Random Automata Networks: Why Playing Dice is not a Vice

Christof Teuscher
Portland State University, teuscher@pdx.edu

Follow this and additional works at: https://pdxscholar.library.pdx.edu/systems_science_seminar_series

Part of the Systems Architecture Commons, and the Theory and Algorithms Commons

Let us know how access to this document benefits you.

Recommended Citation
Teuscher, Christof, "Random Automata Networks: Why Playing Dice is not a Vice" (2010). Systems Science Friday Noon Seminar Series. 25.
https://pdxscholar.library.pdx.edu/systems_science_seminar_series/25

This Book is brought to you for free and open access. It has been accepted for inclusion in Systems Science Friday Noon Seminar Series by an authorized administrator of PDXScholar. Please contact us if we can make this document more accessible: pdxscholar@pdx.edu.
Introduction
- Unstructured assemblies

Turing’s Unorganized Machines
- Different types
- Learning
- Genetic training

Random Boolean Networks
- The initial NK model
- Perturbation analysis
- Solving tasks
- Learning and generalization

Conclusion:
- Main message: The benefits of disordered systems can be harnessed if we challenge the traditional computing paradigms.

Embracing Randomness
- The top-down way we fabricate electronic chips is not sustainable at the current pace of progress.
- Bottom-up self-assembled computers are the holy grail of molecular and nanotechnology.
- We lack control over such techniques, thus, such machines will be partly or largely unstructured and imperfect.
- Such interconnects would be easier and cheaper to build in massive scale.
- “Irregular networks are a more realistic approach to modeling biological information processing than cellular automata”.

Fabricating Unstructured Nanowire Assemblies

Key challenges:
- precise positioning and alignment
- low-resistance contacts

Polyaniline (PANI) conductive polymer, LANL, Wang et al.

Gracias team, John Hopkins University

Gu et al., Three-Dimensional Electrically Interconnected Nanowire Networks Formed by Diffusion Bonding, Langmuir 2007, 23, 879-882.

Turing Day 2002
Computing science 90 years from the birth of Alan M. Turing

B. Jack Copeland
Martin Davis
Andrew Hodges
Douglas Hofstadter

Jonathan Swinton
Gianluca Tempesti
Christof Teuscher

THE ALAN TURING YEAR
A Centenary Celebration of the Life and Work of Alan Turing

June 23, 2012, is the Centenary of Alan Turing’s birth in London. During his lifetime, Alan Turing made a unique impact on the history of computing, computer science, artificial intelligence, developmental biology, and the mathematical theory of computability.

2012 will be a celebration of Turing’s life and scientific impact, with a number of major events taking place throughout the year. Most of these will be linked to places with special significance in Turing’s life, such as Cambridge, Manchester and Bletchley Park.

The Turing Year is coordinated by the Turing Centenary Advisory Committee (TCAC), representing a range of expertise and organisational involvement in the 2012 celebrations. Organisations and individuals wanting to contribute ideas or support for the Turing Year are invited to contact any of the current TCAC members.
Turing’s Unorganized Machines

Alan Turing, Intelligent Machinery, 1948, National Physical Laboratory Report.

Turing’s Unorganized Machines

- A. M. Turing, Intelligent Machinery
  - written in 1948 and dismissed as a “schoolboy” essay
  - published in 1968
  - revived by Copeland and Proudfoot
  - Synthese, 1996
  - Scientific American, April 1999
- Rosenblatt, 1957
- McCulloch and Pitts, 1943

“An unorganized machine is a machine made up in a comparatively unsystematic way from some kind of standard components.”

Turing’s three brilliant ideas:
1. Organized machines: TM, classical computer
2. Unorganized machines: neural network
3. Self-organizing (self-modifying) machines: neural network, biological system displaying growth

Link and Network Types
"The A-type unorganized machines are of interest as being about the simplest model of a nervous system with a random arrangement of neurons."
— Alan Turing, “Intelligent Machinery”, 1948.
B-type Unorganized Machines

B-type Connection

- two stable states (opened/closed)
- two meta stable states

Introverted pair

B-type Connection (cont.)

B-type Connection (cont.)

Disabled connection (meta stable)

Enabled connection

$|A|B = 00$

$x \in \{0,1\}$

$S_0$

$10$

$1x$

$S_1$

$S_2$

$S_3$

$0x$

$01$

$11$

$xx$
BI-type Connection

Interfering Inputs
0 1 enabled/closed
1 0 disabled/opened

BI-type Network

Self-organization
Supervised Learning

“Genetic or Evolutionary Search”

Pattern Classification

Genetic algorithm
Pattern Classification and Control Tasks

Complex Dynamics

network activity level = % of nodes set to 1

Building a Multiplexer

Kauffman NK Networks
Kauffman NK Networks

- $N =$ number of nodes
- $K =$ interaction between the nodes, i.e., the number of incoming links per node

NK Network Properties

- $K = 2$ represents a phase transition, gas-liquid-solid, complex regime
- $K < 2$: solid, frozen, ordered
- $K > 2$ liquid, chaotic
- $K=0$: nothing happens!
- $K=N$: Fully connected network:
  - Average attractor cycle length increases exponentially with $N$.
  - $N=500$: the average state cycle is $10^{75}$ steps long! (→ the system appears to be chaotic)
  - Number of attractors decreases linearly with $N$.
  - For example, for $N=500$ and $K=500$, there are about 185 different attractors.
- $K=2$: complex regime
  - Number of state cycle attractors: $\sim \sqrt{N}$
  - Median state cycle length: $\sim \sqrt{N}$
  - Inherently stable to most transient perturbations.

NK versus Genetic Networks

- For $N=100,000$ there exist around 317 attractors.
- In humans the number of genes roughly equals 100,000.
- The number of known cell types is 256, which is not too far away from 317.
- Cell types may actually represent different attractors of the genetic network.
- Also valid in other species:

Random Boolean Network Matlab Toolbox

- Features:
  - Different updating schemes
  - Critical values, attractors, etc.
  - Easy to use
- Alternatives:
  - Andy Wuensche’s DDLab
  - Carlos Gershenson’s RBNLab

http://www.teuscher.ch/rbntoolbox
Specific Research Questions

- How can we analytically deduce the local node rules from the global task description?
- Do Random Boolean Networks (RBNs) perform better than Cellular Automata (CAs) on “global” tasks?
- Joint work with Bertrand Mesot, IDIAP

1D and 2D Firefly Synchronization Task

No perfect, uniform 2-state CA exists.

Density Classification Task

- The density classification task consists of correctly determining whether the initial configuration contains a majority of 1’s or a majority of 0’s, by making the system converge, respectively, to an all 1’s state, or to a state of all 0’s

Firefly Task Formalization

$p_{t+1} = Q_k(p_t)$

$p_t = \text{fraction of nodes in state 1 at time } t$
Density Task Formalization

\[ p_{1}^{t+1} = Q_{K}(p_{1}^{t}) \]

\[ p_{1}^{t+1} - p_{1}^{t} \]

- \( p_{1}^{t} \): fraction of nodes in state 1 at time \( t \)

Some Math

Probability of having \( d \) neighbors in state 1 at time \( t \), knowing the percentage of 1 in the global network state:

\[ P(W_{1}^{t} = d \mid P_{1}^{t} = p_{1}^{t}) = \binom{K}{d} (p_{1}^{t})^{d} (1 - p_{1}^{t})^{K-d} \]

Finding the Rules

- Density classification task:
  - \( K=2 \): no solution
  - \( K=3 \): \( Q_{3}(x) = 3x^{2}(1-x) + x^{3}, a_{0}=a_{1}=0 \) \( \Rightarrow \) majority rule
  - \( K=4 \): 20 rules
  - \( K=5 \): 6 polynomials

- Synchronization task:
  - Two functions that satisfy the conditions:
    \[ Q_{K}(x) = (1-x)^{K} \text{ and } Q_{K}(x) = \sum_{i=0}^{K-1} x^{i} (1-x)^{K-i} \]
  - \( \gamma K \)-rules: output 1 iff all inputs 0 or output 0 iff all inputs 1

Density Classification Performance

- \( K \) odd: the bigger \( K \), the better the performance
- The bigger \( N \), the better the performance

\[ N \]

\[ 70, 75, 80, 85, 90, 95, 100 \]
CA Rules for Comparison

- Density classification rules by Mitchell et al., 1996
  - \( \Phi_{\text{GKL}} \): hand-designed, derived by Gacs et al.
  - \( \Phi_{\text{maj}} \): majority rule
  - \( \Phi_{\text{exp}} \): evolutionary algorithm, uses a “block expansion” strategy
  - \( \Phi_{\text{par}} \): evolutionary algorithm, uses a “particle-based” strategy

- Synchronization task, Das et al., 1995
  - \( \Phi_1 \) - \( \Phi_3 \), \( \Phi_{\text{sync}} \): evolved

- CAs with \( r=3 \) neighborhood, RBN with \( K=2r+1 \)
- \( 10^4 \) initial configurations

Density Classification Task

<table>
<thead>
<tr>
<th>Automaton</th>
<th>Rule</th>
<th>( P_{149,10^4} )</th>
<th>( P_{599,10^4} )</th>
<th>( P_{999,10^4} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA</td>
<td>( \Phi_{\text{maj}} )</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>CA</td>
<td>( \Phi_{\text{exp}} )</td>
<td>0.652</td>
<td>0.515</td>
<td>0.503</td>
</tr>
<tr>
<td>CA</td>
<td>( \Phi_{\text{par}} )</td>
<td>0.769</td>
<td>0.725</td>
<td>0.714</td>
</tr>
<tr>
<td>CA</td>
<td>( \Phi_{\text{GKL}} )</td>
<td>0.816</td>
<td>0.766</td>
<td>0.757</td>
</tr>
<tr>
<td>RBN</td>
<td>( \vartheta_3 )</td>
<td>0.766</td>
<td>0.771</td>
<td>0.769</td>
</tr>
<tr>
<td>RBN</td>
<td>( \vartheta_5 )</td>
<td>0.823</td>
<td>0.825</td>
<td>0.820</td>
</tr>
<tr>
<td>RBN</td>
<td>( \vartheta_7 )</td>
<td>0.850</td>
<td>0.848</td>
<td>0.852</td>
</tr>
</tbody>
</table>
### Synchronization Task

<table>
<thead>
<tr>
<th>Automaton</th>
<th>Rule</th>
<th>$P_{149,10^4}$</th>
<th>$P_{599,10^4}$</th>
<th>$P_{999,10^4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA</td>
<td>$\phi_1$</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>CA</td>
<td>$\phi_2$</td>
<td>0.33</td>
<td>0.07</td>
<td>0.03</td>
</tr>
<tr>
<td>CA</td>
<td>$\phi_3$</td>
<td>0.57</td>
<td>0.33</td>
<td>0.27</td>
</tr>
<tr>
<td>CA</td>
<td>$\phi_{\text{sync}}$</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>RBN</td>
<td>$\gamma_3$</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>RBN</td>
<td>$\gamma_5$</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>RBN</td>
<td>$\gamma_7$</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

### Results in a Nutshell

- No $K=2$ RBN can perfectly solve the density classification task.
- CAs: the bigger $N$, the worse the result.
- RBNs: the bigger $N$, the better the result.
- The majority rule is the only perfect rule for $K=3$, whereas many rules exist for $K > 3$.
- The two symmetrical $\gamma_k$-rules are the best rules for the synchronization task.
- Random rewiring possible.

### RBN versus Small-World Topologies

- Density task: majority rule
- Synchronization task: $\gamma$-rule

### RBN Node Updating Schemes

- Node updating schemes:
  - Classical RBNs (CRBN) $\rightarrow$ well known and many results
  - Asynchronous RBNs (ARBN) $\rightarrow$ little results
  - Deterministic Asynchronous RBNs (DARBNs)
  - Generalized Asynchronous RBNs (GARBNs)
  - Deterministic Generalized Asynchronous RBNs (DGARBN)
RBN Node Updating Schemes

<table>
<thead>
<tr>
<th>RBN</th>
<th>updating scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRBNs</td>
<td>synchronous, deterministic</td>
</tr>
<tr>
<td>ARBNs</td>
<td>asynchronous, non-deterministic</td>
</tr>
<tr>
<td>DARBNs</td>
<td>asynchronous, deterministic</td>
</tr>
<tr>
<td>GARBNs</td>
<td>semi-synchronous, non-deterministic</td>
</tr>
<tr>
<td>DGA RBNs</td>
<td>semi-synchronous, deterministic</td>
</tr>
</tbody>
</table>


RBN Topology Evolution

- Very simple local rewiring algorithm:
  - Measure the activity of each node in an attractor
  - Active nodes: remove incoming connections
  - Inactive nodes: add new connections
- \( K = 2 \) for \( N \to \infty \)
  - Threshold networks (Bornholdt and Rohlf)
  - Turing’s unorganized machines (Teuscher and Sanchez, ICANN2001)
  - CRBN, ARBN, DARBN, GARBN, DGARBN

RBN Topology Evolution: Connectivity

- \( N = 15, T = 1000 \)

External Perturbations (1)

- Motivating question: How much connectivity does a system need to be efficient, robust, and cheap?
- Damage spreading is relevant for many applications
  - Disease spreading
  - Computer viruses
  - Failures on power grids
  - Nano-scale networks, e.g., single event upsets (SEU)
  - Gene expressions
- Study done for both Random Boolean Networks (RBN) and Random Threshold Networks (RTN).
External Perturbations (2)

- Mean-field approaches (e.g., annealed approximation introduced by Derrida and Pomeau) provide an analytical treatment of damage spreading and an extract determination of the critical connectivity $K_c$.
- These approaches rely on the assumption that $N \to \infty$ and study the rescaled damage $d(t)/N$.
- For (finite) real-world networks and problems, these limits are often not very relevant.
- We are interested in the "sparse percolation limit," where the initial perturbation does not scale up with the system size $N$.
- Again, applies to many real-world problems.

Measuring Damage Spreading in Random Boolean Networks (RBN)

- Network 1
  - Initial state: 0 1 0 1 1 0 0 1
  - Final state: 1 1 0 1 0 1 1 0
- Network 2
  - Initial state: 0 1 1 1 0 0 1
  - Final state: 0 0 0 1 0 1 1 0

Hamming distance

Damage Spreading and System Scaling in RBNs

- "edge of stability" $k_s$
- "edge of chaos" $k_c$ (Kauffman, Derrida, and others)
- $K_s=1.875$
- Also done for RTNs.
From Random Networks to Extreme Local Networks: The Watts-Strogatz Small-World Model

Figure 1. Watts-Strogatz model interpolates between a regular lattice (left) and a random graph (right). Randomly rewiring just a few edges can reduce the average distance between nodes, but has little effect on the clustering coefficient, C. The result is a “small-world” graph.

How does $K_s$ move?

(a) Random networks
(b) SW networks with $p=0.9$
(c) SW networks with $p=0.8$
(d) Local networks

Wiring cost, robustness, and communication characteristics

- **Wiring cost**: sum of geometric distances of the links
- **Path length**: average shortest path length (communication characteristics)
- **Damage**: Hamming distance (robustness)

Learning and Generalization
### Learning and Generalization

- Broeck and Kawai (1990)
  - Predicted the learning probability using mean field techniques in feedforward Boolean networks.
  - Defined problem complexity measure, *phase volume*.

\[ I = 7: \text{20\% fraction of teaching allow the networks to generalize on all input patterns in 100\% of the runs.} \]
\[ I = 3: \text{All input patterns are needed to achieve 100\% learning probability.} \]

From Broeck and Kawai (1990). Learning probability for addition modulo 2 (Even-odd.)

#### Computational Tasks

**Mapping**
\[ I=4 \]

**Even-odd**
\[ I=5 \]

#### Results for RBNs

- Even-odd
- Fixed \( K=2, N=20 \)
- 500 runs

- Learning probability increases with \( I \) and \( s \).
- For \( I=7 \), with 20\% of input space only reaches about 30\% learning probability. This is \textbf{3 times worse} than feedforward nets at this point.

#### Cumulative Generalization Score for \( I=3 \)

- Even-odd
- \( I=3 \)
- \(<K>=1.0-4.9\)
- \( N=10-20 \)

- No change with \( N \). The task is too easy.
- The higher \(<K>\), the higher the generalization.
Critical Connectivity

Conclusion 😊

www.cafepress.com/rbnrock