) Finite size scaling for $T = 4$.



(b) Finite size scaling for $T = 8$.

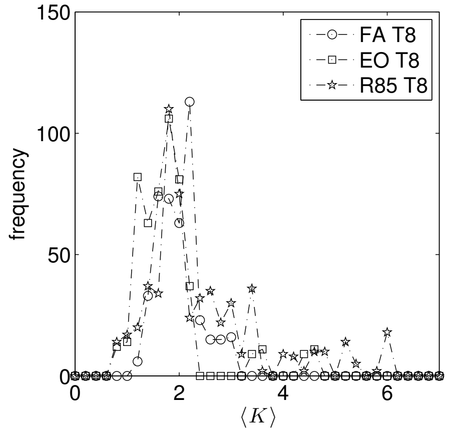
Figure 4.5: Finite size scaling using Gaussian mixture model of population statistics during Evolutionary Steady State (ESS). We used a N/N_{max} weighting for the data to emphasize the larger network sizes in estimation.

task	T	a	b	c
FA	4	23.25 ± 29.97	-1.005 ± 0.7401	2.078 ± 0.69
EO	4	16.56 ± 15.19	-0.8003 ± 0.5521	1.517 ± 0.851
R85	4	-0.1497 ± 2.4687	0.4834 ± 2.8056	3.713 ± 5.264
FA	8	33.13 ± 15.85	-1.173 ± 0.2713	2.017 ± 0.19
EO	8	31.62 ± 35.88	-1.228 ± 0.6435	1.855 ± 0.352
R85	8	-0.01119 ± 0.16419	0.9383 ± 2.8967	3.163 ± 1.36

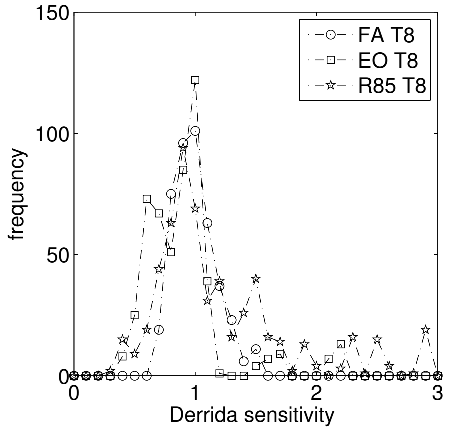
Table 4.3: Parameters of the finite size scaling using Gaussian mixture model of population statistics during the Evolutionary Steady State (ESS) with 95% confidence interval bounds. The estimation is done with the nonlinear least square method. The estimated parameters are for a power-law equation of the form ax^b+c.

task	$T = 1$	$T = 4$	$T = 8$
FA	1.37	1.80	1.90
EO	1.68	1.64	1.87
R85	2.02	2.17	2.25

Table 4.4: Empirical peak of the $\langle K \rangle$ distribution during ESS for $N = 100$. For $T = 8$ for the even-odd and the full-adder tasks, we have re-adjusted the values based on the experimental data with an initial connectivity of $\langle K \rangle = 1.0$ and $\langle K \rangle = 3.0$. The distribution is created by binning the $\langle K \rangle$ data with bin size of 0.01.

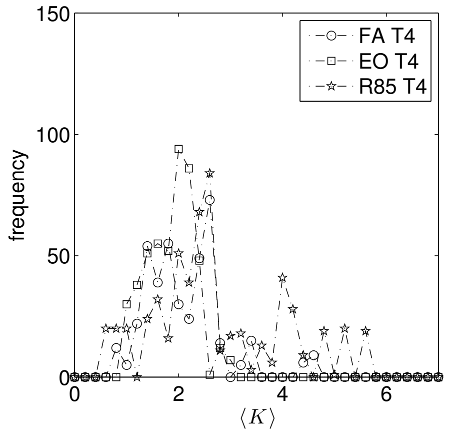


(a)

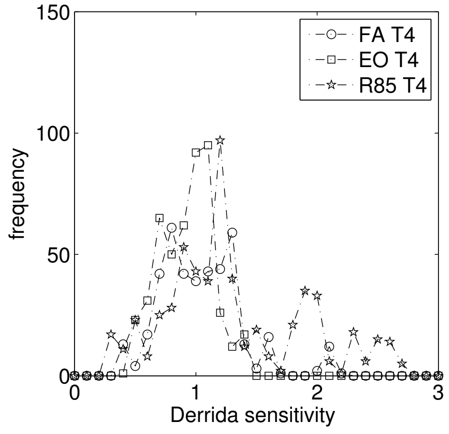


(b)

Figure 4.6: The final $\langle K \rangle$ distribution and the Derrida sensitivity [83] distribution of $N = 100$ networks with a generalization score of $G \geq 0.8$. The provided information in the input patterns is sufficient to push the networks toward critical connectivity. The peak of the Derrida sensitivity measure at 1 suggests that these networks show critical dynamics as well.

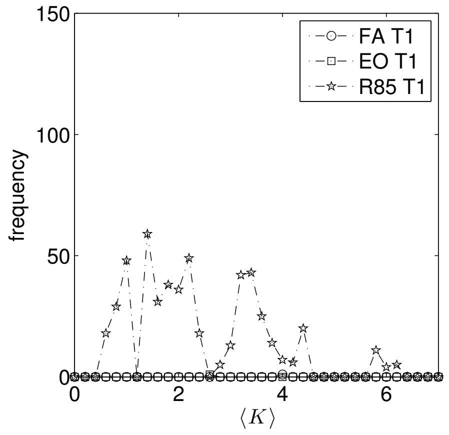


(a)

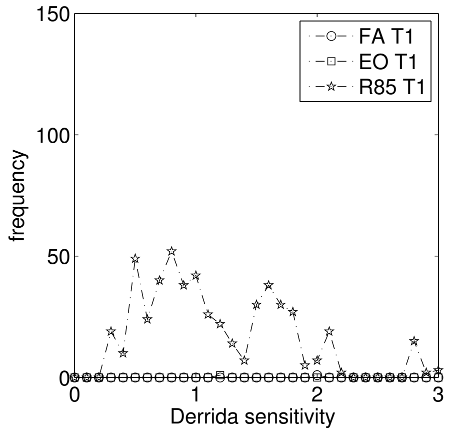


(b)

Figure 4.7: The final $\langle K \rangle$ distribution and the Derrida sensitivity [83] distribution of $N = 100$ networks with a generalization score of $G \geq 0.8$. The provided information in the input patterns is sufficient to push the networks toward critical connectivity. The peak of the Derrida sensitivity measure at 1 suggests that these networks show critical dynamics as well.

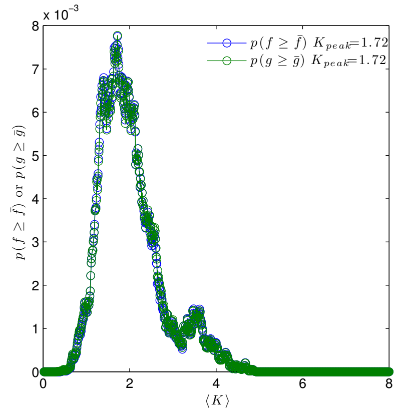


(a)

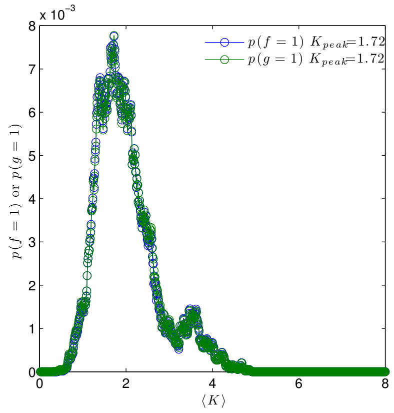


(b)

Figure 4.8: The final $\langle K \rangle$ distribution and the Derrida sensitivity [83] distribution of $N = 100$ networks with a generalization score of $G \geq 0.8$. The networks are unable to successfully compute the full-adder and the even-odd task. The wide distribution of $\langle K \rangle$ and Derrida sensitivity measure suggests there is no convergence in terms of connectivity and dynamics in the evolved population. The information training samples with size $T = 1$ is not to impose evolutionary pressure on the population toward criticality.

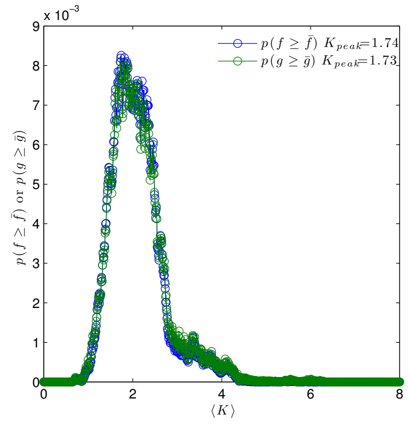


(a) Population robustness.

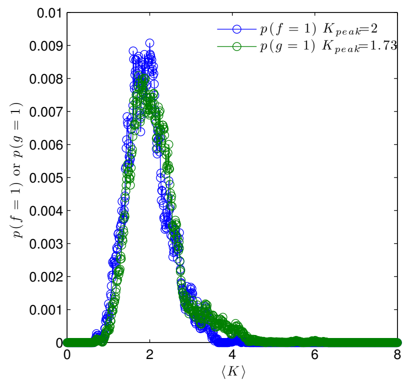


(b) Computation robustness.

Figure 4.9: Probability distribution of network learning and generalization as a function of $\langle K \rangle$ for the even-odd task trained with four patterns.

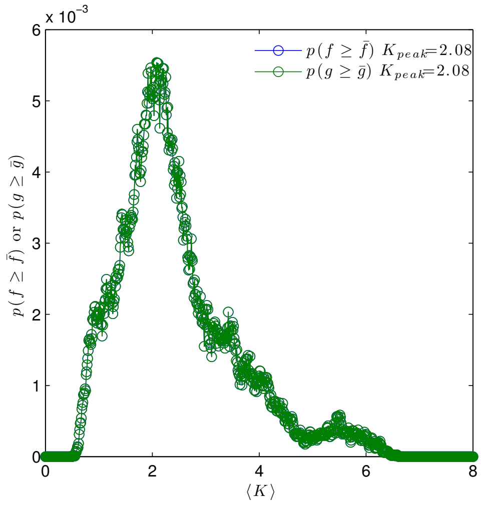


(a) Population robustness.

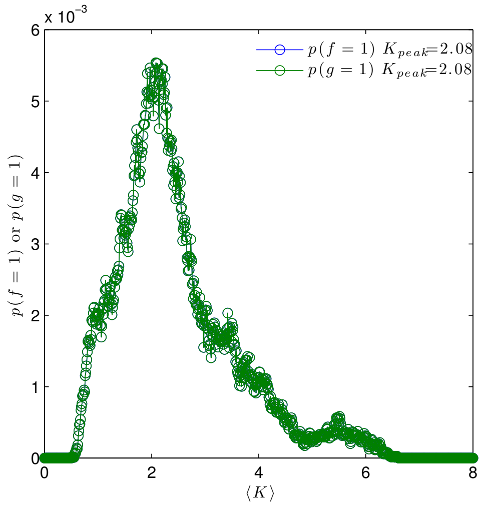


(b) Computation robustness.

Figure 4.10: Probability distribution of network learning and generalization as a function of $\langle K \rangle$ for the full-adder task trained with four patterns.



(a) Population robustness.



(b) Computation robustness.

Figure 4.11: Probability distribution of network learning and generalization as a function of $\langle K \rangle$ for the R85 task trained with four patterns.

4.2 EVOLUTION OF THE DEGREE DISTRIBUTION

4.2.1 Network Evolution Leads to Exponential Degree Distribution

In this section, we investigate the evolution of the degree distribution of the nodes under the new mutation scheme (i.e., link addition and deletion). We will show that the variable $\langle K \rangle$ mutation drives the degree distribution from a Poissonian to an exponential form. We also compare the results with what we have seen in Chapter **??**.

We saw in our earlier experiments that under the $\langle K \rangle$-preserving rewiring mutation scheme, the degree distribution of the networks in the population does not change, i.e., the final population maintains a Poissonian degree distribution. The Poissonian degree distribution is merely the result of random selection during mutation and fitness maximization with respect to the computational tasks.

To study the degree distribution of the final population, we choose all the networks with $\langle K \rangle = 2.0 \pm 0.2$ and average their degree distributions. Figure 4.12 shows the degree distribution, on a log-linear scale, of the population at the end of the evolution. We also included the degree distribution for Poissonian random graphs and exponential random graphs. As we can see, the evolved population shows an exponential degree distribution (indicated by a line in the log-linear plot). A similar result has been reported by Bornholdt and Rohlf [14] during the evolution of networks toward critical connectivities. Moreover, we observe the same degree distribution across the two extreme training sizes $T = 1$ and $T = 8$ for all three tasks. The blue dash-dotted line shows the expected degree distribution of an exponential degree distribution obtained on a 95% confidence interval for $\langle K \rangle = 2.0$. The average connectivity of the final population is not exactly 2.0, however. For the full-adder task and the even-odd task trained with eight input patterns, we see that the slope of the degree distribution is higher than the slope of the $\langle K \rangle = 2.0$ exponential random graphs, despite the fact that the connectivity of these networks is less then 2.0. We would expect the slope to be smaller than exponential random graphs since during evolution average connectivity of the population evolves to $\langle K \rangle < 2.0$. This discrepancy is due to the maximum in-degree limit in our networks, namely $K_{max} = 8$.

4.2.2 Exponential Degree Distribution in Network Growth Model

The emergence of an exponential degree distribution has also been observed in a unbiased network growth model [22]. If we change the perspective, we see that the network growth model also applies to our mutation scheme because the probability of the addition or deletion of links in our network are equal and independent. During the evolutionary steady state, if we assume that we could apply all the link deletions until the networks are depleted completely of their links, and then start adding links randomly, we have effectively approximated the network growth model. As we add links to the networks without bias, we simply connect pairs of nodes. This will initially lead to the creation of the islands of small connected networks that are themselves not connected together. Each of these networks keep growing as additional links will connect unconnected clusters. This process, mathematically and experimentally, results in an exponential degree distribution on each growing island. As we continue adding links, we reach a state at which the subnetworks connect together to form a single connected component that spans all of the nodes in the original network.

This explanation, however, leads to a contradiction with the result from the construction of random graphs by Erdös and Rényi [23]. The scenario that we described is exactly what happens in an Erdös-Rényi (ER) network with one distinction: in the construction of an ER network there is no link depletion stage. ER networks are constructed by adding links to a network in an unbiased fashion. The apparent contradiction between the two scenarios is a result of an implied assumption in the link depletion/restoration process. The implied assumption is that when we connect smaller networks with exponential degree distribution, we get a larger network of exponential degree distribution. Evidently this is not true, otherwise ER networks would have the same exponential distribution.

4.2.3 Exponential Distributions and Dynamical Stability

A second explanation for the emergence of an exponential degree distribution could be related to the maximization of robustness during the steady state [76]. Since the mutation rate in our population is very high, the structure of the ensemble of the networks in the population from one generation to the next can vary profoundly. This structural difference has immense implications in terms of the dynamics of the network and consequently the produced output signals. In order for the networks to resist this variation in the dynamics, the selection process must secure some source of stability for the dynamics of the networks through some invariant signals. In exponential networks, most of the nodes will not have any inputs at all. They may, however, be connected to the rest of the network providing some constant signals during the operation of the network. This is exactly the source of stability in the dynamics. Although the evolved networks are in the complex regime and the dynamics may jump from one attractor to another as a result of fairly simple mutation steps, the existence of a large number of nodes with constant signals makes the network resilient to random structural perturbations. This may also be solid evidence for higher resilience in exponential random graphs as opposed to ER graphs with Poissonian degree distribution. To confirm this explanation, we have to show that indeed most of the nodes are connected to the network with respect to their output links. We verify this by generating the out-degree distribution of the ensemble of the networks in the population. Figure 4.13 illustrates the out-degree distribution of the three tasks trained with $T = 1$ and $T = 8$. The ER graph out-degree distribution is included for comparison. We observe two things: (1) the out-degree distribution does not show any change from the null model and (2) the out-degree distribution has a Poissonian form with few nodes that are not connected to other nodes relative to the network size. Thus, this is concrete evidence linking the complex interplay of our dynamics-dependent objective function and the random selection. The same trends in the evolution of the in-degree and the out-degree distribution is observed in [76].

4.2.4 Exponential Degree Distribution and Task Complexity

In Section 4.1.6, we showed the emergence of robustness in the population as a result of evolution dynamics, however, seems to be independent of the complexity of the computational task. Bornholdt and Sneppen [15] name robustness as a principle in evolutionary processes. Robustness in this study is measured by the probability of the individuals having a fitness greater or equal to the average fitness during the ESS (mean fitness of the individuals in the population averaged over the last 15,000 generations). For all tasks, we observed this probability distribution to show a sharp peak near K_c. In addition, our study of the evolution using different system sizes also shows the emergence of the same in-degree and out-degree distributions in the networks. Even the networks that evolve to solve the R85 task show the same in-degree and out-degree distribution that results in the same level of robustness. The same effect is also observed in training with full-adder and even-odd task with only one training example. In R85 task, the amount of information provided to the population is minimal. However, despite the minimal imposed constraint on the evolution of the population, the same level of robustness evolves and develops in the networks. Thus, we can conclude that the emergence of robustness in networks that compute under unbiased evolution is completely independent of the task complexity and constraints, the training samples, and the

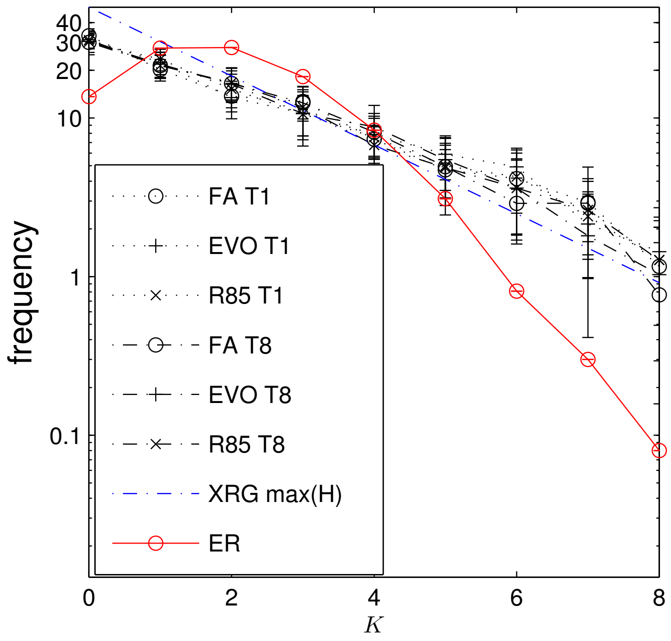


Figure 4.12: The in-degree distribution of the networks after the learning process changes from a Poissonian to an exponential distribution for all three computational tasks. All networks start from a random topology using the Erdös-Rényi model (ER). XRG corresponds to the maximum entropy exponential random graphs.

network size N.

4.2.5 Degree Distribution and $\langle K \rangle$-Preserving Mutation

One unanswered question is "Why under the $\langle K \rangle$-preserving mutation do we not observe the rise of an exponential degree distribution?" After all, even when $\langle K \rangle$ does not change, the networks can still use the stability of the exponential networks to compute reliably in the face of structural disturbances. Is there something inherent in fixing $\langle K \rangle$ that does not let the degree distribution evolve? The number of links added to or deleted from the networks is, on average, equal during the

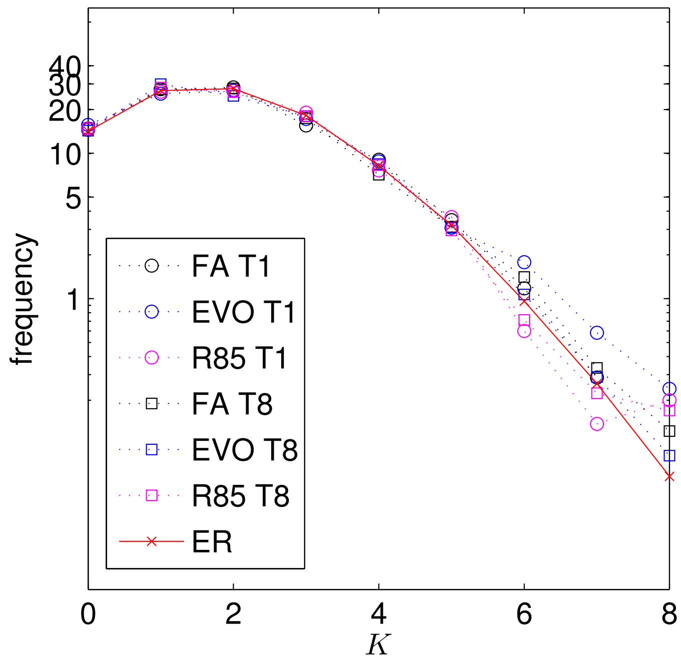


Figure 4.13: The out-degree distribution of the networks after the learning process changes from a Poissonian to an exponential distribution for all three computational tasks. All networks start from a random topology using the Erdös-Rényi model (ER).

steady state. Therefore, one would expect to see each mutation step caused by rewiring (changing a source or a destination of a link) done by two mutation steps in addition and deletion of the links. If the exponential degree distribution is preferred by the selection during link addition and deletion over the generations, then the rewiring scheme should also lead to the same result. However, in the study of evolution under fixed $\langle K \rangle$ (Chapter **??**), our focus was to study the learning probability. The calculations in learning probability do not let the GA reach the steady state, because we stop the training as soon as we have a fitness of one or if we reached the maximum of 2,000 generations. Thus, the reason we did not see the degree distribution evolution is that the GA stopped too early.

To investigate this issue, we evolve two populations of networks with size $N =$ 20 (to be comparable to earlier fixed $\langle K \rangle$ experiments). The first population is mutated using the rewiring scheme, the second population is mutated using the addition and deletion of links, but we perform this in a symmetric way to keep $\langle K \rangle$ fixed. That is, we add the same number of links to the network that we delete from the network in every mutation step. We evolve both populations for 5,000 generations. The population that used the rewiring scheme does not show any change in degree distribution. The second population that uses the symmetric link addition and deletion does, however, show an exponential degree distribution. The shape of the Poissonian distribution in the first population is slightly deformed. We conclude that that the evolution might simply be taking more time. We repeat this experiment, but this time for 20,000 generations. We still see the same result in the evolved population. The population that is mutated using the rewiring scheme does not show any significant change in the degree distribution. On the other hand, the population that is mutated using the addition and deletion of the link, even with preserving $\langle K \rangle$, still evolves toward an exponential degree distribution.

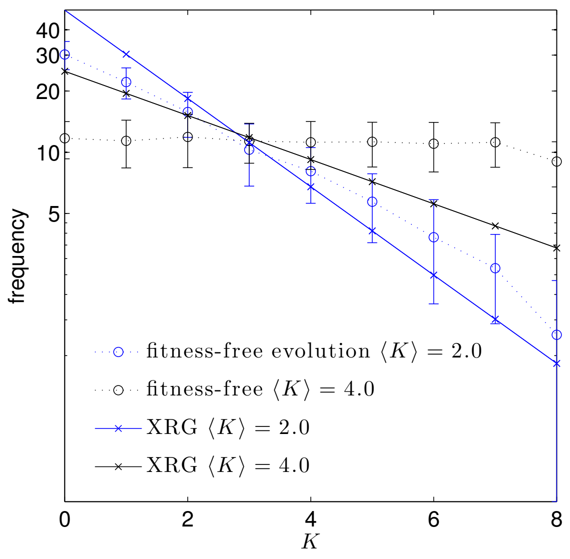


Figure 4.14: The in-degree distribution of the networks after fitness-free mutation changes from a Poissonian to an exponential distribution for all three computational tasks. All networks start from a random topology using the Erdös-Rényi model (ER). Evolution of the exponential degree distribution is independent of an evolutionary pressure.

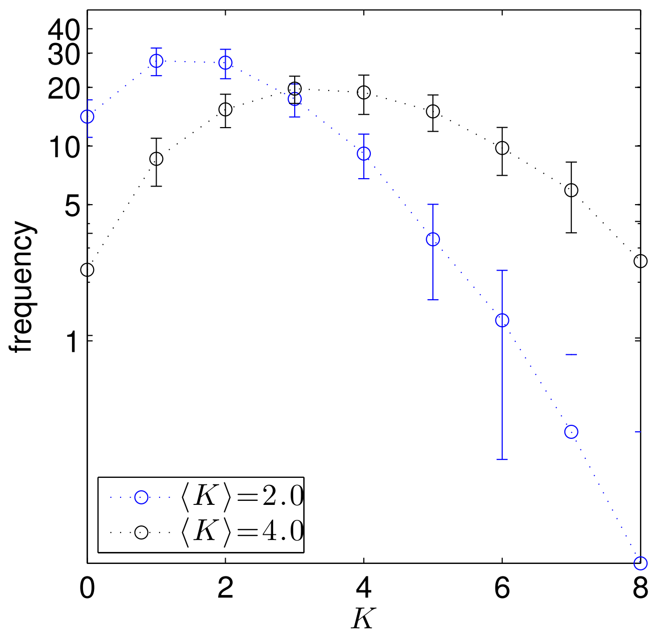


Figure 4.15: The out-degree distribution of the networks after fitness-free mutation remains a Poissonnian distribution. The out-degree does not evolve with or without a fitness function.

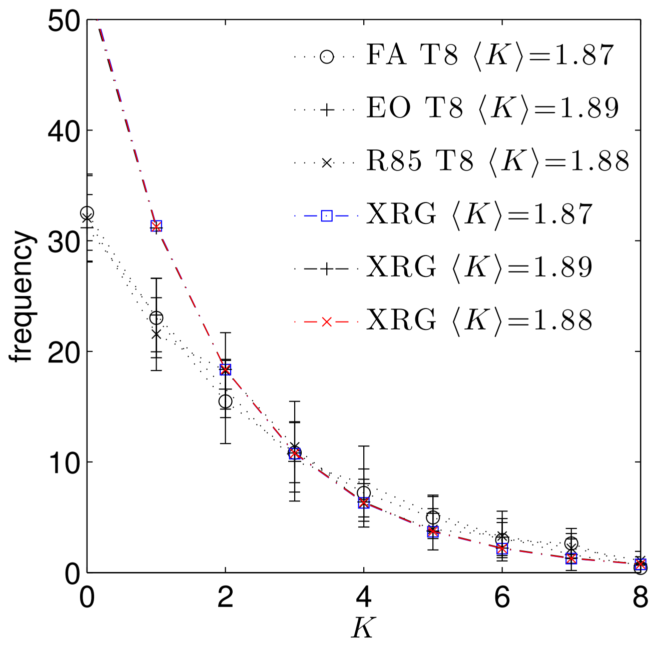


Figure 4.16: The in-degree distribution of the networks in an evolved population in comparison to maximum entropy exponential graphs (XRG) with the same connectivity $\langle K\rangle$. The evolution of the exponential in-degree distribution is independent of the task. The mismatch between the theoretical estimated in-degree distribution for XRG networks and evolved networks is the result of upper bound on the node in-degree $K_{max} = 8$.

4.2.6 Exponential Degree Distribution and Fitness-Free Evolution

We revisit our experiments with evolving populations with no fitness function. We used this experiment before in section 4.1.3 as a null model to verify the hypothesis of the existence of K_c. Here, we use the same trick to see if the degree distribution of the networks change if we do not subject the population to a fitness function, i.e., no computational task. The selection will not have any leverage to discriminate against a subset of the population. Therefore, we have a population that freely evolves in a random fashion. The mutation in the individuals follow the same addition and deletion of the links and the average connectivity $\langle K \rangle$ is free to change as well. As before, we create the degree distribution of the networks over the entire final population (Figure 4.14). Although the selection is unaware of the dynamics in the networks, the in-degree distribution of evolved populations is of an exponential form. We have also plotted the in-degree distribution for $\langle K \rangle = 2.0$ to comparison to the previous results. The in-degree distribution for $\langle K \rangle = 4.0$ is included since the average connectivity of the population revolves around $\langle K \rangle = 4.0$ in a binomial form (see Section 4.1.3). Thus, we conclude that the evolution of the exponential form in the in-degree distribution is more related to the mutation scheme than to the selection process.

Figure 4.15 illustrates the out-degree of the population after a fitness-free evolution. We have plotted the out-degree distribution of both $\langle K \rangle = 2.0$ and $\langle K \rangle = 4.0$ networks for comparison. The out-degree distribution does not show any evolution and continues to resemble a Poissonian. It is not clear why the out-degree distribution is unaffected under the mutation of the links. But the conclusion of this observation is, as before, that the evolution of the degree distributions somehow only depends on the mutation scheme and not the selection, and is therefore independent of the computation performed by the network.

4.2.7 Exponential Distributions in Statistical Physics

The exponential distribution is a topic that is exhaustively studied in statistical physics and thermodynamics to describe the state of a system using its microstates. More specifically, one would like to study the average energy of a system that is in thermodynamic equilibrium with its environment. The environment in this context has the characteristic of a heat bath. That is, its temperature does not change regardless of the amount of energy that it absorbs or dissipates to the objects that it comes into contact with. When a system comes to contact with the heat bath, the energy starts to flow from the system in the direction of the lower temperature. By definition, this flow of energy does not change the temperature of the heat bath, but the temperature of the object in contact with the heat bath will increase or decrease until it is equal to the temperature of the heat bath. From that moment on, the energy flow between the object and heat bath will be balanced. That is, the heat bath and the object will continue to exchange energy, but in a manner that keeps the net transferred energy zero. In other words, the same amount of energy that goes from the object to the heat bath (the environment) will be absorbed by the object from the heat bath and thus the temperature of the object remains constant. This constant energy is the average energy of the system over time. The system, however, continues to evolve and exchange energy with the environment and therefore its exact energy will change from time to time. The second law of thermodynamics states that the evolution of the system in thermodynamic equilibrium is such that the probability distribution of the system over its microstates assumes maximum entropy over time. This probability distribution for the microstates of the system dictates the most likely state of the system because of the fact that its entropy is maximum. This is due to the fact that the number of states in this distribution is maximum and therefore most likely covers every state in which the system is likely to be found. However, this distribution of the states has to satisfy one constraint, namely the average of the expectation value of the distribution should be equal to the average energy of the system.

The same argument can be applied to the networks as they evolve to compute the desired tasks. Park and Newman [68] showed how to apply the maximum entropy principle to derive properties of complex networks. Recall the fitness function in our study is $1 - E$, where E is the error of the network. The evolution of the system is such that the error becomes zero over time and from then on the system will be in a thermodynamic equilibrium with its objective function (the environment) [92]. This is what we before called the evolutionary steady state (ESS). During this steady state, the average desired observable ($\langle K \rangle$ in our study) does not change. The number of links added to the system is equal to the number of links deleted from the system over time. We can therefore apply the maximum entropy argument and calculate the expected degree distribution that maximizes the entropy of the microstates (in-degrees of the nodes) in a way that satisfies the average connectivity in the system K_c.

4.2.8 Discussion

In this section, we observed the evolution of exponential degree distribution during the variable $\langle K \rangle$ mutation. We proposed several explanation for this observation such as the selection for stability and the maximum entropy principle, and we supported the arguments by comparing our results to various null models. Our study introduced new questions that are yet to be answered. Why does the rewiring mutation not allow a freer evolution in the degree distribution? What is inherently so different between changing the terminals of the links one by one or both at the same time? Why is the convergence of the average connectivity to K_c depends on the task complexity, but the degree distribution does not? These remain as open questions that are beyond the scope of this thesis.

4.3 EVOLUTION OF THE STRUCTURAL PROPERTIES

We take the study of topology further by extending the investigation of the evolution of the degree distribution to the graph-theoretical properties of the evolved networks. We study four graph measures that seem to have the most interesting behavior: the eccentricity, the characteristic path length, the participation coefficient, and the betweenness centrality [78]. We will see how the evolution of the networks lead the population to reach a compromise between the Poissonian random graphs (ER) and the eXponential Random Graphs (XRG). Furthermore, through the study of structural properties, we show how K_c relates to maximum robustness and optimal adaptation and learning in the evolutionary dynamics of the population. We will also explain the relationship between our findings and Fisher's fundamental theorem of natural selection [24].

4.3.1 Graph Measures

We begin by giving a brief overview of each of the graph measures under study. Eccentricity and characteristic path length are both measures of network functional integration [78], meaning how inter-related the nodes are in the network. Eccentricity is the maximum shortest path length between a node and any other node in the network. The characteristic path length is the average distance between any two nodes in the network. The participation coefficient of a node in the graph is the measure of centrality in the network. The participation coefficient of a node i with k_i links is calculated in [78] as follows:

$$y_i = 1 - \sum_{m \in M} \left(\frac{k_i(m)}{k_i} \right)^2, \tag{4.11}$$

where M is the set of modules in the network [65, 66] and $k_i(m)$ is the number of links between node i and the nodes in the module m. The higher the participation, the more inter-modular dependency (coupling), while lower values show a higher cohesion of modules, which suggests high intra-module integration and low inter-module coupling. The betweenness centrality is a measure of centrality for a node i in the network. It is calculated in [78] as follows:

$$b_i = \frac{1}{(n-1)(n-2)} \sum_{\substack{h,j \in N \\ h \neq j, h \neq i, j \neq i}} \frac{\rho_{hj}(i)}{\rho_{hj}}. \tag{4.12}$$

Here N is the set of nodes in the network and $n = |N|$. ρ_{hj} is the number of shortest paths between h and j and $\rho_{hj}(i)$ is the number of shortest paths between h and j that passes through i. This measure is an indicator of the importance of the function of a node for the rest of the nodes in the network. We study the node related measures, the betweenness centrality and the participation coefficient by taking the average of measure over all the nodes in the network.

4.3.2 Evolution of Graph Measures

We now study how these graph measures vary with average connectivity in the null networks, the ER and the XRG, and in the evolved networks. Figure 4.17 shows the average eccentricity of the ER and XRG network ensembles as the function of $\langle K \rangle$. The ER networks show a very sharp rise of the eccentricity in the region $1 \leq K \leq 2.0$. For $\langle K \rangle > 2.0$, the rise slows down and reaches a peak at a value slightly higher than 2.0 followed by a exponential decrease for $\langle K \rangle > 2.0$. The XRG graphs show a milder increase than the ER networks in the value of eccentricity until $K \approx 2.0$ and a very slow decrease for $\langle K \rangle > 2.0$. In an ER network, therefore, near $\langle K \rangle = 2.0$, the communication between nodes on average slows down in comparison to the XRG networks. This is because the signal from a node has to travel through many hops to reach the other nodes. XRG networks are therefore better for global communication in the network. We see that for both extreme cases with training sample size $T = 1$ and $T = 8$, the evolved population shows a decrease in the eccentricity. The maximum decrease from the ER model is around $\langle K \rangle = 2.0$. The standard deviation of the eccentricity measure is shown in Figure 4.18. We see that for both null ensembles and the evolved populations, the deviation is maximized near $\langle K \rangle = 2.0$ with the maximum value belonging to the population trained using $T = 1$. This suggests that not only the evolution is reducing the eccentricity, but it is doing that in a very selective way by maintaining a diverse population.

The betweenness centrality of the null ensembles as well as the evolved populations are presented in Figure 4.19. Betweenness centrality of both ER and XRG ensembles show very similar behavior to eccentricity in that they increase with $\langle K \rangle$ up to $K \approx 2.0$. As before, the ER betweenness shows a sharp decrease for $\langle K \rangle > 2.0$ in comparison to XRG networks. In XRG networks for $\langle K \rangle > 2.0$, the betweenness is independent of the connectivity. We see for both evolved populations the betweenness is almost in the middle between XRG and ER. This suggests that the networks maintain a balanced betweenness in comparison to ER and XRG. The reduced betweenness is explained by the evolution of the exponential distribution. In exponential degree distributions, a higher portion of nodes have no inputs in comparison to Poisson distributions, therefore, the nodes will not be part of any shortest paths in the network. The standard deviation of the betweenness in the networks shows a very sharp peak around $K \approx 2.0$ for ER and evolved networks. The XRG networks have maximum standard deviation in betweenness near $K \approx 2.0$, but the peak is not as sharp as for ER networks (Figure 4.20). We see that the evolution of the population drives the networks to a state of balancing trade-offs between ER and XRG networks.

The average participation coefficient in the networks for the XRG networks shows an exponential decrease as the function of $\langle K \rangle$ (Figure 4.21). For the ER graphs on the other hand, we see a decreasing trend up to $K \approx 2.0$ and then a slow increase for $\langle K \rangle > 2.0$. The evolved population shows a slightly higher values of participation than the ER graphs. This indicates the evolution of higher inter-modular communication in the networks in comparison to the ER graphs. The standard deviation of the participation coefficient is a decreasing function of connectivity in ER and XRG graphs (Figure 4.22). We see, however, that in the $T = 8$ population, this value shows a maximum at $\langle K \rangle = 2.0$.

Figure 4.23 illustrates the average characteristic path length in the networks under study. Both the ER and XRG graphs show a peak around $K \approx 2.0$. The characteristic path length for ER graphs is about 1.5 times larger than the XRG at the peak. The evolved networks show a fair balance between ER and XRG values for this measure. The standard deviation of the characteristic path length in the population is maximum in $1.0 < K < 2.0$ interval followed by a sharp decrease for for $\langle K \rangle > K_{peak}$. In general, the deviation in the evolved population is lower than for ER and XRG networks, which means the evolution narrows down the diversity on the population with respect to this measure. Thus, the balance right in the middle of the ER and XRG graphs seems to be most desirable for computation.

4.3.3 Discussion

By studying and comparing the functional integration and the node centrality measures in the networks, we gained a deeper insight into the evolution of the networks. We see that both the ER and XRG graphs show a diminishing return in their integration and centrality at $K \approx 2.0$. Moreover, the topological diversity of both ER and XRG networks becomes maximal in the same connectivity region. We evolved population of networks to satisfy robust computation. The population achieves this desired property by compromising the topological properties between the ER and the XRG graphs. This trade-off allows the population to benefit from the best properties of two inherently different network structures. Finally, the inherent maximal diversity in the topology of the network near the critical region of the connectivity optimizes the adaptation and robustness in the population.

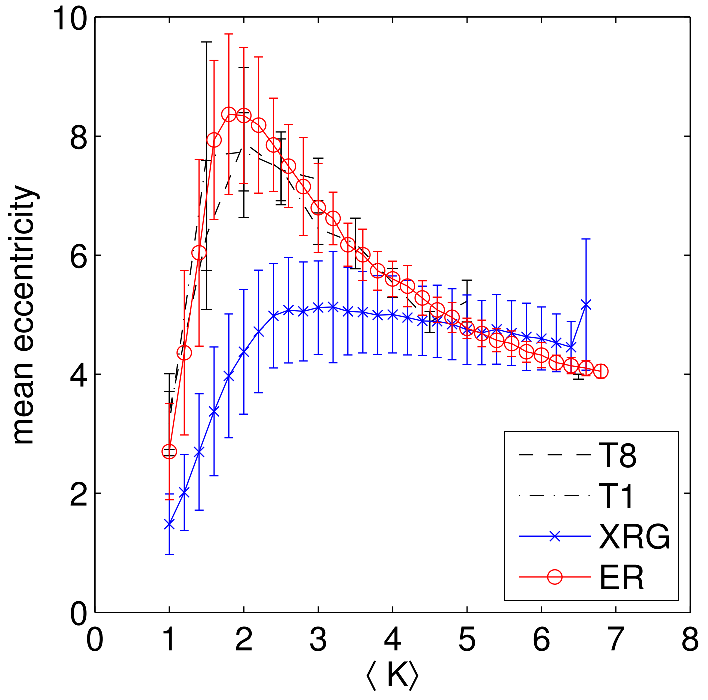


Figure 4.17: The average eccentricity of the two null model ER and XRG graphs and the evolved populations. Evolved networks maintain a trade-off between ER and XRG graphs, suggesting a balance in functional interdependency between the nodes in the network.

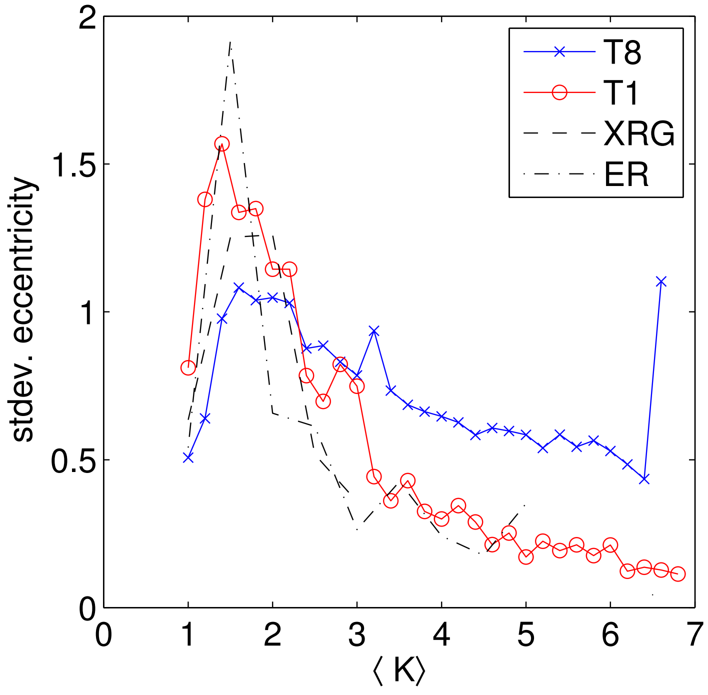


Figure 4.18: The standard deviation of the eccentricity of the two null model ER and XRG graphs and the evolved populations. Maximal diversity in eccentricity in both ER and XRG graphs at $\langle K \rangle = 2.0$ makes networks in this region more evolvable. The Fisher information maximization [27] as a result of selection increases the diversity in the evolved population in comparison to ER and XRG graphs.

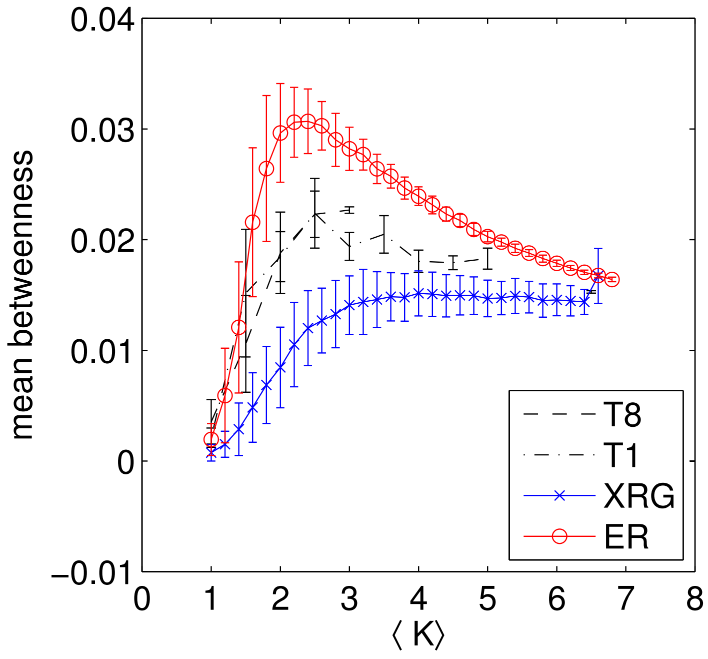


Figure 4.19: The betweenness centrality of the evolved population evolves to a middle value between the ER and the XRG networks. This suggests a trade-off in functional interdependence of the nodes in the networks that maximizes the ability to perform complex computation.

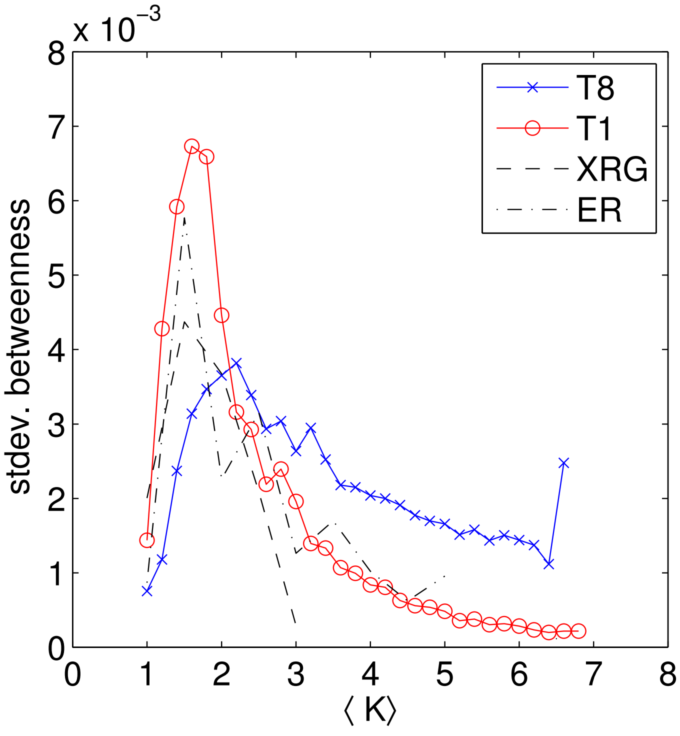


Figure 4.20: Maximum diversity in betweenness centrality at $\langle K \rangle = 2.0$ suggests that this region is the best for the adaptation of the population.

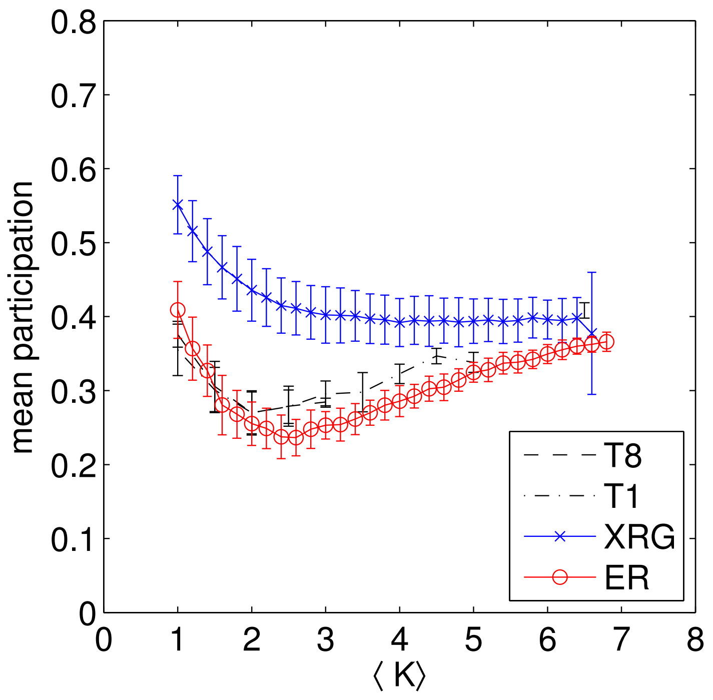


Figure 4.21: The average participation coefficient of the two null model ER and XRG graphs and the evolved populations. Although the degree distribution of the evolved population becomes exponential, the networks maintain participation characteristics of ER graphs, suggesting that the population take advantage of the best features in ER and XRG to optimize adaptation and computation.

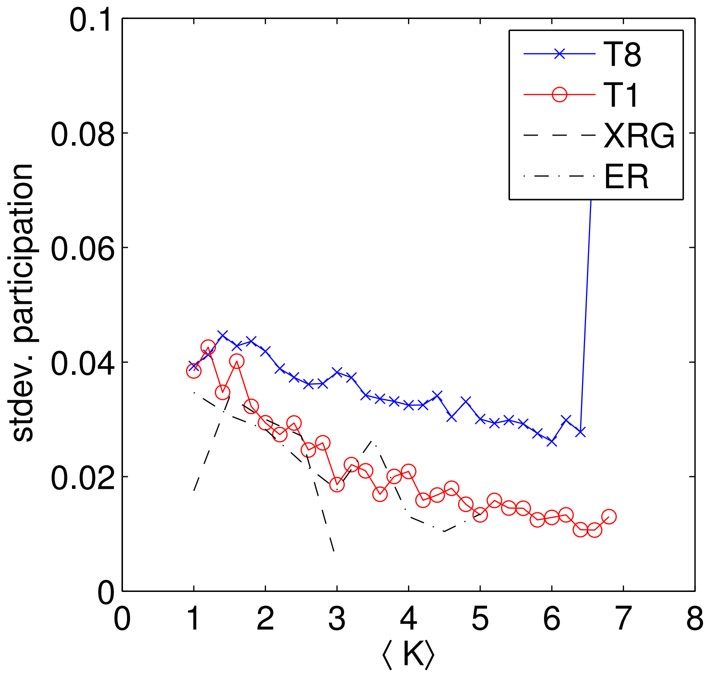


Figure 4.22: Maximum diversity in the ER and XRG network ensembles for lower connectivity is a possible source of convergence of the K_c to values lower than 2.0.

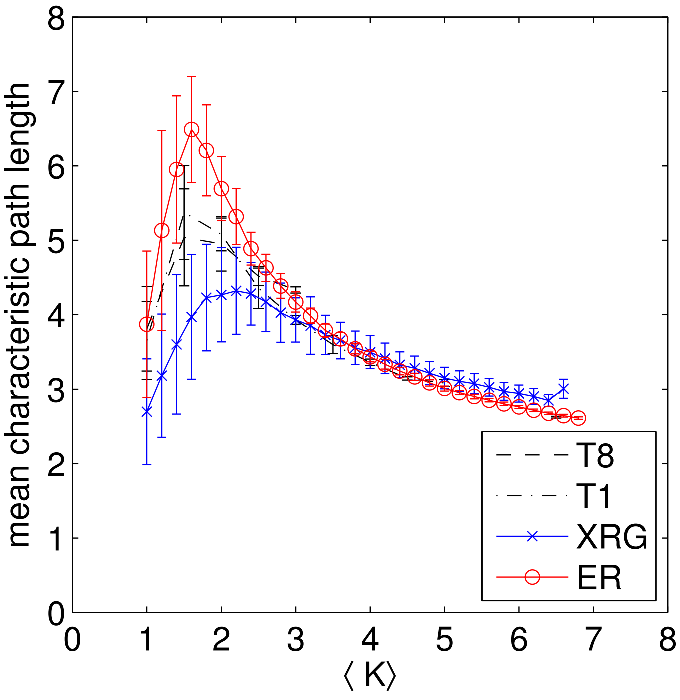


Figure 4.23: The average characteristic path length of the two null model ER and XRG graphs and the evolved populations. The population maintain a middle ground between ER and XRG networks with respect to characteristic path length suggesting that the population benefits from the features in topological structures of both ER and XRG networks.

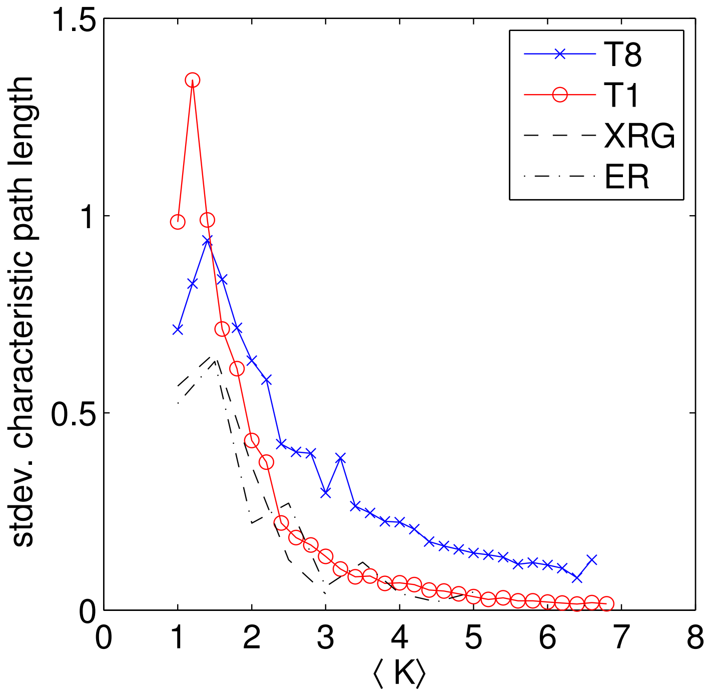


Figure 4.24: Maximum diversity for lower connectivity suggests networks with lower connectivity are well suited for robust computation.

Fisher's fundamental theorem of natural selection states that the rate of improvement in the population is equal to its genetic variance [27, 28, 72, 73]. The evolution of K_c toward 2.0 causes the population to maintain maximum diversity during the steady state. As a result, the rate of improvement in the population is optimized and the population maintains a high robustness in the face of an extremely noisy environment. The selection will also push the networks toward a balance between both ER and XRG graphs to make use of the best of the properties between the two types of graphs.

5

CONCLUSIONS

We have presented a series of arguments and evidence pertaining to the influence of the topology and connectivity of the random networks in computation. To the best of our knowledge, this is the first study that extends the RBN model to perform specific computational tasks. We investigated the learning and generalization capabilities of feedforward networks and RBNs. Furthermore we showed that random networks are capable of robust computation against perturbations. We explored the effects of the mutation and selection on the topology and connectivity of the network and explained the evolution of critical connectivity and exponential degree distribution in the networks. In addition, we described the implication of $K_c = 2.0$ for the robustness using the inherent properties of both ER and XRG random graphs.

These findings have important implications for emerging computational models. The self-assembly of nano-wires and nanoscale components will lead to structures that are not entirely controllable. We showed that through the control of observable macrostates, such as the average connectivity and the distribution of the connectivity between components of a self-assembled network, we are able to produce devices that are optimal for making building blocks for computation, e.g., simple logic gates. This computation is programmable through various self-assembly and evolutionary techniques. Furthermore, the computation in these classes of devices is extremely robust against failures and other perturbations.

We exhaustively studied the characteristic behavior of Random Automata Networks (RAN) from an evolutionary perspective in a learning and generalization context. We based this study on the premise by Teuscher *et al.* [87] that future nanoelectronics fabrication will use self-assembly techniques that lead to unstructured circuits. In self-assembly, both the fabrication process and the final product are prone to failures and defects. RAN information processing in an evolutionary context present us with a unique opportunity to study the computational properties of RAN in noisy environments.

We started by following the footsteps of Patarnello and Carnevali [70] and Van den Broeck and Kawai [93] by replicating learning and generalization in randomly constructed feedforward circuits. We then extended our work to RBN as a restricted case of RAN. By modeling these unstructured circuits using RAN, we showed that these systems are capable of learning and generalization in a traditional task-solving context. However, the existence of recurrency in the networks leads to a complexity catastrophe [47]. The explosion of the solution space hinders the exploration of this space even by the metaheuristics algorithms, and as a result, RBN do not show the same level of learning that feedforward networks show.

In addition to learning and generalization, we studied the adaptation and the robustness in RBNs. We discovered that RBNs — that evolve to solve 3-input Boolean tasks — show maximum balance between computation and robustness to perturbations when the average connectivity in the network is near a critical connectivity $K_c \approx 1.87$ for large systems. This finding conforms with the postulation of the "edge of stability" by Rohlf *et al.* [75]. Moreover, while the connectivity in the networks evolves toward K_c during the evolutionary steady state, the in-degree distribution drifts away from a Poissonian form and becomes exponential. Throughout this process, various graph topological measures, such as the participation coefficient, the eccentricity, the characteristic path length, and the betweenness centrality find a compromised values between the values of the same measures for Poissonian and exponential random graphs. We conjecture that the existence of maximal topological diversity in the population near $K \approx 2$

is a possible source for maintaining genetic diversity in the population and hence we conjecture that maximal learning, adaptation, and robustness occurs for this connectivity level [24, 27, 73].

The emergence of a critical connectivity $K_c < 2.0$ during the steady state speaks for a greater trade-off in the heart of an evolutionary process: exploration vs. exploitation. With higher connectivity, the networks realize more functions, whereas the lower connectivity stabilizes the dynamics. In a normative application of an evolutionary process we often adjust the trade offs between exploration and exploitation through explicit parameters. In a descriptive use of evolutionary algorithms (EA) in this study, balancing between these two strategies is at the mercy of the evolutionary process itself. By adjusting the number of links, the EA increases K to find good solutions and then shifts to lower values of K to stabilize the population while maintaining enough evolutionary momentum to correct for sudden disturbances in the population with respect to the objective function. This balance is struck after the learning phase and during the steady state to ensure robust computation in the face of perturbations.

The complete mathematical description of the evolution of degree distribution and the topological properties demand a more in-depth study that is beyond the scope of this thesis. Extending the computation in networks with more than two states will be left for future work. In addition, the effect of asynchronous updating schemes on criticality is still unanswered. We believe our findings in this study have important applications in manufacturing and programming future nanoelectronic devices. We will have more venues to explore and investigate before this approach to electronics can be applied in realistic situations. This work however, marks a milestone in the study of robust learning and computation in unstructured networks.

REFERENCES

[1] Albert, R., and Barabási, A. Statistical mechanics of complex networks. *Rev. Mod. Phys. 74*, 1 (2002), 47–97.

[2] Aleksander, I. Random logic nets: Stability and adaptation. *International Journal of Man-Machine Studies 5* (1973), 115–131.

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

[4] Amari, S. Learning patterns and pattern sequences by self-organizing nets of threshold elements. *Computers, IEEE Transactions on C-21*, 11 (1972), 1197–1206.

[5] Amit, D., Gutfreund, H., and Sompolinsky, H. Spin-glass models of neural networks. *Phys. Rev. A 32*, 2 (1985), 1007–1018.

[6] Amit, D., Gutfreund, H., and Sompolinsky, H. Storing infinite numbers of patterns in a spin-glass model of neural networks. *Phys. Rev. Lett. 55*, 14 (Sep 1985), 1530–1533.

[7] Amit, D., Gutfreund, H., and Sompolinsky, H. Information storage in neural networks with low levels of activity. *Phys. Rev. A 35*, 5 (1987), 2293–2303.

[8] Amit, D., Gutfreund, H., and Sompolinsky, H. Statistical mechanics of neural networks near saturation. *Annals of Physics 173*, 1 (1987), 30–67.

[9] Asanovic, K., Bodik, R., Catanzaro, B. C., Gebis, J. J., Husbands, P., Keutzer, K., Patterson, D. A., Plishker, W. L., Shalf, J., Williams, S. W., and Yelick, K. A. The landscape of parallel computing research: A view from Berkeley. Tech. Rep. UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Dec 2006.

[10] Back, T., Hammel, U., and Schwefel, H.-P. Evolutionary computation: comments on the history and current state. *Evolutionary Computation, IEEE Transactions on 1*, 1 (1997), 3–17.

[11] Bilke, S., and Sjunnesson, F. Stability of the Kauffman model. *Physical Review E 65*, 1 (2001), 16129.

[12] Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM (JACM) 36*, 4 (1989), 929–965.

[13] Bohland, J., and Minai, A. Efficient associative memory using small-world architecture. *Neurocomputing 38* (2001), 489–496.

[14] Bornholdt, S., and Rohlf, T. Topological evolution of dynamical networks: Global criticality from local dynamics. *Phys. Rev. Lett. 84*, 26 (2000), 6114–6117.

[15] Bornholdt, S., and Sneppen, K. Robustness as an evolutionary principle. *Proc. R. Soc. Lond. B 267* (2000), 2281–2286.

[16] Burnham, K. P., and Anderson, D. R. *Model selection and inference: a practical information-theoretic approach*, 2nd ed. Springer-Verlag, New York, NY, 2002.

[17] Cahon, S., Melab, N., and Talbi, E. G. ParadisEO: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics 10*, 3 (2004), 357–380.

[18] Canning, A., and Gardner, E. Partially connected models of neural networks. *J. Phys. A 21* (1988), 3275.

[19] Carnevali, P., and Patarnello, S. Exhaustive thermodynamical analysis of boolean learning networks. *Europhys Lett. 4*, 10 (1987), 1199–1204.

[20] Darabos, C., Tomassini, M., and Giacobini, M. Dynamics of unperturbed and noisy generalized boolean networks. *Journal of theoretical biology 260*, 4 (2009), 531–544.

[21] Derrida, B., and Pomeau, Y. Random networks of automata: A simple annealed approximation. *EPL (Europhysics Letters) 1* (1986), 45.

[22] Dorogovtsev, S. N., and Mendes, J. F. F. *Evolution of Networks: From Biological Nets to the Internet and WWW.* Oxford University Press, New York, NY, 2003.

[23] Erdös, P., and Rényi, A. On random graphs. *Publ. Math. Debrecen 6* (1959), 290–297.

[24] Fisher, R. A. *The genetical theory of natural selection.* The Clarendon Press, Oxford, 1930.

[25] Flyvbjerg, H. An order parameter for networks of automata. *J. Phys. A 21*, 19 (1988), L955.

[26] Fontanari, J. Generalization in a hopfield network. *J. Phys. France 51*, 21 (1990), 2421–2430.

[27] Frank, S. Natural selection maximizes fisher information. *Journal of Evolutionary Biology 22*, 2 (2009), 231–244.

[28] Frank, S., and Slatkin, M. Fisher's fundamental theorem of natural selection. *Trends in Ecology & Evolution 7*, 3 (1992), 92–95.

[29] Gardner, E. The space of interactions in neural network models. *J. Phys. A 21* (1988), 257.

[30] Gardner, E. Optimal basins of attraction in randomly sparse neural network models. *J. Phys. A 22* (1989), 1969.

[31] Gardner, E., and Derrida, B. Optimal storage properties of neural network models. *J. Phys. A 21* (1988), 271.

[32] Gershenson, C. Classification of random boolean networks. pp. 1–8.

[33] Goudarzi, A., Teuscher, C., and Gulbahce, N. Learning and generalization in random automata networks. In *Bionetics 2010: The 5th International ICST Conference on Bio-Inspired Models of Network, Information, and Computing Systems* (Dec 1–3 2010).

[34] Haselman, M., and Hauck, S. The future of integrated circuits: A survey of nanoelectronics. *Proceedings of the IEEE 98*, 1 (2010), 11–38.

[35] Haykin, S. *Neural Networks and Learning Machines (3rd Edition)*. Pearson Education, Inc., New York, NY, 2009.

[36] Hirotugu, A. A new look at the statistical model identification. *IEEE Transactions on Automatic Control 19*, 6 (1974), 716–723.

[37] Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences 79*, 8 (1982), 2554–2558.

[38] Hopfield, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc Natl Acad Sci U S A 81*, 10 (1984), 3088–92.

[39] Kari, J. Theory of cellular automata: a survey. *Theoretical Computer Science 334*, 1-3 (2005).

[40] Kauffman, S. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of theoretical biology 22*, 3 (1969), 437–467.

[41] Kauffman, S. Emergent properties in random complex automata. *Physica D 10*, 1-2 (1984), 145–156.

[42] Kauffman, S. Requirements for evolvability in complex systems: orderly dynamics and frozen components. *Physica D 42*, 1-3 (1990), 135–152.

[43] Kauffman, S. Coevolution to the edge of chaos: Coupled fitness landscapes, poised states, and coevolutionary avalanches. *Journal of Theoretical Biology 149*, 4 (1991), 467–505.

[44] Kauffman, S., Peterson, C., Samuelsson, B., and Troein, C. Genetic networks with canalyzing boolean rules are always stable. *Proc Natl Acad Sci U S A 101*, 49 (2004), 17102–7.

[45] Kauffman, S., and Weinberger, E. The NK model of rugged fitness landscapes and its application to maturation of the immune response. *Journal of Theoretical Biology 141*, 2 (1989), 211–45.

[46] Kauffman, S. A. Antichaos and adaptation. *Scientific American 265*, 2 (1991), 78–84.

[47] Kauffman, S. A. *The Origins of Order: Self–Organization and Selection in Evolution.* Oxford University Press, New York, NY, 1993.

[48] Krauth, W., and Mézard, M. Storage capacity of memory networks with binary couplings. *J. Phys. France 50*, 20 (1989), 3057–3066.

[49] Kube, K., Herzog, A., and Michaelis, B. Increased storage capacity in hopfield networks by small-world topology. *Lecture Notes in Computer Science 4221* (2006), 111–114.

[50] Langton, C. Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D* (1990).

[51] Larremore, D. B., Shew, W. L., and Restrepo, J. G. Predicting criticality and dynamic range in complex networks: Effects of topology. *Phys. Rev. Lett. 106*, 5 (2011), 058101.

[52] Li, W., and Packard, N. The structure of the elementary cellular automata rule space. *Complex systems 4*, 3 (1990), 281–297.

[53] Li, W., Packard, N., and Langton, C. Transition phenomena in cellular automata rule space. *Physica D 45*, 1-3 (1990), 77–94.

[54] Liu, M., and Bassler, K. Emergent criticality from coevolution in random boolean networks. *Physical Review E 74*, 4 (2006), 41910.

[55] Liu, M., and Bassler, K. Finite size effects and symmetry breaking in the evolution of networks of competing boolean nodes. *J. Phys. A 44* (2011), 045101.

[56] Martland, D. Auto-associative pattern storage using synchronous boolean networks. *Proceedings of the First IEEE International Conference on Neural Networks 3* (1987), 355–366.

[57] Martland, D. Behaviour of autonomous, (synchronous), boolean networks. *Proceedings of the First IEEE International Conference on Neural Networks 2* (1987), 243–250.

[58] Martland, D. Dynamic behavior of boolean networks. *Neural Computing Architectures: The Design of Brain-Like Machines, chapter 11* (1989), 217–235.

[59] McCulloch, W., and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology 5* (1943), 115–133. 10.1007/BF02478259.

[60] Mesot, B., and Teuscher, C. Deducing local rules for solving global tasks with random boolean networks. *Physica D 211*, 1-2 (2005), 88–106.

[61] Mitchell, M., Crutchfield, J., and Hraber, P. Evolving cellular automata to perform computations: Mechanisms and impediments. *Physica D 75* (1994), 361–391.

[62] Mitchell, M., Crutchfield, J. P., and Hraber, P. T. Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex systems 7* (1993), 89–130.

[63] Mitchell, M., Crutchfield, J. P., and Hraber, P. T. Dynamics, computation, and the "edge of chaos": A re-examination. In *Complexity: Metaphors, Models, and Reality. Reading*, D. P. G. Cowan and D. Melzner, Eds. Addison-Wesley, Reading, MA, 1994.

[64] Newman, M. The structure and function of complex networks. *SIAM Review 45*, 2 (2003), 167–256.

[65] Newman, M. Fast algorithm for detecting community structure in networks. *Physical Review E 69*, 6 (2004), 066133.

[66] Newman, M. E. J. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E 74*, 3 (2006), 036104.

[67] Nykter, M., Price, N., Larjo, A., Aho, T., Kauffman, S., Yli-Harja, O., and Shmulevich, I. Critical networks exhibit maximal information diversity in structure-dynamics relationships. *Phys. Rev. Lett. 100*, 5 (2008), 58702.

[68] Park, J., and Newman, M. Statistical mechanics of networks. *Physical Review E 70*, 6 (2004), 66117.

[69] Parrondo, J. M. R., and Van den Broeck, C. Vapnik-Chervonenkis bounds for generalization. *J. Phys. A 26*, 9 (1993), 2211–2223.

[70] Patarnello, A., and Carnevali, P. Learning networks of neurons with boolean logic. *Europhys Lett. 4*, 4 (1987), 503–508.

[71] Patarnello, S., and Carnevali, P. Learning capabilities of boolean networks. 1989, ch. 7, pp. 117–129.

[72] Plutynski, A. What was fisher's fundamental theorem of natural selection and what was it for? *Studies in History and Philosophy of Science Part C: Studies in History and Philosophy of Biological and Biomedical Sciences 37*, 1 (2006), 59–82.

[73] Price, G. R. Fisher's 'fundamental theorem' made clear. *Annals of Human Genetics 36*, 2 (1972), 129–140.

[74] Raman, K., and Wagner, A. The evolvability of programmable hardware. *Journal of The Royal Society Interface* (2010).

[75] Rohlf, T., Gulbahce, N., and Teuscher, C. Damage spreading and criticality in finite random dynamical networks. *Phys. Rev. Lett. 99*, 24 (2007), 248701.

[76] Rohlf, T., and Winkler, C. Network structure and dynamics, and emergence of robustness by stabilizing selection in an artificial genome. *submitted to 8th German Workshop on Artificial Life (GWAL 8)* (2008).

[77] Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review 65*, 6 (1958), 386–408.

[78] Rubinov, M., and Sporns, O. Complex network measures of brain connectivity: uses and interpretations. *Neuroimage 52*, 3 (2010), 1059–69.

[79] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning internal representations by error propagation. In *Parallel Distributed Processing*, D. E. Rumelhart and J. L. McClelland, Eds., vol. 1. MIT Press, Cambridge, MA, 1986, ch. 8, pp. 318–362.

[80] Sauer, N. On the density of families of sets. *Journal of Combinatorial Theory, Series A 13*, 1 (1972), 145–147.

[81] Serra, R., Villani, M., and Agnostini, L. On the dynamics of scale-free boolean networks. *Lecture Notes in Computer Science 2859* (2003), 43–49.

[82] Serra, R., Villani, M., and Agostini, L. On the dynamics of random boolean networks with scale-free outgoing connections. *Physica A 339*, 3-4 (2004), 665–673.

[83] Shmulevich, I., and Kauffman, S. Activities and sensitivities in boolean network models. *Phys. Rev. Lett. 93*, 4 (2004), 048701.

[84] Socolar, J. E. S., and Kauffman, S. A. Scaling in ordered and critical random boolean networks. *Phys. Rev. Lett. 90*, 6 (2003), 068702.

[85] Teuscher, C. *Turing's Connectionism. An Investigation of Neural Network Architectures*. Springer-Verlag, London, 2002.

[86] Teuscher, C., Gulbahce, N., and Rohlf, T. Assessing random dynamical network architectures for nanoelectronics. *Nanoscale Architectures, 2008. NANOARCH 2008. IEEE International Symposium on* (2008), 16–23.

[87] Teuscher, C., Gulbahce, N., and Rohlf, T. An assessment of random dynamical network automata. *International Journal of Nanotechnology and Molecular Computation 1*, 4 (2009), 58–73.

[88] Teuscher, C., Gulbahce, N., Rohlf, T., and Goudarzi, A. Random dynamical network automata for nanoelectronics: A robustness and learning perspective. In *Theoretical and Technological Advancements in Nanotechnology and Molecular Computation: Interdisciplinary Gains*, B. MacLennan, Ed. IGI Global, Hershey, NY, 2011, ch. 19, pp. 295–314.

[89] Tomassini, M., Giacobini, M., and Darabos, C. Evolution of small-world networks of automata for computation. *Lecture Notes in Computer Science 3242* (2004), 672–681.

[90] Turing, A. M. Intelligent machinery. In *Machine Intelligence*, B. Meltzer, Ed., vol. 5. Edinburgh University Press, Edinburgh, 1969, pp. 3–23.

[91] Valiant, L. A theory of the learnable. *Communications of the ACM 27*, 11 (1984), 1134–1142.

[92] Van den Broeck, C. Statistical physics of learning from examples - a brief introduction. *Acta Phys Pol B 25*, 6 (1994), 903–923.

[93] Van den Broeck, C., and Kawai, R. Learning in feedforward boolean networks. *Phys. Rev. A 42*, 10 (1990), 6210–6218.

[94] Vapnik, V. N. *Estimation of dependences based on empirical data.* Springer-Verlag, New York, NY, Jan 1982.

[95] Vapnik, V. N. An overview of statistical learning theory. *Neural Networks, IEEE Transactions on 10*, 5 (1999), 988 –999.

[96] Vapnik, V. N., and Ya. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications 16*, 2 (1971), 264–280.

[97] von Neumann, J. *Theory of Self-Reproducing Automata.* University of Illinois Press, Urbana, Illinois, 1966.

[98] Widrow, B. Generalization and information storage in network of adaline 'neurons'. In *Self-Organizing Systems*, G. J. M. Yovitz and G. Goldstein, Eds. Spartan Books, Washington, DC, 1962, pp. 435–461.

[99] Wittaker, E. T., and Robinson, G. The trapezoidal and parabolic rules. In *The Calculus of Observations: A Treatise on Numerical Mathematics*, 4th ed. Dover, New York, NY, 1967, pp. 156–158.

[100] Wolfram, S. Statistical mechanics of cellular automata. *Rev. Mod. Phys. 55*, 3 (1983), 601–644.

[101] Wootters, W., and Langton, C. Is there a sharp phase transition for deterministic cellular automata? *Physica D 45*, 1-3 (1990).

Appendix A

APPENDIX

Here, we have listed the number of free parameters (DF), the sum squared error (SSE), adjusted AIC (AIC_{adj}), ΔAIC_{adj}, the relative likelihood (ℓ), and the Akaike weight (w_ℓ) for four models (see Section 4.1.3) fitted to the scaling scaling data for the three computational tasks, i.e., the R85, the full-adder, and the even-odd tasks and two training sample sizes $T = 4$ and $T = 8$.

model	DF	SSE	AIC_{adj}	ΔAIC_{adj}	ℓ	w_ℓ
M1	3	0.0051	-46.8340	0.0000	1.0000	0.9901
M2	4	0.0050	-37.6253	9.2087	0.0100	0.0099
M3	2	0.1627	-24.7630	22.0710	0.0000	0.0000
M4	2	3.2150	-0.8929	45.9411	0.0000	0.0000

Table A.1: The even-odd task, $T = 8$.

model	DF	SSE	AIC_{adj}	ΔAIC_{adj}	ℓ	w_ℓ
M1	3	0.0036	-49.5691	0.0000	1.0000	0.9997
M2	4	0.0086	-33.3053	16.2638	0.0003	0.0003
M3	2	0.1083	-28.0155	21.5536	0.0000	0.0000
M4	2	3.7001	0.2313	49.8004	0.0000	0.0000

Table A.2: The even-odd task, $T = 4$.

model	DF	SSE	AIC_{adj}	ΔAIC_{adj}	ℓ	w_ℓ
M1	3	0.0124	-39.7291	0.0000	1.0000	0.9577
M2	4	0.0085	-33.4806	6.2485	0.0440	0.0421
M3	2	0.2080	-22.7980	16.9311	0.0002	0.0002
M4	2	4.0152	0.8851	40.6142	0.0000	0.0000

Table A.3: The full-adder task, $T = 8$.

model	DF	SSE	AIC_{adj}	ΔAIC_{adj}	ℓ	w_ℓ
M1	3	0.0170	-37.2356	0.0000	1.0000	0.9827
M2	4	0.0178	-27.5166	9.7190	0.0078	0.0076
M3	2	0.1087	-27.9884	9.2472	0.0098	0.0096
M4	2	4.8883	2.4592	39.6948	0.0000	0.0000

Table A.4: The full-adder task, $T = 4$.

model	DF	SSE	AIC_{adj}	ΔAIC_{adj}	ℓ	w_ℓ
M1	3	0.0142	-38.6577	4.9534	0.0840	0.0775
M2	4	0.0162	-28.2748	15.3363	0.0005	0.0004
M3	2	0.0154	-43.6111	0.0000	1.0000	0.9221
M4	2	4.6586	2.0742	45.6853	0.0000	0.0000

Table A.5: The R85 task, $T = 8$.

model	DF	SSE	AIC_{adj}	ΔAIC_{adj}	ℓ	w_ℓ
M1	3	0.0292	-32.8997	3.6981	0.1574	0.1359
M2	4	0.0344	-22.2688	14.3290	0.0008	0.0007
M3	2	0.0371	-36.5978	0.0000	1.0000	0.8634
M4	2	5.2193	2.9834	39.5812	0.0000	0.0000

Table A.6: The R85 task, $T = 4$.