
UNIVERSITÉ DE LAUSANNE
FACULTÉ DES HAUTES ÉTUDES COMMERCIALES

Towards Robust Network Based Complex Systems
From Evolutionary Cellular Automata to Biological Models

THÈSE

Présentée à la Faculté des Hautes Études Commerciales
de l'Université de Lausanne, Suisse
en cotutelle avec l'Université de Turin, Italie

par

Christian DARABOS

Titulaire d'un Diplôme d'Informaticien
de l'Université de Lausanne, Suisse

Titulaire d'un Master en Bioinformatique
de l'Université de Turin, Italie

Pour l'obtention des grades de
Docteur en Systèmes d'Information

2010



UNIL | Université de Lausanne
HEC Lausanne
Le Doyen
Bâtiment Internef
CH-1015 Lausanne

IMPRIMATUR

Sans se prononcer sur les opinions de l'auteur, le Conseil de la Faculté des hautes études commerciales de l'Université de Lausanne autorise l'impression de la thèse de Monsieur Christian DARABOS, titulaire d'un diplôme d'informaticien de l'Université de Lausanne, titulaire d'un master en bioinformatique de l'Université de Turin, en vue de l'obtention du grade de docteur en Systèmes d'information.

La thèse est intitulée :

**TOWARDS ROBUST NETWORK BASED COMPLEX SYSTEMS
FROM EVOLUTIONARY CELLULAR AUTOMATA
TO BIOLOGICAL MODELS**

Lausanne, le 19 avril 2010

Le doyen

Daniel Oyon

Members of the Jury

Co-supervisors

Prof. **Marco Tomassini**
University of Lausanne
Faculty of Business and Economics
Department of Information Systems
CH-1015 Lausanne, Switzerland
marco.tomassini@unil.ch

Prof. **Ferdinando Di Cunto**
University of Torino
Faculty of Medicine and Surgery
Department of Genetics, Biology
and Biochemistry
IT-10126 Torino, Italy
ferdinando.dicunto@unito.it

Internal Experts

Prof. **Alessandro Villa**
University of Lausanne
Faculty of Business and Economics
Department of Information Systems
CH-1015 Lausanne, Switzerland
alessandro.villa@unil.ch

Dr. **Paolo Provero**
University of Torino
Molecular Biotechnology Center
Computational Biology Unit
IT-10126 Torino, Italy
paolo.provero@unito.it

External Experts

Prof. **Roberto Serra**
University of Modena and Reggio Emilia
Faculty of Communication and Economics
Department of Social, Cognitive and
Quantitative Sciences
I-42121 Reggio Emilia, Italy
roberto.serra@unimore.it

Dr. **Andreas Kremer**
Erasmus Medical Center
Department Bioinformatics
NL-3015 GE Rotterdam
The Netherlands
a.kremer@erasmusmc.nl

University of Lausanne
Faculty of Business and Economics

Doctorate in Business Information Systems

I hereby certify that I have examined the doctoral thesis of

Christian DARABOS

and have found it to meet the requirements for a doctoral thesis.

All revisions that I or committee members
made during the doctoral colloquium
have been addressed to my entire satisfaction.

Signature : Marco Tomassini Date : 29/3/2010

Prof. Marco TOMASSINI
Thesis director at the University of Lausanne

University of Lausanne
Faculty of Business and Economics

Doctorate in Business Information Systems

I hereby certify that I have examined the doctoral thesis of

Christian DARABOS

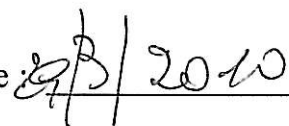
and have found it to meet the requirements for a doctoral thesis.

All revisions that I or committee members
made during the doctoral colloquium
have been addressed to my entire satisfaction.

Signature :



Date :



Prof. Ferdinando DI CUNTO
Thesis director at the University of Torino

University of Lausanne
Faculty of Business and Economics

Doctorate in Business Information Systems

I hereby certify that I have examined the doctoral thesis of

Christian DARABOS

and have found it to meet the requirements for a doctoral thesis.

All revisions that I or committee members
made during the doctoral colloquium
have been addressed to my entire satisfaction.

Signature :  Date : 29/3/2010

Prof. Alessandro VILLA
Internal member of the doctoral committee

University of Lausanne
Faculty of Business and Economics


Doctorate in Business Information Systems

I hereby certify that I have examined the doctoral thesis of

Christian DARABOS

and have found it to meet the requirements for a doctoral thesis.

All revisions that I or committee members
made during the doctoral colloquium
have been addressed to my entire satisfaction.

Signature :  Date : 29/3/2010

Dr. Paolo PROVERO
External member of the doctoral committee

University of Lausanne
Faculty of Business and Economics

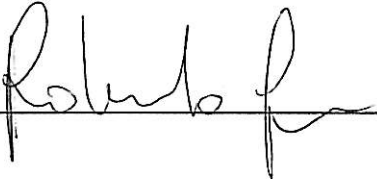
Doctorate in Business Information Systems

I hereby certify that I have examined the doctoral thesis of

Christian DARABOS

and have found it to meet the requirements for a doctoral thesis.

All revisions that I or committee members
made during the doctoral colloquium
have been addressed to my entire satisfaction.

Signature :  Date : 29/3/2010

Prof. Roberto SERRA
External member of the doctoral committee

University of Lausanne
Faculty of Business and Economics

Doctorate in Business Information Systems


I hereby certify that I have examined the doctoral thesis of

Christian DARABOS

and have found it to meet the requirements for a doctoral thesis.

All revisions that I or committee members
made during the doctoral colloquium
have been addressed to my entire satisfaction.

Signature :



Date :



Dr. Andreas KREMER
External member of the doctoral committee

To my mother,
near or far, her support is unfaltering.

Abstract

Sitting between your past and
your future doesn't mean you are
in the present.

Dakota Skye

Complex systems science is an interdisciplinary field grouping under the same umbrella dynamical phenomena from social, natural or mathematical sciences. The emergence of a higher order organization or behavior, transcending that expected of the linear addition of the parts, is a key factor shared by all these systems. Most complex systems can be modeled as networks that represent the interactions amongst the system's components. In addition to the actual nature of the part's interactions, the intrinsic topological structure of underlying network is believed to play a crucial role in the remarkable emergent behaviors exhibited by the systems. Moreover, the topology is also a key a factor to explain the extraordinary flexibility and resilience to perturbations when applied to transmission and diffusion phenomena. In this work, we study the effect of different network structures on the performance and on the fault tolerance of systems in two different contexts.

In the first part, we study cellular automata, which are a simple paradigm for distributed computation. Cellular automata are made of basic Boolean computational units, the cells, relying on simple rules and information from the surrounding cells to perform a global task. The limited visibility of the cells can be modeled as a network, where interactions amongst cells are governed by an underlying structure, usually a regular one. In order to increase the performance of cellular automata, we chose to change its topology. We applied computational principles inspired by Darwinian evolution, called evolutionary algorithms, to alter the system's topological structure starting from either a regular or a random one. The outcome is remarkable, as the resulting topologies find themselves sharing properties of both regular and random network, and display similitudes Watts-Strogatz's small-world network found in social systems. Moreover, the performance and tolerance to probabilistic faults of our small-world like cellular automata surpasses that of regular ones.

In the second part, we use the context of biological genetic regulatory networks and, in particular, Kauffman's random Boolean networks model. In some ways, this model is close to cellular automata, although is not expected to perform any task. Instead, it simulates the time-evolution of genetic regulation within living organisms under strict conditions. The original model, though very attractive by it's simplicity, suffered from important shortcomings unveiled by the recent advances in genetics and biology. We propose to use these new discoveries to improve the original model. Firstly, we have used artificial topologies believed to be closer to that of gene regulatory networks. We have also studied actual biological organisms, and used parts of their genetic regulatory networks in our models. Secondly, we have addressed the improbable full synchronicity of the event taking place on Boolean networks and proposed a more biologically plausible cascading scheme. Finally, we tackled the actual Boolean functions of the model, i.e. the specifics of how genes activate according to the activity of upstream genes, and presented a new update function that takes into account the actual promoting and repressing effects of one gene on another. Our improved models demonstrate the expected, biologically sound, behavior of previous GRN model, yet with superior resistance to perturbations. We believe they are one step closer to the biological reality.

Synopsis

Que vous soyez assis entre votre passé et votre futur n'implique pas que vous soyez dans le présent.

Dakota Skye

La science des systèmes complexes est un domaine interdisciplinaire regroupant sous une même dénomination certains phénomènes dynamiques des sciences sociales, naturelles, de la physique ou des mathématiques. L'émergence d'une organisation d'ordre supérieur ou de comportements transcendants ceux attendus de l'addition linéaire des composants est une propriété cruciale partagée par tous ces systèmes. La plupart des systèmes complexes peuvent être modélisés sous la forme de réseaux qui représentent les interactions entre les composantes du système. Au delà de la nature des interactions, c'est la structure topologique du réseau sous-jacent qui joue un rôle essentiel dans l'émergence des comportements remarquables des systèmes. En outre, la topologie est également un facteur expliquant l'extraordinaire flexibilité et la résilience aux perturbations des systèmes complexes en réseaux appliqués à des phénomènes de transmission et de diffusion. Dans travail présent, nous analysons l'effet de différentes structures de réseaux sur la performance et sur la tolérance aux pannes des systèmes dans deux contextes différents.

Dans la première partie, nous étudions les automates cellulaires, qui sont un paradigme simple de calcul distribué. Les automates cellulaires sont faits d'unités de calcul booléen, les cellules, qui s'appuient sur des règles simples et l'information fournie par les cellules environnantes pour accomplir une tâche globale. La visibilité limitée des cellules peut être modélisée par un réseau où les interactions entre les cellules sont régies par la structure sous-jacente, habituellement celle d'un graphe régulier. Afin d'accroître les performances des automates cellulaires, nous avons choisi d'adapter cette topologie. Nous avons appliqué des principes de calculs inspirés de l'évolution darwinienne, appelés algorithmes évolutionnaires, afin de modifier la structure topologique du système en partant soit d'un réseau régulier, soit un réseau aléatoire. Les résultats sont remarquables; les topologies résultantes partagent des propriétés empruntées aux deux types de réseaux, réguliers et aléatoires, et montre des similitudes avec les réseaux "petits-mondes" de Watts-Strogatz inspirés de réseaux sociaux. De plus, la performance et la tolérance aux fautes probabilistes de nos nouveaux automates cellulaires surpasse celle des automates réguliers avec les règles artificielles.

Dans la seconde partie, nous utilisons le contexte des réseaux de régulation génétiques d'organismes biologique et, en particulier, le modèle de réseaux booléen de Kauffman. À certains égards, ce modèle est proche des automates cellulaires, mais n'a pas pour but d'accomplir une tâche. Au lieu de cela, ce modèle simule l'évolution temporelle des réseaux de régulation génétiques d'organismes vivants, sous des conditions strictes. Le modèle original, bien que très intéressante de par sa simplicité, souffre d'importantes lacunes, dévoilées par les récentes avancées en génétique et en biologie. Nous proposons d'utiliser ces nouvelles découvertes pour améliorer le modèle original. Tout d'abord, nous avons utilisé des topologies artificielle potentiellement plus proche de celle des réseaux de régulation géniques. Nous avons également étudié des organismes biologiques réels, et utilisé des parties de leur réseau de régulation génétique dans nos modèles. Deuxièmement, nous avons abordé l'improbable synchronicité des événements se déroulant sur les réseaux booléens et avons proposé un schéma en cascade plus plausible biologiquement. Enfin, nous avons abordé les

fonctions booléennes du modèle, à savoir les règles qui régissent l'activation des gènes en fonction de l'activité des gènes en amont, et nous présentons une fonction de mise à jour originale qui tient compte de l'effet promoteur et répresseur d'un gène sur un autre. Nos modèles améliorés démontrent non-seulement le comportement attendu, déduit du modèle original, mais avec une meilleure résistance aux perturbations. Enfin, nous pensons que ce modèle fait un pas important vers plus de réalité biologique.

Acknowledgements

Don't worry about the future; or worry, but know that worrying is as effective as trying to solve an algebra equation by chewing bubblegum.

Mary Schmich

Many thanks to...

... **Prof. Marco Tomassini**, my Ph.D. advisor, supervisor, mentor, and a brilliant scientist for sharing with me his absolute passion and devotion to research. He made me discover what it means to love science, and offered me the freedom to think independently, and to grow personally and intellectually. He really is “il più grande”.

... **Prof. Ferdinando Di Cunto**, for agreeing to become my Ph.D. advisor in the cotutelle, for sharing his invaluable biological insight, for being incredibly kind, patient, and tolerant with me and my models.

... **all the members of my jury**, Dr. Andreas Kremer, Prof. Roberto Serra, Dr. Paolo Provero, Prof. Alessandro Villa, for agreeing to share your expertise in the domain at hand. I appreciate your comments and the time spent on reviewing my work.

... **Dr. Mario Giacobini**, with whom everything started about 6 years ago. We worked together on my masters dissertation. We became great friends. He is the reason I started a Ph.D. in the first place and the reason I undertook the adventure of a cotutelle with the University of Torino. He's been part of all my successes, our successes... Mario and his family, Silvia, Martina, Davide, and his parents Giacomo and Maria Giuseppina, are the kindest, most generous people I have ever met: I slept under their roofs, ate their food, took part in their daily life. They made me feel like I belong. I cannot thank them enough.

... **Prof. Monty Kier**, and his wife Martha, for their kindness, their generosity, their encouragements, their insight in the world of science and its underlying politics.

... **those who shared my office at some point**: Enea, Marc, Fabio, and Javier. Enea, we shared good times, stressful times, teaching, winter school, conferences, travels, stomach bugs, you are the most easy going person I know. Marc, we had some good rants, good raves, and awesome laughs. Fabio, you're the most patient person I know, you've put up with so much with me, never calling me out on anything, how do you do this ? I sometimes cannot stand myself, you always seem to be able to.

... **the ISI faculty and staff**, Prof. François Grize, Prof. Yves Pigneur, Prof. Benoît Garbinato, who taught me about how a department works, gave me a task for faculty politics, and gave me great advices and encouragements. Ms. Elisabeth Fournier for her outstanding work as administrative assistant, her professionalism, and her efficiency. All these qualities are greatly appreciated and underestimated by many. François Vessaz for his help maintaining and administrating the computer systems, the grid, the database and backups.

... **my friends**, in no particular order: Leslie and MJ, Alexandra and Loïc, Aline, Julie, Mano, Alain, and all the others, you guys know who you are and how much I love you. Leslie, now I want that burger, you promised !

... **my mother** who's been living far away for a few years but always felt really close. She's always so proud of me. I know that no matter what I do, or what my choices are, I can count on your unfaltering support. Thanks to my very limited family, my cousins Barbara and Anett, and uncle Zoltan, I love you guys, even if I don't show it enough.

... **the people I have met along the way**, Biljana, Christian, Christof, Joshua, Davnah, Una-May, Dan and Sara, and many others. You are my friends, you help me out, you inspire me, you influence my career and shape my future in more ways than you think...

Well, this was fun... now let's get a move on !

Christian Darabos
Lausanne, April 2010

Preface

Once you have eliminated the impossible, whatever remains, however improbable, must be the truth.

Sir Arthur Conan Doyle

This Ph.D. dissertation differs from the usual standards adopted in Switzerland. This is the reason why I offer some technical explanation about the format of this document and the international collaborations that allowed me obtain two Ph.D. degrees from two different European universities.

Collection of Articles

This Ph.D. thesis dissertation is in the form of a *collection of articles*. Three out of the four articles presented here have been published in international specialized journals. The fourth and most recent one has been submitted and is under review. Besides the introduction (Part I) and the final considerations (Part IV), each part of this document introduces novel concepts and original ideas, and the related articles are included as annexes (identified by the letters A through D) within the document, not at the end. To avoid as much repetition as possible, I do not present every single publication we have produced over the five-year period of my Ph.D. The complete list of journal, conference and workshop publications can be found at the end of this document. Nevertheless, some degree of overlap between the articles and introductory parts remains inevitable as I have not edited the content of the articles in any way. For this same reason, I ask for your indulgence about the mismatching styles and formatting of the figures, tables, captions, etc.

This work consists of theoretical introductions to each topic investigated, original methods are proposed and evaluated empirically using computer simulations. Results of simulations are exclusively presented and analysis within each published article in each main parts of this dissertation. A unique bibliography has been compiled and included at the end of this document. This bibliography contains all the publications cited or referenced in this work.

Finally, I also have participated to international collaborations on projects conducted in parallel to my main Ph.D. thesis. Some of these collaborations have led to publications of their own, notably on NK-Landscapes and interconnected random Boolean networks. Please refer to the complete publications list at the end of this manuscript if you are interested in reading those particular studies.

Preliminary Work

The Article A of this work, at the end of Part II, was published in 2005 and partially relies on a model and results that we obtained during my Masters project in 2004. Technically, it was published during my actual engagement as a Ph.D. student, although the study began prior to that. Nevertheless, as the first year-and-a-half or so of my doctoral work and the Article B have

been built on this preliminary work, I feel it is appropriate to include it in this document to help make this dissertation self-contained.

Cotutelle

A *cotutelle* is a unique class of international dual-award program, defined under Swiss, Italian and French law. Specific cotutelle agreements have been signed by the conference of university rectors of each country. The objectives of the cotutelle are to offer opportunities to facilitate international research for doctoral candidates, including access to the latest research equipment in two countries/institutions, and to put in place a mechanism to enhance cooperation and collaboration between the researchers and institutions involved. Depending on the specifics of the cotutelle agreement signed by the rectors of the two universities involved, the candidate will receive a single joint doctorate title or one from each institution. I will personally fall into the latter category.

This Ph.D. project has been conducted as a cotutelle between the University of Lausanne, in Switzerland, and the University of Torino, in Italy. The aim was to leverage the specific fields of excellence of both parties. At the University of Lausanne, I was able to find the support I needed in terms of dynamical modeling of Complex Systems and information systems, whereas the University of Torino provided the biological and bio-computational knowledge that we lacked at my home institution.

Roadmap

We've heard that a million monkeys at a million keyboards could produce the complete works of Shakespeare; now, thanks to the Internet, we know that is not true.

Robert Wilensky

From Abstract to Applied: introducing current biological knowledge into computational models

This work is an example of how one can introduce recent discoveries and technological advances into computational models. In the first phase, we worked on improving the performance of a simple paradigm for distributed computation. This was achieved by applying principles inspired by Darwinian evolution to alter the distributed system's topological structure. In the second phase, we use the context of biological genetic regulatory networks and in particular a model proposed in the late 60's. Since then, biology has made tremendous progress and these new discoveries can be used to improve the original model. From the structure of the regulatory network, to timing of the event taking place on it, to the specifics of how gene activate, we have added a significant amount of current knowledge into the original model, studying it, analyzing it and validating it on specific real-life study cases.

Part I: Background & Definitions

This is an introductory part on the common topics we have developed throughout this work: *network based complex systems*. We give the motivations of this project, and provide definitions about the main tool used: *modeling*, and more specifically *agent based modeling*, *network science* and *graph theory*.

Part II: Emergence of Real-Life Topologies

For the first step of this work into modeling, we have applied network science and graph theory concepts to theoretical computational problems with *cellular automata*, a nature-inspired paradigm of distributed computation, and how biologically inspired algorithms can help evolve real-life like network structures. Cellular automata are usually set on regular structures, toroidal, or grid-like, but in our case, we have used *evolutionary algorithm*, a biologically inspired heuristic, to evolve and adapt the interconnections amongst the cells. Results are inspiring as we were able, with very little bias, to obtain topological structures that are similar to those found in complex system networks.

In recent years, the study of network structures emerging from real-life phenomena, natural or man-made, has become a new academic field known as *network science*. We are interested in understanding if and how the topological structure of the network systems participates in their tolerance to faults, which is usually above average. Indeed, *robustness* is a key factor of any social or biological entity, and is oftentimes an intrinsic property of the system and can be related to the specific way the components are linked to one another. Increased robustness is a well-studied phenomenon in communication networks, which are reputed for superior resilience to random failures with minimal links redundancy while very vulnerable to targeted attacks on highly connected hubs. In our evolved topologies, which share properties of both regular and nature-found structures, we obtain the same resistance to errors but also more reliability in case of attacks on nodes with higher connectivity.

Part III: Improving Models of Genetic Regulatory Networks

In this part we progress towards current problems of systems biology, where we study the effect of different artificial topologies on a dynamical model for genetic regulatory networks and its response to failure. We work with the Kauffman's original Boolean model, questioning the random structure of the networks used in this model, and the completely synchronous or asynchronous timing of events. We analyze, compare and contrast results on two types of structures: completely random and *scale-free*. In addition, we introduce a novel, semi-synchronous dynamical behavior, that we believe is closer to biological reality.

This work leads us into the final analytical part of this thesis, where we study how the model operates when using real-life biological regulatory subnetworks. In recent years, small parts of living organisms' actual regulatory networks have been discovered using high-throughput sequencing technologies. This new information is crucial in order to validate our models and add more biological realism.

Part IV: Discussion & Conclusions

Due to the article-based format of this dissertation, the intermediate discussions, conclusions and future work ideas are located at the end of each article. Thus, final reflexions in Part IV are centered around the theses proposed in Section 1.1, addressing each one specifically in the context of the more general contributions of this work to the fields studied.

Connecting Thread

Throughout this work, there is a conscious desire to go from the abstract to the concrete, from the theoretical to the applied, trying to input as much biological knowledge as possible into our abstract dynamical models. There are biological and real-life components to every single part of this thesis; from the evolutionary computation techniques and the type of nature-like topologies resulting, to the type of failures studied in Part II, to the kind of problems tackled, and finally to the real-life regulatory network topologies used in Part III. These biological components overlap in the different parts of this work. This shows just how, observing from different angles, we always find our way back to nature and how fields like computational and systems biology are central to the study of different sociological, economical or scientific fields.

Contents

Imprimatur	iv
Members of the Jury	v
Certificats	vii
Abstract	xxi
Synopsis (in French)	xxiii
Acknowledgements	xxv
Preface	xxvii
Roadmap	xxix
I Background & Definitions	1
1 Motivation	3
1.1 Theses	3
2 Introduction & Definitions	5
2.1 Complex Systems	5
2.2 Network Science	6
2.2.1 Formal Definitions	7
2.2.2 Network Structures	9
2.3 Modeling & Simulation	12
2.3.1 Agent-Based Simulation	14
2.4 Disruption, Failures, Faults & Robustness	15
2.5 Failure Types	15
2.5.1 Probabilistic Faults	16
2.5.2 Measuring the Effect of Perturbations	16
II Emergence of Real-Life Topologies	19
3 Evolutionary and Genetic Algorithms	21
3.1 Genetic Algorithms	22
3.1.1 Encoding	22
3.1.2 Initial Population	23
3.1.3 Phenotype Evaluation	23
3.1.4 Genetic Operators	23
3.1.5 Structured Evolutionary Algorithm	26

4 Cellular Automata for Computation	29
5 Articles in this Part	31
A Evolution and Dynamics of Small-World Cellular Automata	33
A.1 Introduction	33
A.2 Useful definitions for graphs	34
A.3 The cellular automata problems	36
A.3.1 The majority task	36
A.3.2 The synchronization task	37
A.4 Artificial evolution of small worlds	38
A.4.1 Evolution of graphs for the density task	38
A.4.2 Evolution of automata graphs for the synchronization task	42
A.5 Limiting the number of shortcuts	43
A.5.1 Task flexibility of the evolved networks	44
A.6 Robustness in the presence of random faults	45
A.7 Conclusions	46
B Performance and Robustness of Cellular Automata Computation on Irregular Networks	49
B.1 Introduction	49
B.2 Small-World and Scale-Free Graphs	50
B.3 Cellular and Networked Automata	51
B.4 Generalized Automata Networks Computations	53
B.4.1 Tasks Performance on Generalized Automata Networks	54
B.5 The Effects of Noise and Failures	58
B.5.1 Probabilistic Updating Perturbations	58
B.5.2 Permanent Link Failures Perturbations	63
B.6 Conclusions	66
III Improving Models of Genetic Regulatory Networks	69
6 The Biology Behind It	71
6.1 Genetic Regulatory Networks	72
7 Modeling Genetic Regulatory Networks	75
7.1 Random Boolean Networks	76
7.2 Extension of Random Boolean Networks	77
8 Biologically Inspired Faults	79
9 Articles in this Part	81
C Dynamics of Unperturbed and Noisy Generalized Boolean Networks	83
C.1 Introduction	83
C.2 Classical Random Boolean Networks	84
C.3 From Random to Generalized Boolean Networks	85
C.3.1 Discrete State Approach	86
C.3.2 Random vs Scale-Free Networks	86
C.3.3 Timing of Events	89
C.4 Methodology and Simulations	90
C.5 Finding Attractors	91
C.5.1 Number of Attractors	91

C.5.2	Variety of the Attractors	92
C.5.3	Length of the Attractors	93
C.5.4	Scaling	94
C.6	Fault Tolerance of Random Boolean Networks	95
C.6.1	The Effect of Perturbation	97
C.6.2	Derrida Plots	100
C.7	Conclusions and Future Work	101
D	Additive functions in Boolean networks	103
D.1	Introduction	104
D.2	Yeast and Embryonic Stem Cells Regulatory Networks	105
D.3	Random Boolean Networks Modeling	106
D.3.1	Regimes of RBNs and how to identify them	106
D.3.2	Criticality Distance	108
D.4	Modeling the Yeast and the Embryonic Stem Cells Regulatory Networks	109
D.5	Regimes Characterization in Real-Life Networks	110
D.5.1	Validation of the Model on a Network with Known Update Rules	111
D.6	Enhancer Proportion vs. Threshold of Critical ADA Networks	113
D.7	Dynamical Behaviors of Real-Life Regulatory Networks	115
D.7.1	Simulation of the Embryonic Stem Cell Regulatory Networks	116
D.7.2	Simulation of the Yeast Cell-Cycle Regulatory Networks	117
D.8	Resilience to Small Perturbations	118
D.9	Discussion, Conclusions and Future Work	118
IV	Discussions, Conclusions & Future Perspectives	121
10	Conclusions & Future Perspectives	123
10.1	Conclusions and Scientific Contributions	123
10.2	Future Perspectives	124
10.3	Final Considerations	125
V	Appendixes	127
A	Tools	129
B	UML Class Diagram - Jcell	131
C	UML Class Diagram - RBN	137
D	UML Class Diagram - eaRBN	141
	List of Publications	147
	Bibliography	150

Part I

Background & Definitions

Chapter 1

Motivation

To steal ideas from one person's work is plagiarism; to steal from many is research

Albert Einstein

When trying to solve a new problem, mankind draws inspiration from its surroundings. We have always tried at first to mimic how nature faces its challenges to address our own. This work is no different. In fact, the very problems we study are posed by nature and we apply methodologies inspired by nature to understand and to address them. To do just that, we build dynamical models. We could have chosen to apply mathematical systems of differential equations, a strictly theoretical option. Instead, we have chosen a more empirical method: agent-based models (ABMs) [Axe97] that we will define in Section 2.3.

What problems do we address? In general terms, we study what intrinsic properties of complex network systems make them both evolvable, and yet stable to small perturbations. Beyond the actual interaction amongst the components of the studied network systems, we analyze the effect of the topology and how the structure of the network as a whole contributes to these outstanding properties of evolvability and robustness. In order to study the importance of the global structure beyond the particulars of each link, we have chosen to study two different, yet comparable, examples of network based Complex Systems (CS). Firstly, we have used cellular automata (CA) for computation (introduced in Section 4). These are very simple, yet powerful, instances of distributed computation able to tackle prototypical problems. As these toy examples have been thoroughly investigated and are very well understood, they are excellent tools we can use to validate new paradigms. In a second phase, we have chosen a systems biology approach, modeling regulatory networks of biological organisms (introduced in Section 6.1). Modeling biological phenomena is much more challenging, as these systems are much more complex and thus less well known. They offer however the advantage of providing an applied environment, where findings can be used beyond their theoretical value.

Naturally, we do not compare directly these two systems. We merely use them as the contexts in which we study the effect of the topology on the global dynamics and on the tolerance to faults (introduced in Section 2.4). Nevertheless, both cellular automata and genetic regulatory networks share two common attributes: they are instances of dynamic complex systems (introduced in Section 2.1) and they are modeled as graphs (or networks).

1.1 Theses

Beyond the general motivation offered above, we detail hereafter the research questions and theses that we will address and develop.

First Thesis: In the context of structured cellular automata computation on prototypical tasks, evolutionary algorithms (EA) have proven able to evolve rules that outperform hand-made rule for the majority problem. We hypothesize that EAs can also evolve the structural topology of the cell connections by rewiring their links to increase the performance using a basic majority-based rule.

Second Thesis: Complex network systems that have grown in an organic or unsupervised incremental manner, such as telecommunication or social networks, show outstanding resilience to random failures. The structures evolved using EAs as support for CA computation show numerous topological similarities with these networks. We believe that they also share their exceptional ability to continue functioning despite transient probabilistic faults.

Third Thesis: Models of biological systems are routinely improved by changes or additions that bring them closer to reality. We hypothesize that Boolean models of genetic regulatory networks (GRN) models can be enhanced further by refining the sequence of the events taking place with an update timing that is more biologically plausible than that of fully synchronous or asynchronous systems.

Fourth Thesis: Genetic regulatory networks are both flexible enough to sustain Darwinian evolution and yet stable enough to sustain life after small perturbations, such as mutations or transcriptional errors. We are of the opinion that this robustness is not only due to the physical and chemical properties of biological compound, but also to an intrinsic stability provided by the structural topology of the network. Models of biological systems ought to share this integrity and can therefore be used to study the robustness.

Fifth Thesis: The recent developments in high-throughput genetic sequencing technologies have provided important amounts of data of a quality never obtained in the past. From this data, parts of regulatory networks of living organisms have been inferred. Additionally, the actual way genes regulate each other is now more commonly known. We speculate that all this new information can be included in the GRN models to ameliorate them to a point closer than ever to reality. We also believe that state-of-the-art model should outperform all others in terms of robustness and evolvability.

Chapter 2

Introduction & Definitions

My mother always said, life was like a box of chocolates, you never know what you're gonna get.

Forrest Gump

In this chapter, we introduce the concepts that are recurrent throughout the entire dissertation. These fundamental notions oftentimes overlap and use each other in their definition, and are to be considered a set of tool useful to understand the problems posed in the next chapters.

2.1 Complex Systems

A complex system (CS) is a dynamic system composed of (heterogeneous) interconnected constituents, where the state, the interactions, or any other component of the system is susceptible to change overtime. The behavior of CSs goes beyond that expected of the linear sum of its parts. This characteristic of CSs is called *emergence* [Wea48]. The first examples that comes to mind are ants and ants' colonies. The prowess of the entire colony cannot simply be explained by the strict addition of each individual ant's contribution. Indeed, there are additional social and chemical components that can explain the behavior of the group as an entity. CSs are used to model and simulate human economy, social dynamics, biological and ecological phenomena, modern energetic and telecommunication infrastructures.

The term *complex* is not to be confused with *complicated*. Some systems can be considered as complicated because they are constituted of a great number of heterogeneous parts. However, their are not complex as they do not exhibit any emergent behavior that cannot be explained by the strict cumulated actions of these parts. Moreover, complicated systems are built according to predefined blueprints and for a purpose. This a priori intention does not exist in complex systems.

Complex systems is also considered a transversal field, that is studied across many areas of natural, mathematical, as well as social sciences, and include disciplines such as cybernetics, systems biology or ecology, network science and systems theory. A number of properties that are common to all CSs are highlighted in a special issue of *Science*:

- A complex system is a highly structured system, which shows structure with variations [GK99];
- A complex system is one whose evolution is very sensitive to initial conditions or to small perturbations, one in which the number of independent interacting components is large, or one in which there are multiple pathways by which the system can evolve [WI99];
- A complex system is one that by design or function or both is difficult to understand and verify [WBI99];
- Complex systems are systems in process that constantly evolve and unfold over time [Art99].

Complex systems go hand in hand with modeling and simulation (introduced in Section 2.3). Simple phenomena, such as mechanical physics, can be explained and predicted using sets of equations, possibly differential. Models and simulation are used in CSs to predict patterns [vH78]. Additionally, CSs are in general associated with networks and graph theory (introduced in Section 2.2), and they are composed of interconnected parts. Each vertex of the network represents one of the components and each edge, possibly directed, represents the interaction between two components, or the influence of one component on another.

A common misconception is that complex systems are equivalent chaotic systems from *Chaos theory* described in Strogatz's works [Str01a, Str01b]. Chaos can be seen as extremely complicated information, rather than the absence of order, also both chaotic and complex systems are extremely dependent on initial conditions. Nevertheless, chaotic systems are deterministic; with perfect knowledge of the initial conditions and of the context of an action, the course of this action can be predicted in chaos theory. On the contrary, complex systems have a non-deterministic component [Pri97]. The emergence of complexity theory shows a domain between deterministic order and randomness that is complex [Cil98]. This is referred as the *critical* or *edge of chaos* [Bak96].

2.2 Network Science

Networks are formally described by graph theory tools and properties, and are the essence of incredibly numerous scientific, sociological and technological phenomena. Networks are used to give an abstract representation of interacting objects or individuals, where each object is placed on a node and links represent relationships between two objects. These representations, or models, are in turn used to study the static and dynamic behavior of the system as a whole.

In this work, as we do not deal with theoretical graphs, we will be using the terms network and graph interchangeably. Networks are composed of *vertices* or *nodes* that are linked to one another by *edges* (see example in Figure 2.1a). Edges may be *directed* (Figure 2.1b), i.e. point from one vertex to another without reciprocity.

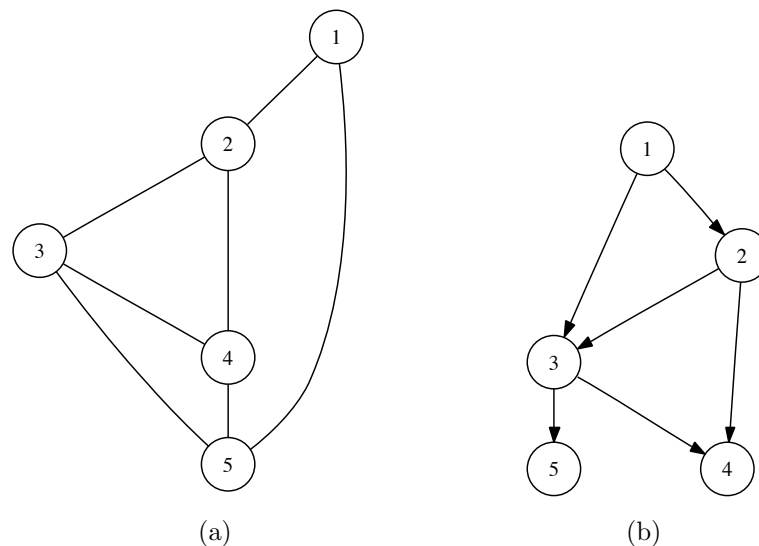


Figure 2.1: Examples of graphs. (a) Undirected graph and (b) directed graph.

We will use networks as support structures for our complex system models. This is an extremely common manner of representing interaction amongst the components of a system. Therefore, when considering a model for human social interaction, one way would be to place each individual on a node and draw an undirected link between two nodes that hold people knowing each other. In general human acquaintances, we can safely assume that we do not need directed edges as

reciprocity is almost a given. Other phenomena, such as sexual attraction within human social structures for instance, require extra information, for instance, the direction in which the attraction goes. In fact, *sociometry* has emerged after Moreno's work [Mor53], which contains a representation of the social structure of a group of elementary school students. The boys were friends of boys and the girls were friends of girls with the exception of one boy who said he liked a single girl. The feeling was not reciprocated. This type of *directed graphs* does not intrinsically prevent or forbid reciprocity or self-loops. Finally, extra information in the form of *weighted edges*, such as the strength of a bond in chemical binding representation, shows how many electrons are shared between to atoms within a molecule, or the distance in kilometers between two cities. Then any combination of weights and direction is possible. Although not used in this particular work, graphs can be inhomogeneous in the type of their vertices and/or edges. In the examples above, each vertex was of the same type of every other. However, we can imagine a "pet ownership" network, where there are two types or nodes (owners and pets) and two types of edges (ownership and acquaintance).

A new scientific discipline, called *network science* has emerged in recent years, and is now blossoming. Network science examines the interconnections among diverse physical or engineered networks, information networks, biological networks, cognitive and semantic networks, and social networks. This field of science seeks to discover common principles, algorithms and tools that govern network behavior. The American Research Council defines Network Science as "the study of network representations of physical, biological, and social phenomena leading to predictive models of these phenomena." In other words, it is the science that studies complex relational data as network models. One of the earliest works in the field was Euler's "Seven Bridges of Koenigsberg" written in 1736, where he mathematically resolved the problem of crossing every bridge once only by abstracting each bridge as a vertex and the path between two bridges as an edge. The resulting mathematical structure is called a graph. This is believed to have laid the foundation of graph theory, of static modeling, and introduced the idea of *topology*. In this work, we will use the words network *structure* or *topology* interchangeably, although the strict mathematical meaning of the two concepts may differ.

2.2.1 Formal Definitions

After giving a taste for what networks and graphs are, let us mathematically define a few notions and terms from graph theory. Admittedly, a similar, yet shorter section exists in the first article A.2, and we will add a few more definitions below. A more detailed account of graph properties can be found for example in the works of Newman [New03], Watts [Wat99], and Albert-Barabási [AB02].

Graph: Let V be a nonempty finite set called the set of N vertices, and let E be a binary relation on V (i.e. a set of unordered pairs of vertices). $G = (E, V)$ is called an undirected or directed graph and V is the set of vertices of G . E is the set of undirected, respectively directed edges between two vertices $u, v \in E$. When vertices (u, v) of an undirected graph form an edge they are said to be *adjacent* or *neighbors*. If the graph is directed, the terms *incoming-neighbors* or *outgoing-neighbors* are used to specify the direction.

The *degree* k of a vertex is the number of edges impinging on it (or, equivalently, the number of its neighbors). The average degree \bar{k} (or $\langle k \rangle$) of a graph is the mean value of all the vertex degrees in G . In the case of a directed graph, each vertex has two degrees: an *in-degree* k_{in} and *out-degree* k_{out} , which are respectively the number of incoming-neighbors and the number of outgoing-neighbors. Please note that the average degree of the graph is defined as $\bar{k} = \bar{k}_{in} = \bar{k}_{out}$.

A *regular graph* is defined as a graph where all the vertices have the same degree K . In this case, the average degree is simply called the degree of the graph $K = \bar{k}$.

The *degree distribution* is the probability distribution of the vertices degrees $P(k)$ over the whole network. In other words, it is the function that gives the probability that a vertex will have a particular degree k . Regular graphs have a *delta* function peaking at K : $P(K) = 1$ and $P(k \neq K) = 0$. Irregular directed graphs have two different degree distributions: $P_{in}(k)$ and $P_{out}(k)$.

A *path* from vertex u to vertex v in a graph is a sequence of edges that are traversed when going from u to v with no edge traversed more than once. The length of a path is the number of edges in it (see the *distance* below). The shortest path between two vertices u and v is the path with the smallest length possible joining u to v . This definition is valid in both directed and undirected graphs.

In graphs, the notion of *distance* does not always bear the Euclidian sense as graphs are non-spatial mathematical objects. Instead, the distance l_{uv} between two vertices u and v of a network is measured in terms of number of edges traversed on the shortest path between u and v .

A graph is *connected* if there is a path between any two vertices. A *completely connected* undirected graph G with $|V| = N$ vertices, usually labeled K_N , has an edge between any two vertices (Figure 2.2). Its total number of edges is $|E| = N(N - 1)/2$. A *sparse* graph is a graph where $|E| \ll N(N - 1)/2$, as opposed to a *dense* graph where $|E| \simeq N^2/2$. A subgraph of $M < N$ vertices that are completely connected is called a *clique*.

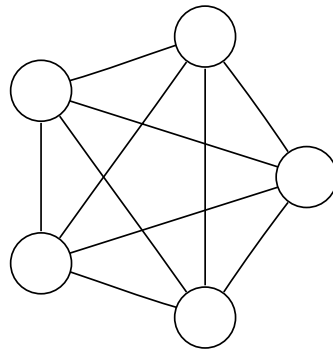


Figure 2.2: Example of a complete graph K_5 .

Clustering Coefficient: Informally, it is the probability that two nodes neighbors of a third are also each other's neighbors. Let us take a particular node j in a graph, and let us assume that it has k edges connecting it to its k neighboring nodes. If all k vertices in the neighborhood were completely connected the number of edges would be equal to $k(k - 1)/2$. The *clustering coefficient* C_j is defined as the ratio between the E_j edges that actually exist between the k neighbors and the number of possible edges between these nodes:

$$C_j = \frac{2E_j}{k(k - 1)}$$

Thus, the clustering coefficient C of the entire graph G is the mean of the C_j over all nodes $j \in V$:

$$C = \frac{\sum_{j=1}^N C_j}{N}$$

The clustering coefficient of a random graph is simply the probability p of two edges u and v to be connected. For a regular lattice, C is given by the following formula:

$$C = \frac{3(K - 2)}{4(K - 1)}$$

C is thus independent of N for a regular lattice, and approaches $3/4$ as K increases.

Characteristic Path Length: The characteristic path length, or *average path length*, L is defined in [Wat99] as the means of the shortest path lengths connecting each vertex $v \in G$ to all other vertices.

$$L = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j>i}^N l_{ij}$$

2.2.2 Network Structures

Regular Structures

Networks can be classified in three main categories: *regular*, *irregular*, and *random* structures. Regular topologies (Figure 2.3) are commonly used in mathematical models and usually have a constant connectivity K for all nodes, a high clustering coefficient and long characteristic path lengths.

The mono-dimensional *lattices* (or *ring*) can have different values of constant connectivity, for example, in Figure 2.3a and 2.3b) the networks show a connectivity $K = 2$, and $K = 4$ respectively. Although it seems regular, the bi-dimensional lattice in Figure 2.3c exhibits several different degree for different nodes: central nodes have degree of $k = 4$, nodes on the edges have $k = 3$, and nodes in the corners of the structure have only a degree of $k = 2$. To obtain a regular structure from this 2D structure, the edges have to be wrapped around in order to obtain a torus (Figure 2.3d).

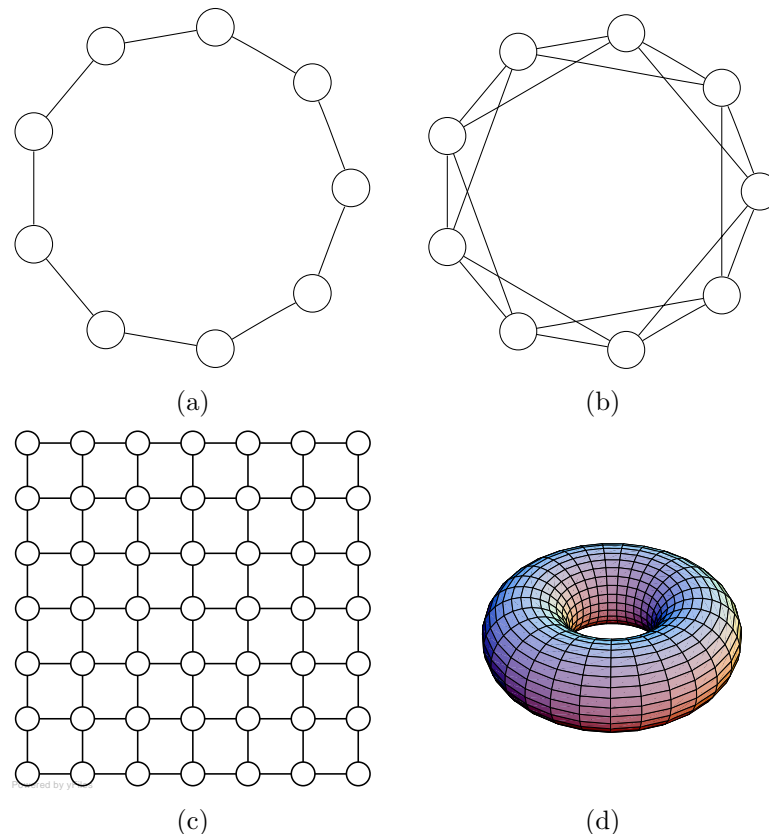


Figure 2.3: Examples of regular graphs. (a) one-dimensional lattice with $k = 2$. (b) one-dimensional lattice with $k = 4$. (c) two-dimensional regular lattice. (d) a torus.

Many more regular structures exist in networks, but do not exhibit the some of the properties. For instance, a *n-tree*, which is a hierarchical directed network where each node has exactly 1

incoming edge and n outgoing edges, or inversely. Because of this restriction, links only go from “upstream” nodes towards “downstream” nodes, there can be no loopbacks and thus the clustering coefficient of trees is zero.

Random Graph

A *random graph* is a graph in which pairs of nodes are connected at random. Different random graph models with different construction methods result in various probability distributions. The most commonly studied is the Erdős-Rényi that draws links given a probability p [ER59a]. Consequently, the total number of edges in a random graph is a random variable $|E| = p(N(N-1)/2)$. In this case, the clustering coefficient is the probability p , and the characteristic path length $L \propto \log N$ is much shorter than that of regular lattices with the same number of nodes, see for instance [WS98]. The degree distribution of random graphs follows a *binomial* distribution centered around $\bar{k} = Np$ pictured in Figure 2.4 below.

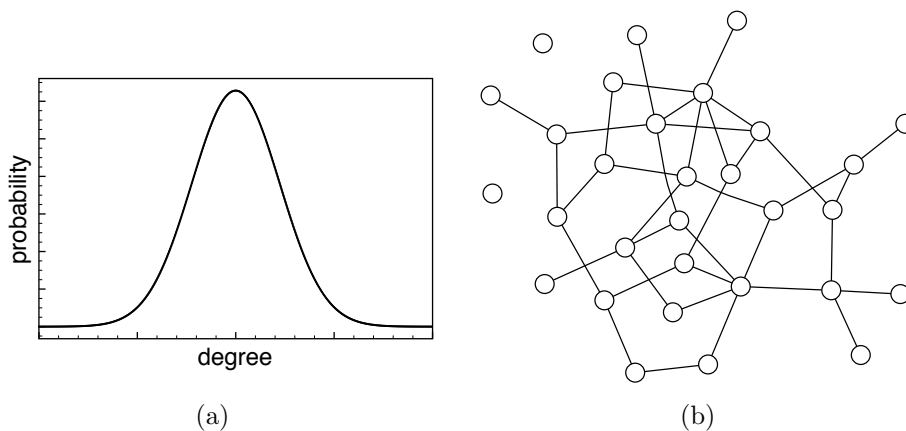


Figure 2.4: Random graphs. (a) binomial degree distribution. The abscissa axis is the degree of a node and the ordinate axis is the probability of that degree to occur within the graph. (b) an example of random graph.

Alternatively, the binomial degree distribution of random graphs can be approximated with a *normal* or a *Poisson* degree distribution $P(k) \simeq \bar{k}^k e^{-\bar{k}}/k!$, where \bar{k} is the mean degree, or a *delta* distribution as in a fixed-degree random networks, same as that of regular graphs.

Irregular Networks

According to Watts and Strogatz’s [Wat99, WS98] observations on experimental data, most real-life networks, in the biology, sociology, economy, as well as man-engineered networks, have mathematical properties that set them apart from both regular lattices and random graphs. Indeed, they found that real-world networks tend to be highly clustered (like lattices), but have small average path lengths (like random graphs). From their work, we will explore the Watts-Strogatz *small-world* networks in the sections below.

Small-Worlds Phenomenon: The *small-world theory* states that everyone in the world can be reached through a chain of social acquaintances that is approximatively 6 people. It gave rise to the famous phrase: “six degrees of separation”.

The original idea can be attributed to Stanley Milgram and thoroughly described in [Mil67] in which he conducted research among the US population at large. In his first “small-world” experiment, Milgram sent 60 letters to various recruits in Wichita, Kansas, who were asked to forward the letter to the wife of a student living at a specified location in Cambridge, Massachusetts. The participants could only pass the letters by hand to personal acquaintances that they thought

might be able to reach the target - whether directly or via a “friend of a friend”. While fifty people responded to the challenge, only three letters eventually reached their destination. In conclusion, one of the letters in this initial experiment reached the recipient in just four days, but only 5% of the letters successfully “connected” to their target. In two subsequent experiments, chain completion was so low that the results were never published. On top of this, researchers have shown that a number of subtle factors can have a profound effect on the results of “small-world” experiments. Studies that attempted to connect people of different races or incomes showed significant asymmetries. Despite these complications, a variety of novel discoveries did emerge from Milgram’s research. After numerous refinements of the apparatus (the perceived value of the letter or parcel was a key factor in whether people were motivated to pass it on or not), Milgram was able to achieve completion rates of 35%, and later researchers pushed this as high as 97%. If there was some doubt as to whether the “whole world” was a small world, there was very little doubt that there were a large number of small worlds within that whole. For those chains that did reach completion the number 6 emerged as the mean average number of intermediaries and thus the expression “six degrees of separation” (perhaps by analogy to “six degrees of freedom”) was born. In addition, Milgram identified a “funneling” effect whereby most of the forwarding (i.e. connecting) was being done by a very small number of “stars” with significantly higher-than-average connectivity: even on the 5% “pilot” study, Milgram noted that “two of the three completed chains went through the same people”. This was also the premises to the famous study of the Hollywood actors study in which Kevin Bacon was established as a central pillar of the network thanks his numerous collaborations.

Since the social model of small worlds has received a great amount of attention, there have been an important number of studies conducted on the topic of small world as a mathematical model [New03, DM03]. We inspire our preliminary work from Watts’ book [Wat99], which is a remarkably readable book where there is an interesting definition of a particular subset of the random-like networks family, which Watts calls the small-world graphs. Watts and Strogatz’s [Wat99, WS98] introduced the concept of small-world networks, in which most pairs of vertices seem to be connected by a short path through the network thanks to *shortcuts*. The existence of short paths between any pair of nodes has been found in networks as diverse as the Internet, airline routes, and neural and metabolic pathways, among others. The presence of short paths is also a characteristic of random graphs, but what sets these real networks apart is a larger clustering coefficient than that of random graphs having a comparable number of nodes and links. The low clustering coefficient tends to prove that there is more local structure in these networks than in plain random graphs. Watts and Strogatz use an algorithm to synthetically build small-world networks starting from a regular ring structure and “rewiring” to create shortcuts across the network. As this method does not add or remove edges in the network, therefore, the average degree \bar{k} of the networks remains identical to that of the original ring. A detailed algorithm to construct small-world network starting from regular ring structure can be found at the end of section A.2.

It is worth mentioning that small-world graphs as defined by Watts and Strogatz are not actually structurally representative of networks found in the real world, which are often, but not always, of the scale-free type, if anything. However, they are easy to construct and measure and, for the purpose of the artificial CA problems with which we are concerned, they are a perfectly legitimate choice for studying the influence of the network structure on the dynamics.

Scale-Free Networks: Scale-free topologies seem common in real-life [AB02]. They distinguish themselves from both random and regular network by the presence a few highly connected vertices, called *hubs*, and the vast majority of the remaining vertices have a low degree. Finally, if the degree reaches $k = 1$ the nodes are called *leaves*, although these are more customary of *tree* or *hierachical* networks [RB03]. Therefore, the degree distribution $P(k)$ is very skewed, usually following a power-law distribution: $P(k) \sim ck^{-\gamma}$, with c and γ positive constants typically between $2 < \gamma < 3$, also called Pareto distribution (see 2.6). In figure 2.6, the vertices’ size is proportional to their degree so it is clear which are hubs and which are not.

This type of structure can be found in communication networks, such as telephone (cellular or

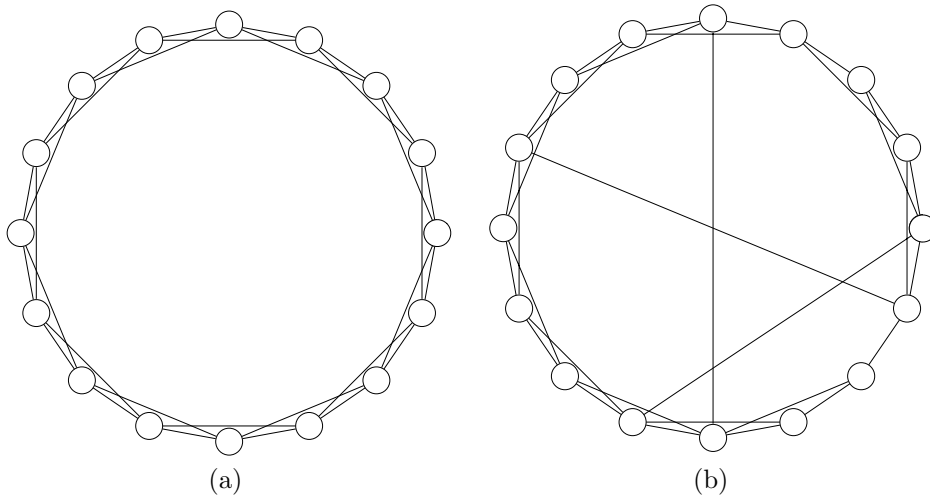


Figure 2.5: Small-World Graph construction. (a) Regular one-dimensional lattice with $k = 4$. (b) A small-world graph obtained by randomly rewiring some of the nearest-neighbor links.

landline), the Internet, or the World Wide Web, but also airline routes. They were first outlined by de Solla Price in 1965 when he analyzed the patterns of citations within scientific publications [dSP65]. The term “scale-free” came about many years later, but he already discovered the heavy-tailed degree distribution of citation networks. A few years later, he proposed an explanation as why and how this phenomenon occurred [Pri76]. This idea is a pillar of Barabási-Albert’s *preferential attachment* algorithm [AB02] that constructs an undirected scale-free graph. This algorithm is detailed in the second article, section B.2.

Alternatively, the *configuration model* can be used to build network models of the scale-free type. This is a top-down approach, detailed in the second article at the end of Section B.2, where the degree distribution $P(K)$ is defined beforehand:

$$P(k) = \frac{1}{Z} k^{-\gamma}$$

where the normalization constant $Z(\gamma) = \sum_{k=1}^{k_{max}} k^{-\gamma}$ coincides with Riemann’s *Zeta function* [Rie92] for $k_{max} \rightarrow \infty$. In the same section, we also offer a refined version of the configuration model that allows to obtain a specific average degree \bar{k} .

Thus hubs are both the strength of scale-free networks and their weakness. The power-law distribution influences not only the topology, but also the dynamics of the phenomena taking place on them. A hierarchy exists between bigger and smaller hub, and in turn to leaves. This feature renders the system highly tolerant to random failures. Indeed, for equally distributed random failure in nodes, there is a much higher probability to hit a leaf, or at least a node with a low degree. Therefore, the likelihood to affect a hub is almost negligible. And even if such an event should occur, the graph will probably remain connected. On the contrary, if attacks are targeted against hubs and even a few of them should fail, the whole system will break down, as it is turned into a set of isolated sub-networks.

2.3 Modeling & Simulation

In sciences, a model is a simplified representation of a complex observable phenomenon using assumptions and conditions to make predictions. The models can be mathematical, graphical and/or conceptual, and are a way of understanding the broken down elements of a whole, focusing on the components of interest to the observed phenomenon in their simplest form. Nowadays, modeling is an inherent part of any scientific activity in any field. Models are usually generated for

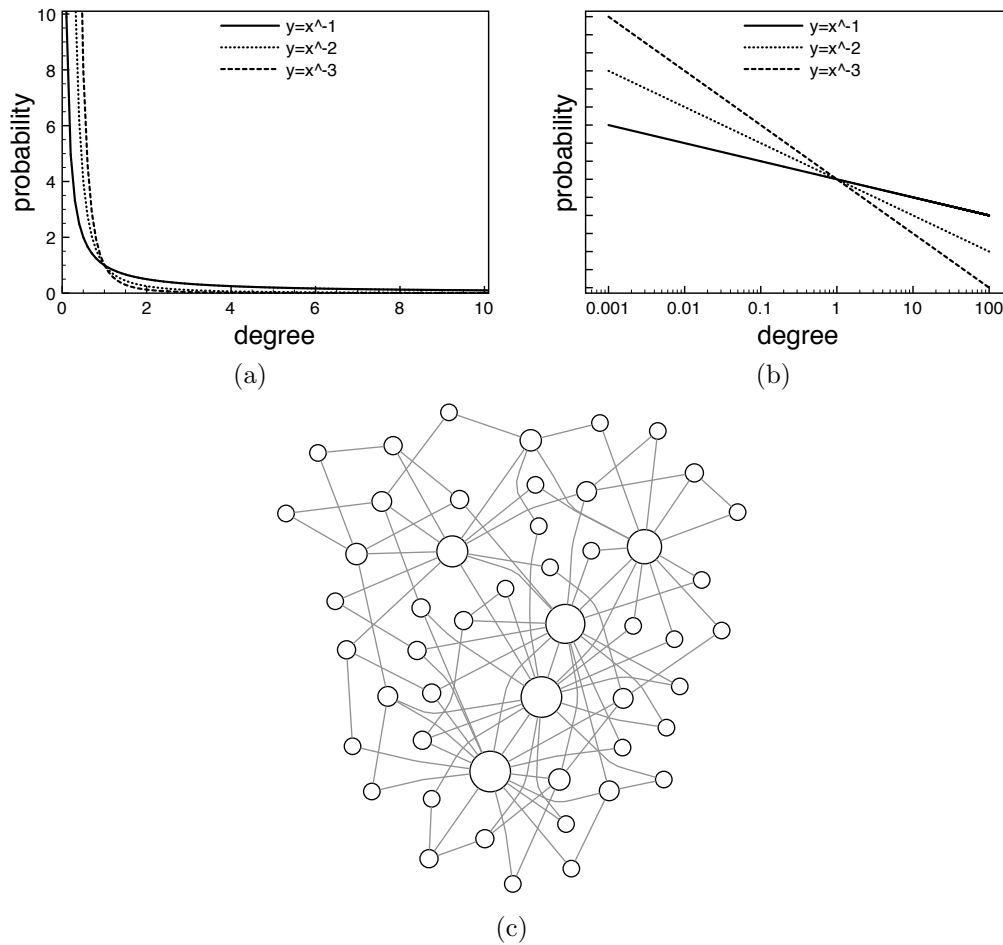


Figure 2.6: Power-law degree distribution. The abscissa axis is the degree of a node and the ordinate axis is the probability of that degree to occur within the graph. (a) *lin-lin* axis system, (b) *log-log* axis system. (c) an example of scale-free graph where the vertices' size is proportionate to their degree.

two different purposes: to predict beforehand or to study afterward the behavior of an empirical objects, phenomena, and physical processes. Models are used to predict the weather, based on previous experience and data, to study the airflow around an airplane's wing without actually constructing it, or to estimate the forces at work on the architecture of a dam.

Most models involve a certain amount of mathematics. They can be completely abstract representations taking the form of sets of algebraic equations, but oftentimes, they are completed by a visual aid such graphic figures and computer animations. Although not strictly necessary, computation is becoming a deep-rooted commodity in models, and their use is proportionate to the complexity of the phenomena and the level of details desired. Indeed, a set of linear equations is sufficient to understand simple events in mechanical physics, such as a wooden cube sliding down a slope at constant speed. Nevertheless, if the level of granularity is reduced, even a common problem becomes highly dimensional, with an important number of variables, as in the study of the aerodynamics of an accelerating object in a fluid. For the scientist, a model is also a way in which the human thought processes can be amplified [Chu68]. Models rendered in computer software allow scientists to leverage computer power to simulate, visualize, manipulate and gain insight on the systems being represented. Moreover, they allow the study of phenomena that would be impractical or impossible to observe otherwise, or under conditions impossible to create experimentally. Measurements made directly on controlled experiments will always be more

accurate that that of models, as they do not rely on assumptions. Therefore, models will never be a complete substitution for direct measurements and experimentation. Nevertheless, models are fundamental scientific tools that help study, gain insight, and understand complex reality.

The term *simulation* is used to designate the dynamic implementation of a model over time [Uni01]. Simulation can be either executed in the real world or on computers, and bring the model to life. It is then possible to study the behavior of a particular element or of the systems as a whole under different sets of conditions and assumptions. Simulations are critical tools, especially when dealing with complex systems that may have a non-deterministic component and are therefore highly sensitive to initial conditions. Simulations are generally used in complex systems to explore the set of parameters or initial conditions of the model. Repeating the same simulation can provide statistically significant results. The amalgam of simulation and visualization is to be avoided. Simulations are usually completely independent of visualization. On the other hand, visualization oftentimes relies on one particular instance of simulation that is of particular interest in the results or in the phenomena behavior. In these cases, visualization is used if it is believed that it will help gain more knowledge or understanding of the results. The results and the models themselves are evaluated primarily, but not only, on their consistency with empirical data, when available. When results do not agree with the observations, the model is either adapted or rejected. When fitting the data, the models are evaluated on their capacity to reproduce the results, on the quality of the insight given, simplicity, efficiency, cost, etc.

In this work, we have focused our efforts on one type of modeling described in the next section.

2.3.1 Agent-Based Simulation

Agent-based modeling (ABM) or *agent-based simulation* (ABS), depending if we are talking of the model itself as a theoretical entity or the dynamic implementation of the model in time, are multiple systems used to computationally simulate the actions and interactions of autonomous agents with a view to assessing their effects on the system as a whole. In this work, we are always generating models in order to use them in simulation, the distinction between *model* and *simulation* is therefore not important. Such methods are commonly used in fields such as game theory, complex systems, emergence, computational sociology, multi-agent systems, and evolutionary computation. The reason why they are so widely used is that they are extremely flexible. Indeed, the degree of granularity with which one chooses to model a phenomenon is not a limiting factor for ABMs. The number of constraints in term of timing, dimensionality or the agents' behavior itself is virtually limitless. With ABMs, one can always make a model more realistic.

ABSs use simple agents' actions and interaction between agents to simulate complex system behavior to study the objects modeled, and model the emergence of complex behavior and higher order patterns. Single agents are by definition not incapable of reaching cognitive closure. In other words, they are usually implemented as willing to perform a given task, but with a local view of the problem and restricted interactions capabilities. The problems typically performed by these distributed systems could often be solved by a system with central controls and a global view of the problem or situation. For example, if agents are used to simulate the behavior of organisms trying to escape a maze, each agent will be implemented with a restricted visibility and interaction with others to make decisions on the direction to take. This fact results in suboptimal behavior, yet realistic with respect to the organisms that are simulated. Naturally, a birds-eye view of the problem would result in immediate and optimal solving of the problem.

The types of problems are usually dynamical in nature, where the global behavior of the system prevails over that of the individual agents, with parameters or conditions that change overtime, even during the simulation. This would be the case of simulating a flying plane with its mass diminishing as the fuel is used up. ABSs are naturally well suited to simulate interacting entities, even of different types, as in rumor spreading within a population. The interactions themselves or the coupling strength among the agents can also vary dynamically during the simulation. As in any complex systems modeling, some elements of randomness can be included. ABSs are especially well adapted to problems with a spatial component, where the actual position and/or distance of mobile agents matter and change overtime. Finally, the effect of how changing one individual

impacts the entire system can be studied, with different scenarios, probability value, etc.

In ABSs, agents obey a set of rules that can be shared by all agent or that is unique to each individual. ABSs are therefore extremely flexible as agents can behave, interact or move independently. This makes it very easy to have one or more agent types, and one or more instances of each agent types. Further simplifications commonly accepted in ABSs are discrete timing, where agents can only act at given time-steps. Also, agents actions are oftentimes instantaneous events. Nevertheless, examples of derogations to both of these conventions are frequent.

The major drawback is combinatorial explosion of the parameter space, thus the computational power necessary to ABMs increases exponentially with the number of conditions. This is a problem that no expansion of the computational power can solve. Sampling statistically relevant sub-ensembles of all possible parameter values usually overcome this problem. Also, results are not always individually reproducible, they are only statistically significant. Finally, the behavior of individual agents, should it be of interest, is oftentimes difficult to identify and extract from the population.

2.4 Disruption, Failures, Faults & Robustness

Failures in systems can occur in various ways, and the probability of some kind of error increases dramatically with the complexity of the systems. They can range from a one-time wrong output to a complete breakdown and can be system-related or due to external factors. Social structures, incrementally grown man-made networks and living organisms are robust to a great variety of changes, and since we study models of the dynamics of interactions within these systems, it is interesting and legitimate to ask questions about their fault tolerance aspects. Moreover, in many biological organisms, researchers use targeted attacks on designated components in order to understand their role in the system as a whole. A common example is the *gene knockout* experiment used by geneticists (introduced in Section 8). The robustness, or fault tolerance, of a system is a measurement of its ability to continue to perform its task in the presence of faults, failures or disruptions.

2.5 Failure Types

Under the umbrella term of *failure*, we regroup all types of disruptions to the system. These can be an actual malfunction within the system, or the voluntary and planned outage or removal of a number of components. In this section, we briefly discuss a few properties of failures and their opposite. Indeed, in general terms, faults and failures in complex systems models can be classified using a combination of the properties hereafter. Each “category” below has an opposite and we give a few intuitive examples of failure or disruption to illustrate cases. Gray areas where the fault is classified between the two values, although not excluded a priori, are extremely rare.

Transient Vs. Permanent: This property designates the duration in time of the failure taking place on the system. *Transient* failures are limited in their duration in terms of time in continuous systems or time-steps in discrete ones, such as those studied in this work. On the other hand, *permanent* faults those that once they occur, they remain indefinitely.

Reproducible Vs. Irreproducible: This type of failures refers usually to transient faults. *Reproducible* failures can be replicated at will in order to be studied or understood as in most scientific methods. *Irreproducible* failures occur without prompting and are usually impossible to replicate, although this latter category would be the most interesting to analyze in order to be avoided. They should not be confused with *asynchronous* or *probabilistic* failures, where the faults itself can be replicated, only the timing of its occurrence is uncertain.

Targeted Vs. Non-Targeted: The component of the system that suffers of a failure can be the victim of a *random* or *probabilistic* fault that has not been voluntarily targeted against it. On the contrary, malignant entities can target attacks against the most important component(s) of the system in order to cause maximal disruption to the entire system. Attacks can be planned according to several criteria, but are usually limited to the nodes that have the most important function within the system or, in the case of homogenous systems where all components are identical and equally important, to the node that has the most connections.

Synchronous Vs. Asynchronous: This property of failures is closely linked to its reproducibility. If the cause or the timing of the failure is clearly established, it becomes potentially possible, although sometimes arduous, to reproduce and thus to study. When the timing of the failure becomes unpredictable, and the cause impossible to reproduce, the failure can border non-reproducibility, thus making it difficult to study.

Additive Vs. Suppressive: *Suppressive* failure is a clear deficiency of the system, where a component or an interaction between components becomes missing or unavailable. An *additive* disruption is also very common, but cannot strictly speaking be defined as a failure. An example of additive disruption would be the addition of new nodes in a computer or communication networks. The stress caused by the extra information flowing on the system can cause failure. This would be the case at midnight on December 31, when an unusually high number of subscribers try to place cell-phone calls simultaneously. The cellular network becomes saturated and cannot cope with the increase in demand.

Structural Vs. Functional: In this case, we compare whether it is the integrity of the components themselves that suffers a failure or if the structure of the system as a whole is compromised. Failures are called *functional* when one of the components is not functioning normally. In contrast, if an interaction fails, the fault touches the *structure* of the system.

Recoverable Vs. Fatal: Finally, this property describes if the system as a whole is still able to perform this task despite the failure or if the failure is damaging to the system to the point where it collapses and is not able to perform this task

2.5.1 Probabilistic Faults

Throughout this work, we will use *probabilistic* faults to study the behavior of the system and its ability to perform its task. Unrelated faults imply that, for a network with N vertices, the probability $P(N, m)$ that m vertices are faulty at any given time step t is given by:

$$P(N, m) = \binom{N}{m} p_f^m (1 - p_f)^{N-m}$$

where P_f is the probability of failure.

2.5.2 Measuring the Effect of Perturbations

In order to quantify the effect of a perturbation on a binary system, introduce the Hamming distance (HD). The HD between two distinct binary strings is computed as follows: let us assume two configurations of the same length N , the HD between the strings is the number of bits that differ from one to the other (see example in Figure 2.7).

Further in this work, we'll introduce more context-specific ways of measuring the effect of a perturbation, especially with regards to biologically inspired systems and related failures (Section 8).

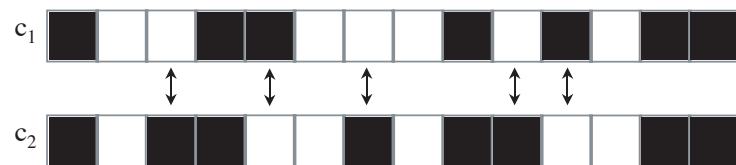


Figure 2.7: Example of Hamming distance between two strings of bits (value are unimportant). In this case the HD is 5.

Part II

Emergence of Real-Life Topologies

Chapter 3

Evolutionary and Genetic Algorithms

Nothing in biology makes sense
except in the light of evolution.

Theodosius Dobzhansky

Evolutionary Algorithms (EAs) are family of stochastic search methods inspired by Darwin’s evolution theory [Hol75, MFH91]. Such algorithms operate on a *population* of candidate solutions to a considered problem, applying the principle of “survival of the fittest” to produce new solutions approaching an optimum. In EAs, solutions are said to “evolve”. Evolution is divided in *generations* that correspond to a population of candidate solutions. The initial population, at the first generation, consists of man-made or arbitrary solutions. EAs rely on objective functions that evaluate each candidate solution according to how well it solves the problem and respects the constraints. A new generation is created by selecting individuals in the population according to their adequacy towards the objectives (fitness level). We apply stochastic operators (i.e. mutation, recombination) on the selected candidate solutions to generate offspring solutions to populate the new generation. There exist several mutation and crossover schemes, depending on the encoding, and several selection schemes to choose from. In Section 3.1, we detail the specific subcategory of EAs we have used for this project, namely genetic algorithms. This process leads to the evolution of a population of individuals that are better suited to their environment, i.e. the problem, than the individuals they were created from at the previous generation. Usually, the size of the population remains constant, with better-performing offspring replacing their *parents*. Multi-objective EA methods rely on the non-domination criterion, also called Pareto-domination criterion [Coe99], which compares different solutions according to all objective functions simultaneously. At any time, the evolving population contains a set of solutions that represents a variety of trade-offs between the different objectives. Evolution stops when the objective functions are adequately satisfied, see Figure 3.1. When the amount of time and computational power are limited, EAs are known to offer suboptimal, yet excellent solutions. The success of finding suitable solutions using EA methods relies on two main factors: the encoding of the candidate solution in the population and the definition of the objective functions. These factors are the main challenges faced by the researcher. Other decisive components of EA performance in time and solution quality include the size of the population, the genetic operators and the probabilities with which they are used, and the quality of the initial population. Although very demanding in computer resources, EAs are naturally suited for parallel or distributed implementation [AT02].

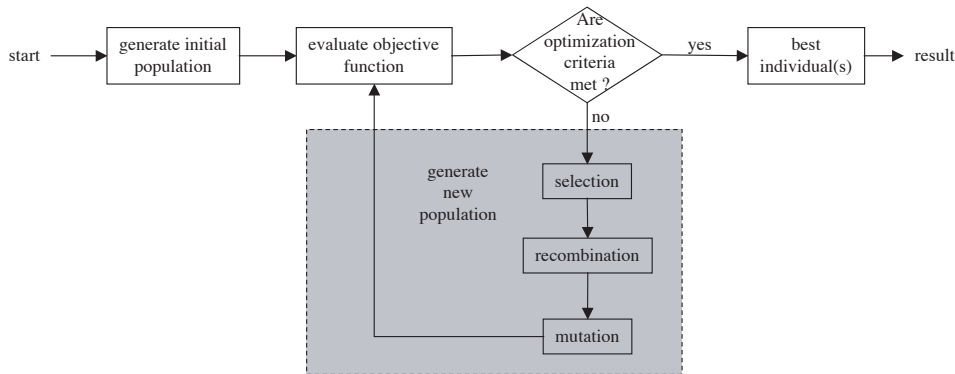


Figure 3.1: Structure of a single population evolutionary algorithm

3.1 Genetic Algorithms

Genetic Algorithms (GAs) are EAs where the candidate solutions are customarily encoded in string of binary values. This allows the use of very simple genetic operators, such as one or two-point crossover, and random mutations. Definitions below remain valid for all EAs although specific examples are presented in a binary form that is GA-compatible.

3.1.1 Encoding

Encoding both the problem and the individuals (i.e. the possible solutions) is a two-step process. Firstly, the *natural problem* itself must be translated into a mathematical model. Now, we have to convert this mathematical model into an EA-compatible form. Potential solution become *chromosomes*, also called *genotypes*, using the chosen encoding scheme. This is the mould out of which all our *individuals*, or *phenotypes*, will be created. At all times, an individual can be translated into a solution to the problem (i.e. to evaluate its fitness, see 3.1.3), and vice-versa. Therefore, since the phenotype is problem-specific, there is no general encoding scheme that would suit every problem. In addition to that, the phenotypes have to be compatible with all the chosen genetic operators (c.f. 3.1.4), that is to say you must be able to perform crossover and mutation using the individual's genotype. GAs use a common and flexible schemes consisting of a chain of bits (0's and 1's).

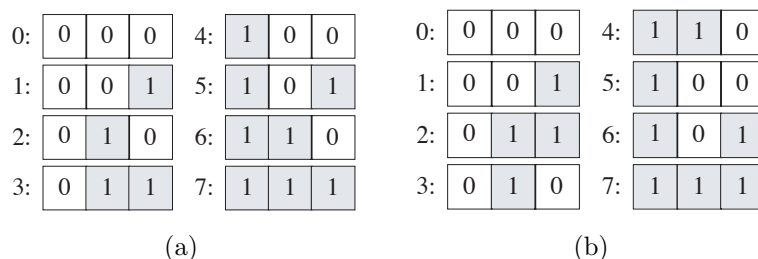


Figure 3.2: Possible encoding: binary (a) and Gray's (b).

In the examples of Figure 3.2, the individual's encodings which are used are both binary, but Gray's encoding (b) has the further advantage of keeping the Hamming distance (defined in Section 2.5.2) between two successive representations of the individuals down to 1, therefore they remain close to each other in the space of solutions by consecutive numbers, close in terms of the Hamming distance.

3.1.2 Initial Population

In order to explore as much as possible of the space of possible solutions, the initial population should be evenly scattered throughout the entire space. In the example of Figure 3.3, the initial population generated in (a) has a much higher probability of finding the optimal solution than the one generated in (b).

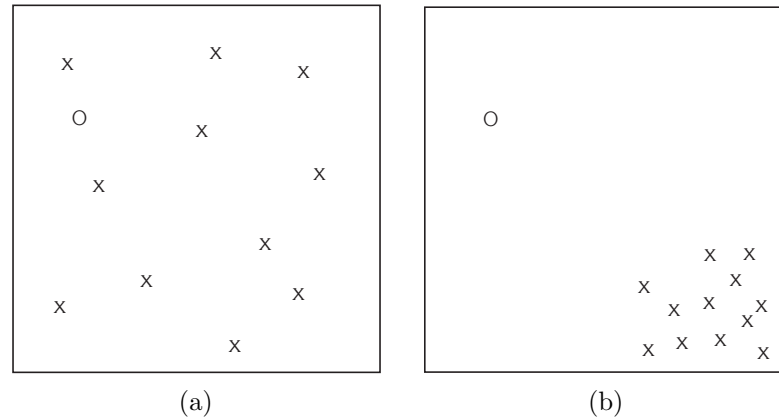


Figure 3.3: Example of a space of the possible solutions, the optimum is represented by a circle, while individuals of the initial population by Xs.

On the other hand, if the optimum is known to be in a precise zone of the search space, the initial individuals ought to be grouped in that area. This will favor *exploitation* instead of *exploration* (c.f. 3.1.4) in contrast with the left case (a).

3.1.3 Phenotype Evaluation

This stage, also called *fitness assignment* [Gol89], consists of submitting an individual, or more precisely its representation (see 3.1.1), to the problem and evaluating its performance at solving it. This will permit a classification and comparison of the individuals, useful at the selecting stage of the EA to produce the next generation (see 3.1.4).

3.1.4 Genetic Operators

Countless different selection, mutation, recombination and replacement methods have been proposed, matching more or less closely the ways of nature. Although many are very interesting and accurate, here, we will describe the most common ones.

Selection

Selection determines which individuals are chosen for recombination. First, the fitness is assigned (in 3.1.3), then the selection is performed. Parents are usually selected according to a function of their fitness:

- *roulette-wheel* selection, also called *stochastic sampling with replacement* [Bak87], is a simple and early selection scheme. This algorithm maps each individual on a roulette-wheel, with share-size proportional to the individual's fitness. Then, the roulette is "spun", offering to each individual a fair chance to be selected for recombination. The process is repeated until the desired number of individuals is obtained (called *mating population*).
- *ranking selection* is very similar to the roulette-wheel selection, only this time share sizes on the roulette-wheel are not proportional to the individual's fitness but to its rank with respect to the other individuals' fitness values in the population.

- *tournament selection* [BT95, GD91]: a number of individuals are chosen randomly from the population and the best individual from this group is selected as a parent. This process is repeated as often as necessary to fill the mating population. The size of the tournament selection, i.e. the number of individuals involved, ranges from 2 (for *binary-tournament*) to the number of individuals in the population.

Recombination

Recombination (or *crossover*) produces new individuals in combining the information contained in the parents' genotype. Depending on the representation of the individual's encoding, there exist both binary valued and real valued recombinations. Because of the rarity of the second case, we will omit it. Several binary valued recombination methods exist, amongst which we will mention:

- *single-point* crossover, where only one position is selected uniformly at random and the genotypes exchanged between the individuals from this point, two new offspring are produced.
- *multi-point* crossover, where several crossover positions are chosen at random with generally no duplicates and sorted in ascending order. Then the genotypes between successive crossover points are exchanged between the two parents. The resulting individuals produced are the two new offspring. The first section is usually not exchanged between individuals. parents to produce two new offspring. The idea behind multi-point crossover, and many of the variations on the crossover operator, is that parts of the chromosome representation that contribute the most to the performance of a particular individual may not necessarily be contained in adjacent substrings. Furthermore, the disruptive nature of multi-point crossover appears to encourage the exploration of the search space, rather than favoring the convergence to highly fit individuals early in the search, thus making the search more robust ([DJS92]).
- *uniform* crossover [Sys89]. The scheme explained above is generalized to make every position of the genotype a potential crossover point. A crossover mask, the same length as the individual structure, is created at random and the parity of the bits in the mask indicates which parent will supply the offspring with which bits. However in reality, uniform crossover results in almost random offspring, similarly to mutation (see Section 3.1.4) but more disruptive.

Uniform crossover, like multi-point crossover, has been claimed to reduce the bias associated with the length of the binary representation used and the particular coding for a given parameter set. This helps to overcome the bias in single-point crossover towards short substrings without requiring precise understanding of the significance of the individual bits in the individuals' representation.

Mutation

In nature, in addition to recombination, a phenomenon called *mutation* is observed. In a few words, it can be described as random alteration of living organisms' genotype, where errors in the process of DNA replication occur, leading to "mutants". These variations are infinitesimal and occur very frequently. Nevertheless, an effect on the phenotype are comparatively rare, due to the important quantity of non-coding sequences in DNA. Nevertheless, in nature, they can have devastating effects on living organisms; the mutated gene may cause its owner to become fitter than ever or, on the contrary, to not be viable, or even not have any visible effect at all. This phenomenon has a tendency to boost evolution. Therefore, it is commonly used in GAs too. It has the further advantage of recovering from the loss of genotypes occurring during selection. As in the case of recombination, there are two distinct mutation schemes used in EAs: the real-valued mutation and the binary mutation. Again, as the real-valued representation is more complex, and not used in this work, we will discuss only the binary mutation. As mentioned before, mutation is ruled by the mutation probability or *mutation rate*. Two different approaches exist; this mutation rate can either stay constant or change (i.e. increase or decrease) during the generations of the EA. There is no general rule. In nature this depends mainly on environmental factors such as pollution or radiation. In our case it depends on the chosen implementation, and both should be considered

in order to better explore the solution space. For binary valued individuals mutation means the flipping of variable values, because every variable has only two states, from 0 to 1 or vice-versa. For every individual the variable value to change is randomly chosen. Of course, the mutation of a position will depend on the previously defined mutation probability.

Replacement

Once the offspring have been produced by selection, recombination and mutation of individuals from the previous population, the fitness of the offspring is determined. Depending on the means used, fewer offspring than the size of the original population might be produced; then, to maintain the size of the original population constant, parents might have to be reinserted into the population. Similarly, if not all offspring are needed in each generation or if more are generated than the size of the old population then a reinsertion scheme must be used to determine which individuals are to exist in the new population. This, of course, is true only in fixed-size populations. The selection method determines the reinsertion scheme: local reinsertion for structured EAs (c.f. 3.1.5) and global reinsertion for all other selection methods (below).

Different schemes of global reinsertion exist:

- produce as many offspring as parents and replace all parents by the offspring (pure reinsertion);
- produce less offspring than parents and replace parents uniformly at random (uniform reinsertion);
- produce less offspring than parents and replace the worst parents (elitist reinsertion);
- produce more offspring than needed for reinsertion and reinsert only the best offspring (fitness-based reinsertion). Parents with fitness that equals their offspring are generally replaced anyway.

Pure reinsertion is the simplest scheme. Every individual lives for one generation only. This scheme is, for instance, used in standard GAs. However, it is very likely that very good individuals are replaced without producing better offspring and thus, good information is lost. To avoid this situation, the elitist combined with fitness-based reinsertion is recommended: with each generation, a given number of the least-fit parents are replaced by the same number of the most-fit offspring. Therefore, the best parent will live alongside the best offspring, lasting over generations and combining their information in order to create better individuals (e.g. better solutions). Despite our efforts, parents may be replaced by offspring with a lower fitness, thus the average fitness of the population can decrease. However, if the inserted offspring are extremely bad, they will be replaced with new offspring in the next generations.

Exploration vs. Exploitation

All of the above schemes, crossover, mutation and selection, depend strongly on a reduced number of parameters. Finding the right value and the right balance between all these is a matter of experimentation. Certainly, some basic rules and common sense apply to avoid steering evolution in an unwanted direction. EAs have to find the right balance between exploration and exploitation. On the one hand, we tend to want to thoroughly explore the space of solution by, for instance, increasing the mutation rate. On the other hand, we would like to better exploit the good individuals we already have, by combining them more and not risk them leaving a good area of the solutions' space through too much mutation. The first method may miss the good solutions by moving around too much and modifying good solutions more than necessary, whereas the second one may remain cornered into exploring restricted areas in search for a local optimum, losing sight of the absolute optimum. Hence, a trade-off must be found between these two extremes.

3.1.5 Structured Evolutionary Algorithm

The spatial dimension of EAs must be considered as it emerges in nature and bears interesting properties. In nature, behaviors are usually locally restricted since selection, recombination and replacement can usually only take place in a relatively confined neighborhood. As a result, crossing between individuals from distant continents may be scarce. This can also be another constraint EAs borrowed from nature and can definitely be seen as one of the trade-off between exploration and exploitation [Tom05]. Consequently, we will try to take into account the spatial location of our individuals with respect to each other. Again, there are several widely used representations that involve placing the whole population on some sort of topology, thus restricting their possible contact between individuals. As a consequence, the individuals' interactions are confined into *neighborhoods* and global communication can no longer take place. The most studied structure for EAs is the 2-dimensional grid, where each individual is placed on a node of a toroidal structure (see 2.2.2). Possible relationships are restricted to individuals directly connected by an edge. Of course, the number of neighbors is variable, for instance 4, 8 or 12, but always much smaller than the population size (see Figure 3.4).

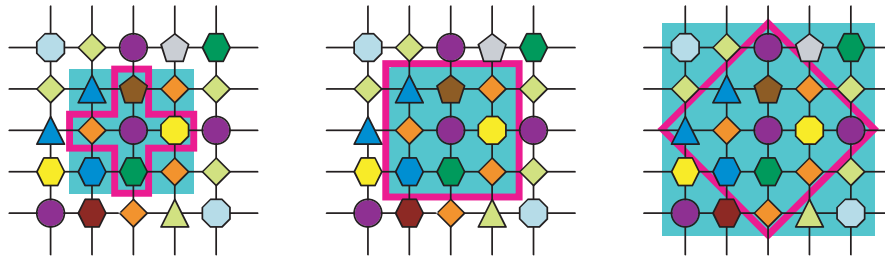


Figure 3.4: Examples of different neighborhood structures with sizes 4 (also called von Neumann), 8 (also called Moore) and 12.

Because of the restrictions described above, some adaptations of the previously defined genetic operators are necessary. In particular concerning the selection and replacement schemes. In structured EAs, discussed in [GS91, VSKB92], every individual resides inside a constrained environment called the local neighborhood in contrast to the other selection methods where the whole population or subpopulation is the selection pool. In this case, individuals interact only with individuals inside their region. The neighborhood is defined by the structure in which the population is distributed and can be seen as the group of potential mating partners. In order to generate a new generation we consider each node of the structure simultaneously. In its neighborhood, including the considered individual itself, two parents are selected according to a chosen scheme (see section 3.1.4). These two mating partners are used to produce a single offspring that is possibly mutated. This new potential solution, once evaluated, will replace the considered individual if it happens to have a better fitness (see Figure 3.5).

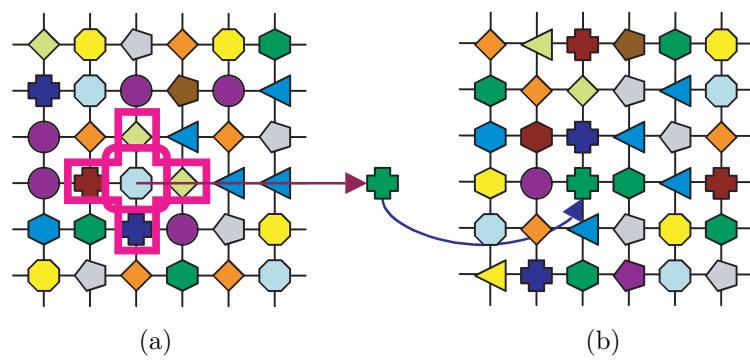


Figure 3.5: For each individual in the population an offspring is produced (a) combining parents selected in the individual's neighborhood and the new offspring replaces it in the next generation (b).

Chapter 4

Cellular Automata for Computation

Lately I've been working to convince myself that everything is a computation.

Rudy Rucker

Cellular automata (CAs) provide simple discrete deterministic mathematical models for physical, biological and computational systems. Despite their simple construction, cellular automata have been proven to be capable of complicated behavior, and to generate complex patterns with universal features [MOW84, Wol94]. They are therefore a typical example of emergence in complex systems.

CAs are constituted of elementary parts called *cells* arranged on mono-dimensional or bi-dimensional regular lattices. Each cell carries a value from a finite set of states. As a deterministic entity, the cell's state will change with discrete time-steps according a predefined set of rules (e.g. look-up table, or a mathematical function) involving the state of the cell itself and of its nearest (i.e. connected) neighbors. In the present work we will only handle the case of mono-dimensional (linear) CA's with binary values but, of course, any topology can be used to lay a CA on as well as any alphabet different from the binary 0's and 1's.

CAs were introduced by von Neumann [vN66] as abstract models with embedded biological properties, such as self-reproduction. Any system with many identical discrete elements undergoing deterministic local interactions may be modeled as a cellular automaton. More complex CAs are obtained using non-linear local evolution. Physical, biological and mathematical examples of "useful" CA models englobe the aggregation phenomena in snowflakes' growth or the organization of simple entities by repeated application of a simple set of rules. The best-known example of a two-dimensional CA is the game of "life" [Gar70, Gar71], which happens to be proven as "computationally universal", capable of evaluating any Turing computable function. According to Church's thesis in the formal theory of computation, such cellular automata may thus potentially simulate any possible computational system.

In our case, Boolean automata for which the cellular state $s \in \{0, 1\}$ are used. Its current state and that of its neighborhood's determine the state of a cell at the next time-step. The regular cellular array (grid) is d -dimensional, where $d = 1, 2, 3$ is used in practice. For one-dimensional grids, a cell is connected to r local neighbors (cells) on either side where r is referred to as the *radius* (thus, each cell has $2r + 1$ neighbors, including itself), see Figure 4.1. The term *configuration* refers to an assignment of ones and zeros to all the cells at a given time step. It can be described by $s(t) = (s_0(t), s_1(t), \dots, s_{N-1}(t))$, where N is the lattice size. Often, CAs have periodic boundary conditions $s_{N+i}(t) = s_i(t)$. Here we will consider an extension of the concept of CA in which, while the rule is the same on each node, nodes can be connected in any way, that is, the topological

structures are general graphs, provided the graph is connected and self and multiple links are disallowed.

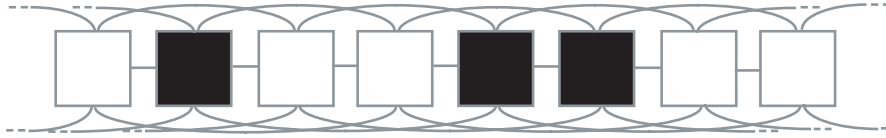


Figure 4.1: Linear CA, with a radius $r = 3$ and a random (initial) configuration. Black represents state 1 and white is 0.

CAs are, as mentioned before, capable of universal computation. Nevertheless, they are usually studied on prototypical tasks, such as the game of life, the density classification task (described in the article section A.3.1, and the synchronization task (in article section A.3.2. The bi-dimensional representation of the tasks in those section shows the temporal evolution of the linear CA. Each line shows the state of the entire CA at successive discrete time-steps. Our models performance on the density classification and the synchronization tasks is studied in details in the article A.

Chapter 5

Articles in this Part

In Einstein's general relativity the structure of space can change but not its topology. Topology is the property of something that doesn't change when you bend it or stretch it as long as you don't break anything.

Edward Witten

In Articles A and B we study complex network systems in the context of cellular automata for distributed computation as described in Section 4 on real-world-like network topologies.

In Article A

In this first work, we evolve the underlying topology of CAs starting from regular and random structures (see Section 2.2.2) using structured genetic algorithms (as described in 3.1). This work is mainly inspired by two sources: Watts' work [Wat99] where he uses hand-made small-world network as the structure for CAs and Packard's work [Pac88] and later Mitchell's work [MCH94] where she uses GAs to evolve the update rules of CAs. In both these works, the resulting CAs perform as well as or better than "classical" CAs with hand-written majority rule on lattices. Using very little bias, these newly evolved topologies fall in the small-world range and share many properties with the real-world structures described in the background sections. Moreover, they show outstanding performance while extraordinarily robust in the face of probabilistic failure. Interestingly, we show that whether starting from a random graph or a regular lattice, the evolution in both cases give rise to topologies that meet halfway between order and randomness. This article repeats somewhat the formal definition about graphs, then describes the two different CA prototypical tasks. It describes the GA in details, specific to the task and the starting point structure. It also evaluates the performance of each cases and its ability to perform the other task.

In Article B

In this second publication, we propose to compare the performance of our evolved small-world-like topologies as support for CA computation with that of several different hand-made real-world networks: scale-free structures, introduced in section 2.2.2. Scale-free structures are common in communication networks with an unplanned organic growth, such as the Internet, and in the context of information flow, they show superior resilience to non-targeted failures when compared to regular or random structures. On the other hand, scale-free networks are vulnerable to targeted attacks on hubs. Strictly speaking, targeted attacks are essentially equivalent to random failure

on homogenous systems where both the nodes type and their connection degree are comparable or identical. We study the effect of *transient* and *permanent* failures (see 2.4) in CAs of different structure. Interestingly, our evolved structures show comparable and even superior robustness with respect to scale-free networks. Moreover, there is a clear difference in terms of fault tolerance when comparing scale-free networks constructed with different methods. This hints that, although both Barabási-Albert and configuration model scale-free networks show a long-tailed power-law degree distribution, it is not a sufficient property to assume that their performance and robustness in distributed computation will be comparable.

Article A

Evolution and Dynamics of Small-World Cellular Automata

Authors:	M. Tomassini, M. Giacobini, Ch. Darabos
Published in:	Complex Systems
Volume:	15
Number:	4
Pages:	261–284
Year:	2005

Abstract

We study an extension of cellular automata to arbitrary interconnection topologies for the majority and the synchronization problems. By using an evolutionary algorithm, we show that small-world type network topologies consistently evolve from regular and random structures without being designed beforehand. These topologies have better performance than regular lattice structures and are easier to evolve, which could explain in part their ubiquity. Moreover, we show experimentally that general graph topologies are much more robust in the face of random faults than lattice structures for these problems.

Keyword

small-world networks, cellular automata, evolutionary computation, distributed computation

A.1 Introduction

Networks, which can be formally described by the tools of graph theory, are a central model for the description of many phenomena of scientific, social and technological interest. Typical examples include the Internet, the World Wide Web, social acquaintances, electric power networks, neural networks, and many others. In recent years there has been substantial research activity in the science of networks, motivated by a number of new results, both theoretical and applied. The pioneering studies of Watts and Strogatz [WS98, Wat99] have been instrumental in initiating the movement, and they have been followed by many others in the subsequent years. Their key observation was that most real networks, both in the biological world as well as man-made structures, have mathematical properties that set them apart from regular lattices and from random graphs, which were the two main topologies that had been studied until then. In particular, they introduced the concept of *small-world networks*, in which most pairs of vertices seem to be connected by a short path through the network. The existence of short paths between any pair of nodes has been found in networks as diverse as the Internet, airline routes, neural networks, metabolic networks, among others. The presence of short paths is also a characteristic of random

graphs, but what sets these real networks apart is a larger *clustering coefficient* than that of random graphs having a comparable number of nodes and links. The clustering coefficient roughly represents the probability that two nodes that are neighbors of a third one, are also neighbors of each other, which means that there is more local structure in these networks than in plain random graphs. In section A.2 we offer a brief quantitative introduction to relevant graph concepts that are used in this work. An excellent recent review of the field is to be found in [New03].

The topological structure of a network has a marked influence on the processes that may take place on it. Regular and random networks have been thoroughly studied from this point of view in many disciplines. In computer science, for instance, variously connected networks of processors have been used in parallel and distributed computing [Lei92], while lattices and random networks of simple automata have also received a great deal of attention [Gar95, Kau93]. On the other hand, due to their novelty, there are very few studies of the computational properties of networks of the small-world type. One notable exception is Watts' book [Wat99] in which cellular automata (CAs) computation on small-world networks is examined in detail. However, there is no hint in these works as to how such networks could arise in the first place, without being designed by a prescribed algorithm. Many man-made networks have grown, and are still growing, incrementally and explanations have been proposed for their actual shape. The Internet is a case in point, for which a preferential attachment growth rule has given good results [AB02]. This rule simply prescribes that the likelihood for a new node of connecting to an existing one depends on the node's degree: high-degree nodes are more likely to attract other nodes. However, there exist many networks in nature for which some kind of Darwinian variation and selection in a population of networks has surely taken place. This is the case, for instance, in the emergence of biological neural networks.

Thus, how these automata networks might have come to be selected is an interesting yet unanswered question. In this work, we let a simple artificial evolutionary process find "good" network structures according to a predefined fitness measure, without prescribing the fine details of the wiring. We take as prototypical problems the *majority classification* problem and the *synchronization* tasks, which are the same that Watts discusses in [Wat99] as a useful first step. This will also allow us to compare the products of artificial evolution with Watts' results. We will also investigate the effect of some structural constraints on the evolutionary process. Another aspect of interest is how evolved networks compare with known lattice-CA solutions in terms of robustness in the presence of noise. This point will be explored in some detail.

In the next section some background material on graphs is briefly discussed. Section A.3 describes the CA problems and previous results. Section A.4 presents our evolutionary search for efficient networks. In section A.6 the fault-tolerance properties of the evolved networks are studied. Section A.7 gives our conclusions and ideas for future work.

A.2 Useful definitions for graphs

For ease of reference, here we collect a few definitions and some nomenclature for graphs that are used throughout this work. The treatment is necessarily brief: a more detailed account can be found for example in [New03, Wat99].

Let V be a nonempty set called the set of *vertices* or *nodes*, and let E be a symmetric binary relation on V , i.e. a set of unordered pairs of vertices. $G = (E, V)$ is called an *undirected graph* and E is the set of *edges* or *links* of G . In *directed graphs* edges have a direction, i.e. they go from one vertex to another and the pairs of vertices are ordered pairs. Here we only deal with undirected graphs.

When vertices (u, v) of an undirected graph form an edge they are said to be *adjacent* or *neighbors*. The *degree* k of a vertex is the number of edges impinging on it (or, equivalently, the number of neighbors). The *average degree* $\langle k \rangle$ is the average of all the vertex degrees in G .

A *path* from vertex u to vertex v in a graph G is a sequence of edges that are traversed when going from u to v with no edge traversed more than once. The *length* of a path is the number of edges in it. The *shortest path* between two vertices u and v is the path with the smallest length joining u to v .

A graph is *connected* if there is a path between any two vertices. A *completely connected* undirected graph G with $|V| = N$ vertices has an edge between any two vertices. The total number of edges is $N(N - 1)/2$.

A *random graph* is a graph in which pairs of nodes are connected with a given probability p . Consequently, the total number of edges in a random graph is a random variable whose expectation value is $p[N(N - 1)/2]$. Several useful results on random graphs are described in [AB02].

Four statistics are particularly useful for small-world and random graphs: the average degree described above, the *clustering coefficient*, the *characteristic path length*, and the *degree distribution*. They are briefly described below and in more detail in [AB02].

Let us take a particular node j in a graph, and let us assume that it has k edges connecting it to its k neighboring nodes. If all k vertices in the neighborhood were completely connected then the number of edges would be equal to $k(k - 1)/2$. The clustering coefficient C is defined as the ratio between the E edges that actually exist between the k neighbors and the number of possible edges between these nodes:

$$C = \frac{2E}{k(k - 1)}$$

The clustering coefficient of a random graph is simply $\langle k \rangle / N = p$, where N is the total number of vertices. For a regular lattice, C is given by the following formula:

$$\frac{3(k - 2)}{4(k - 1)},$$

where k is the (constant) number of nodes that are connected to a given node. C is thus independent of N for a regular lattice, and approaches $3/4$ as k increases.

The characteristic path length L is defined in [Wat99] as the median of the means of the shortest path lengths connecting each vertex $v \in G$ to all other vertices.

The degree distribution $P(k)$ of a graph G is a function that gives the probability that a randomly selected vertex has k edges incident on it. For a random graph $P(k)$ is a binomial peaked at $P(\langle k \rangle)$. But most real networks do not show this kind of behavior. In particular, in scale-free graphs which seem to be common in real-life [AB02], $P(k)$ follows a power-law distribution: $P(k) = ck^{-\gamma}$, with c and γ positive constants.

According to Watts and Strogatz [Wat99, WS98], a small-world graph can be constructed starting from a regular ring of nodes in which each node has k neighbors ($k \ll N$) by simply systematically going through successive nodes and “rewiring” a link with a certain probability p . When the edge is deleted, it is replaced with an edge to a randomly chosen node. This procedure will create a number of shortcuts that join distant parts of the lattice. Shortcuts are defined to be edges that join vertices that would be more than two edges apart if they were not connected directly. These shortcuts are the hallmark of small worlds and, while L scales logarithmically in the number of nodes for a random graph, in small-world graphs it scales approximately linearly for low rewiring probability and tends to the random graph limit as the probability increases. This is due to the progressive appearance of shortcut edges between distant parts of the graph, which obviously contract the path lengths between many vertices. However, small worlds typically have a higher clustering coefficient than random graphs. Small-world networks have a degree distribution $P(k)$ that is close to binomial for intermediate and large values of the rewiring probability p , while $P(k)$ tends to a delta function for $p \rightarrow 0$.

Following Watts [Wat99], we will show our results as a function of the parameter ϕ , which is the fraction of edges in a graph that are shortcuts. The range of ϕ is $[0, 1]$, where a value of 0 (no shortcuts) corresponds to a perfect regular lattice, and 1 corresponds to the random graph limit (every link is a shortcut on the average). In between lies the small-world range, with the typical small-world behavior already present for low ϕ values (around 0.01-0.1). For higher ϕ values, the graphs tend to be more random-like.

Small-world graphs as defined by Watts and Strogatz are not really structurally representative of networks found in the real world, which are often, but not always, of the scale-free type, if anything [ASBS00]. However, they are easy to construct and measure and, for the purpose of

the artificial CA problems with which we are concerned, they are a perfectly legitimate choice for studying the influence of the network structure on the dynamics.

A.3 The cellular automata problems

CAs are dynamical systems in which space and time are discrete. A standard CA consists of an array of cells, each of which can be in one of a finite number of possible states. Here we will only consider boolean automata for which the cellular state $s \in \{0, 1\}$. The regular cellular array (lattice) is d -dimensional, where $d = 1, 2, 3$ is used in practice. In one-dimensional lattices, the topology used here, a cell is connected to r local neighbors (cells) on either side, where r is referred to as the *radius* (thus, each cell has $2r + 1$ neighbors, including itself).

CAs are updated synchronously in discrete time steps, according to a local, identical rule. The state of a cell at the next time step is determined by the current states of a surrounding neighborhood of cells, including the cell itself:

$$s_i^{t+1} = f(s_{i-r}^t, \dots, s_i^t, \dots, s_{i+r}^t), \quad f : k^{2r+1} \rightarrow k$$

where s_i^t denotes the value of site i at time t , $f(\cdot)$ represents the local transition rule, and r is the CA radius. The term *configuration* refers to an assignment of ones and zeros to all the cells at a given time step. It can be described by $\mathbf{s}^t = (s_0^t, s_1^t, \dots, s_{N-1}^t)$, where N is the lattice size. Often CAs have periodic boundary conditions $s_{N+i}^t = s_i^t$. Configurations evolve in time according to a global update rule Φ which applies in parallel to all the cells $\mathbf{s}^{t+1} = \Phi(\mathbf{s}^t)$.

Here we will consider an extension of the concept of CA in which the rule is the same on each node but nodes can be connected in any way, that is, the topological structures are general graphs, provided the graph is connected and self and multiple links are disallowed.

A.3.1 The majority task

The majority (also called density) task is a prototypical distributed computational task for CAs. For a finite CA of size N it is defined as follows. Let ρ^0 be the fraction of 1s in the initial configuration (IC) \mathbf{s}^0 . The task is to determine whether ρ^0 is greater than or less than $1/2$. If $\rho^0 > 1/2$ then the CA must relax to a fixed-point configuration of all 1s; otherwise it must relax to a fixed-point configuration of all 0s, after a number of time steps of the order of the lattice size N (N is odd to avoid the case $\rho^0 = 0.5$). This computation is trivial for a computer having a central control. Indeed, just scanning the array and adding up the number of, say, 1 bits will provide the answer in $O(N)$ time. However, it is nontrivial for a small radius one-dimensional CA since such a CA can only transfer information at finite speed relying on local information exclusively, while density is a global property of the configuration of states [MHC93]. Figure A.1 shows the operation of one of the best CAs obtained through artificial evolution.

It has been shown that the density task cannot be solved perfectly by a uniform, two-state CA with finite radius [LB95], although a slightly modified version of the task can be shown to admit perfect solution by such an automaton [CST96]. The *performance* P of a given rule on the majority task is defined as the fraction of correct classifications over 10^4 randomly chosen ICs. The ICs are sampled according to a binomial distribution (i.e., each bit is independently drawn with probability $1/2$ of being 0). Clearly, this distribution is strongly peaked around $\rho^0 = 1/2$ and thus it makes a difficult case for the CA to solve.

The lack of a perfect solution does not prevent one from searching for imperfect solutions of as good a quality as possible. In general, given a desired global behavior for a CA (e.g., the density task capability), it is extremely difficult to infer the local CA rule that will give rise to the emergence of a desired computation due to possible nonlinearities and large-scale collective effects that cannot in general be predicted from the sole local CA updating rule. Since exhaustive evaluation of all possible rules is out of the question except for elementary ($d = 1, r = 1$) automata, one possible solution of the problem consists in using evolutionary algorithms, as first proposed by Mitchell *et al.* [MCH94, MHC93] for uniform CAs, and by Sipper for nonuniform ones [Sip97].

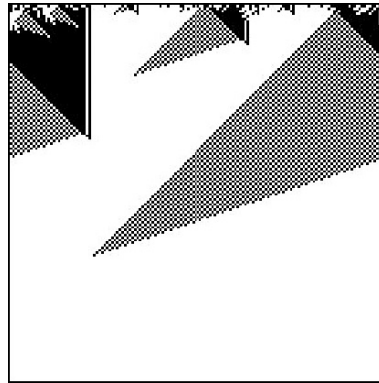


Figure A.1: The operation of an evolved one-dimensional, radius three CA for the density task. The CA cell states are represented horizontally (black stands for 1 and white is 0). Time increases down the page. The CA rule has been obtained through artificial evolution by Mitchell *et al* [MCH94]. The density ρ^0 is 0.416 and the lattice size N is 149.

Watts [Wat99] studied a general graph version of the density task. Since a CA rule table depends on the number of neighbors, given that a small-world graph may have vertices with different degrees, he considered the simpler problem of fixing the rule and evaluating the performance of small-world graphs on the task. The chosen rule was a variation of the *majority* rule (not to be confused with the majority problem). The rule simply says that, at each time step, each node will assume the state of the majority of its neighbors in the graph. If the number of neighbors having state 0 is equal to the number of those at 1, then the next state is assigned at random with equal probability. When used in a one-dimensional CA this rule has performance $P \simeq 0$ since it gives rise to stripes of 0s and 1s that cannot mix at the borders. Watts, however, has shown that the performance can be good on other network structures, where “long” links somewhat compensate for the lack of information transmission of the regular lattice case, in spite of the fact that the node degrees are still low. Indeed, Watts built many networks with performance values $P > 0.8$, while the best evolved lattices with the same average number of neighbors had P around 0.77 [MCH94, MHC93] and were difficult to obtain.

In a remarkable paper [SR97], Sipper and Ruppin had already examined the influence of different connectivity patterns on the density task. They studied the co-evolution of network architectures and CA rules, resulting in non-uniform, high-performance networks, while we are dealing with uniform CAs here. Since those were pre-small world years, it is difficult to state what kind of graphs were obtained. However, it was correctly recognized that reducing the average cellular distance, i.e. the characteristic path length, has a positive effect on the performance.

A.3.2 The synchronization task

The one-dimensional synchronization task was introduced in [DCMH95]. In this task the CA, given an arbitrary initial configuration, must reach a final configuration, within $M \simeq 2N$ time steps, that oscillates between all 0s and all 1s on successive time steps. Figure A.2 depicts the space-time diagram of a CA that solves the task.

As with the density task, synchronization also comprises a non-trivial computation for a small-radius CA, and it is thus extremely difficult to come up with CA rules that, when applied synchronously to the whole lattice produce a stable attractor of oscillating all 0s and all 1s configurations. Das *et al.* were able to automatically evolve very good ring CAs rules of radius three for the task by using genetic algorithms [DCMH95]. Sipper did the same for quasi-homogeneous CAs, i.e. CAs with a few different rules instead of just one [Sip97], attaining excellent performance for radius one CAs. The performance of a CA on this task is evaluated by running it on randomly generated initial configurations, uniformly distributed over densities in the range $[0, 1]$, with the CA being run for $M \simeq 2N$ time steps. Figure A.2 is an illustration of the space-time operation

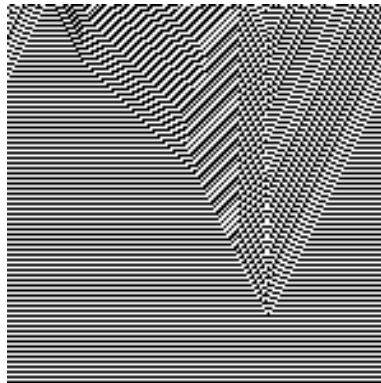


Figure A.2: The operation of an evolved one-dimensional CA for synchronization (source: [Sip97]). The ring size N is 149.

of a typical evolved ring CA that solves the synchronization task. Evolved CAs for this task have performance P close to 1.

Watts [Wat99] also has a brief section on the synchronization task on small worlds. Watts finds that a simple variant of the majority rule used above for the density task, works also for the synchronization task. The rule is called the “contrarian” rule, and it operates in the same way as the majority rule, except that it gives the opposite state as output. We adhered to this rule in order to be able to compare our results with Watts’. The synchronization task is probably less interesting than the density in a small world because, while the ordinary lattice CA have less than optimal performance on the density task, they are near-perfect for synchronization. Nevertheless, the task is a difficult one as it requires precise coordination among many elementary agents, and it is thus representative of distributed cooperative problem solving and worth studying.

A.4 Artificial evolution of small worlds

Evolutionary algorithms have been successfully used for more than ten years to evolve network topologies for artificial neural networks and several techniques are available [TT01]. As far as the network topology is concerned, the present problem is similar, and we use an unsophisticated structured EA with the aim of evolving small-world networks for the density and synchronization tasks. Our EA is spatially structured, as this permits a steady diffusion of good solutions in the population due to a less intense selection pressure [GATT04]. The population is arranged on a 20×20 square grid for a total of 400 individuals. Each individual represents a network topology and it is coded as an array of integers denoting vertices, each one of which has a list of the vertices it is connected to, as the graph is undirected. The information is redundant (e.g. if X is connected to Y, then both have the other in their own connections list). The automaton rule is the generalized majority rule described above for all cases. During the evolution the network nodes are constrained to have a maximum degree of 50. The termination condition is reached after computing exactly 100 generations.

A.4.1 Evolution of graphs for the density task

The fitness of a network of automata in the population is calculated by randomly choosing 100 out of the 2^N possible initial configurations (ICs) with uniform density—i.e. any initial density has the same probability of being selected— and then iterating the automaton on each IC for $M = 2N$ time steps, where $N = 149$ is the automaton size. The network’s fitness is the fraction of ICs for which the rule produced the correct fixed point, given the known IC density. At each generation a different set of ICs is generated for each individual. Selection is done locally using a central

individual and its north, east, south and west first neighbors in the grid. Binary tournament selection is used with this pool. The winner is then mutated (see below) and evaluated. It replaces the central individual if it has a better fitness.

Mutation is designed to operate on the network topology and works as follows. Each node of an individual is mutated with probability 0.5. If chosen, a vertex (called target vertex) will have an edge either added or removed to a randomly chosen vertex (called destination vertex) with probability 0.5. This will only happen if all the requirements are met (minimum and maximum degree are respected). If the source vertex has already reached its maximum degree and should be added one edge or its minimum degree and should be removed one edge, the mutation will not happen. If the same case happens with the target, another one is randomly chosen. This version of the algorithm does not use recombination operators.

Evolution from regular lattices

In this first series of experiments we started from regular rings, which is the customary way for constructing small-world graphs [Wat99]. In order not to unduly bias the evolution, the initial population was composed by individuals that are regular rings with node degree $k = 4$, i.e. each vertex is connected to its four nearest neighbors in the ring, instead of rings with $k = 6$, which is the case treated by Watts. Moreover, we slightly modify each of them by adding an edge with a probability of 0.1 applied to each vertex.

Figure A.3 (a) shows the population entropy, ϕ (see section A.2), fitness, and performance of the best individual (as defined in sections A.3.1 and A.4) as a function of the generation number. The curves represent data from a typical run out of 50 independent runs of the EA.

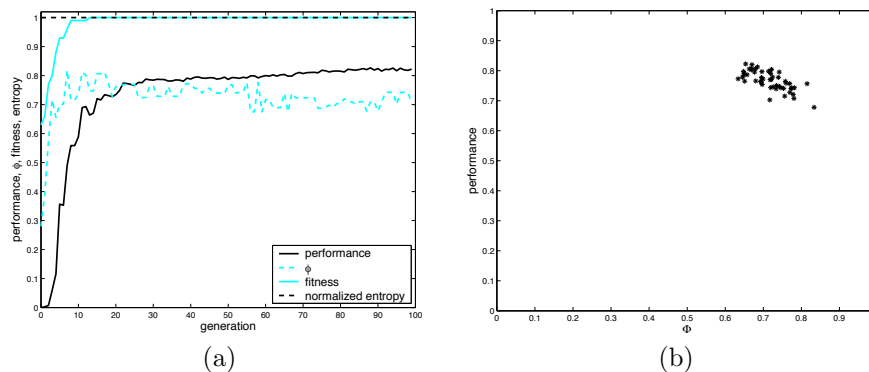


Figure A.3: A typical evolutionary run starting from a perturbed ring population (a). The ϕ - performance values of the 50 best individuals found in the 50 evolutionary runs (b).

We see that fitness quickly reaches high levels, while performance, which is a harder measure of the generalization capabilities of the evolved networks on the density task, stays lower and then stabilizes at a level greater than 0.8. The population entropy remains high during all runs, meaning that there is little diversity loss during evolution. Note that the entropy refers to the “genotype” and not to fitness. This is unusual and probably due to the spatial structure of the evolutionary algorithm, which only allows slow diffusion of good individuals through the grid [GATT04]. The ϕ curve is particularly interesting as it permits a direct comparison with Watts’ hand-constructed graphs [Wat99]. The results fully confirm his measurements, with networks having best performance clustering around ϕ values between 0.6 and 0.8. This is clearly seen in figure A.3 (b) where the 50 best networks found are reported as a function of their ϕ , which is to be compared with figure 7.2, p.190, in [Wat99]. The mean degree $\langle k \rangle$ of the evolved networks is around 7, which compares well with the ring case and Watts’s (see table A.7). Therefore, we see that even a simple EA is capable of consistently evolving good performance networks in the small-world range. This is not the case for the standard ring CAs for the majority task, where good rules are notoriously difficult to evolve. In fact, while we consistently obtain networks having

performance around 0.8 in each evolutionary run, Mitchell *et al.* [MCH94] found that only a small fraction of the runs lead to high-performance CAs. As well, our networks and Watts' reach higher performance: 0.82 against 0.77 for the lattice. Evidently, the original fitness landscape corresponding to the 2^{128} possible ring CAs with radius three is much more difficult to search than the landscape corresponding to all possible graphs with N vertices. To this we may add that the performance of the small-world solutions are better than those of the original lattices as N increases, as was observed by Watts and confirmed by our study (not shown here for lack of space). Work is under way to study the basic statistics of the above landscapes in order to obtain a better understanding of their structures.

The operation of a typical evolved small-world network can be seen in the space-time diagram of figure A.4. Although a direct comparison with the previous figure A.1 is difficult due to the very different network connections, still, one can see that the information transfer is much faster thanks to the distant connections, and the problem is thus solved in fewer steps.



Figure A.4: The operation of an evolved small-world CA for the density task. The density ρ^0 is 0.470 in (a) and 0.523 in (b).

Evolution from random graphs

Although the results of artificial evolution from rings are appreciable, giving rise to networks of automata with small-world topology and good performance, the way the initial population is generated might nevertheless contain a bias towards such graphs. In order to really assess the power of this artificial evolution, we designed a second series of experiments in which all the parameters are the same except that the initial population is now formed by arbitrary random graphs. A random graph having N vertices can be constructed by taking all possible pair of vertices and connecting each pair with probability p , or not connecting it with probability $1 - p$. In the experiments $p = 0.03$ and there is no constraint on the minimum node degree, which means that disconnected graphs are also possible. However, we discarded such graphs and ensure that all the networks in the initial population are connected with average degree $\langle k \rangle = Np$ of 4.47.

Figure A.5 (a) depicts the same curves starting from random graphs as figure A.3 (a) does for the perturbed ring initial population. Here too, 50 independent runs have been performed and a typical one is plotted in the figure.

We see again that genotypic diversity is maintained through evolution as the entropy is always high. Likewise, fitness rises quickly and stays near the maximum. Performance has a different behavior initially. While it starts low and rapidly and steadily increases in the previous case, here it has an approximate value of 0.4 at the beginning. The difference is due to the fact that, in the perturbed ring case, the initial population is still mainly constituted by regular rings, which we know are incapable of performing the density task using the majority rule as CA rule. In the random graph case, a fraction of the networks in the initial population does a better job on the task. The same conclusion can be reached by looking at the ϕ curve. While in the perturbed ring

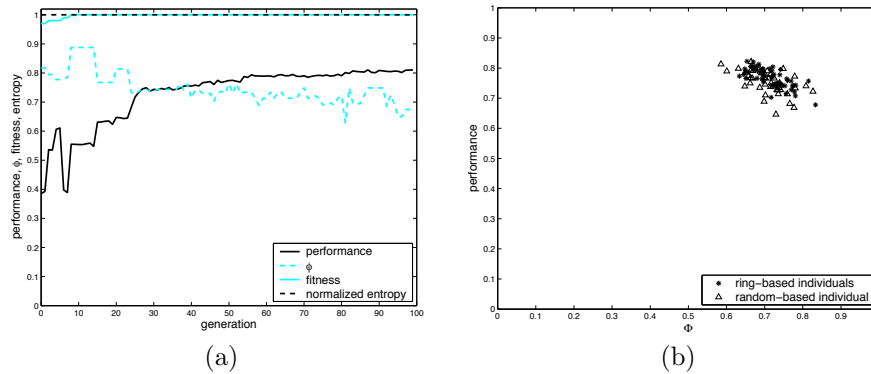


Figure A.5: A typical evolutionary run starting from a random graph population (a). The ϕ -performance values of the 50 best individuals found in the 50 evolutionary runs (b).

case ϕ starts low (ϕ is 0 for a lattice) and then slowly increases toward values around 0.7, in the random graph case the contrary happens: ϕ is rather high at the beginning because truly random graphs predominate during the first part of the evolution, i.e. about 20 generations. After that, graphs are more of the small-world type and converge toward the same ϕ region in both cases. This can be clearly seen in figure A.5 (b), where the best 50 individuals of all runs for both initial rings and random graphs are plotted together. It should be noted that in both figures A.3 and A.5 performance does not stop improving even though fitness has reached its maximum value. This is an indication of the good learning and generalization capabilities of the evolved networks.

The following figure A.6 shows the degree distribution of the best networks found by evolution in the ring case (a), and the random graph case (b). Although the number of vertices is too small for a rigorous statistical treatment, it is easily seen that the distribution is close to binomial in both cases, which is what was expected.

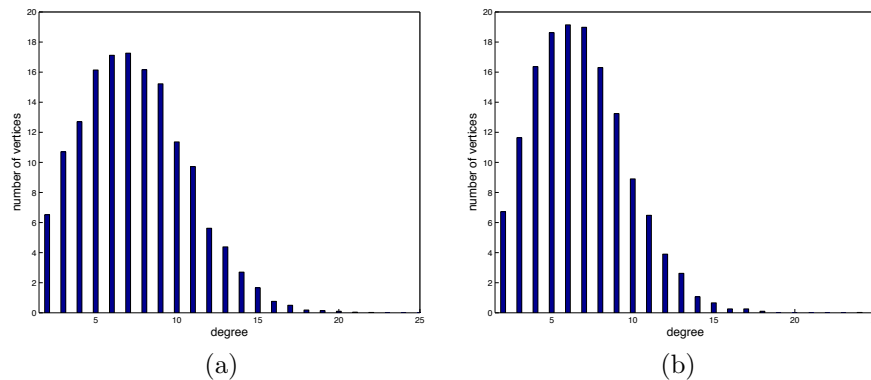


Figure A.6: Degree distribution of best evolved networks. Initial ring population (a); initial random graph population (b).

Finally, figure A.7 summarizes the graph-theoretical statistical properties of the five best evolved individuals for the ring case (left), and for the random graph case (right). It is interesting that, although no provision was explicitly made for it, the average number of neighbors $\langle k \rangle$ ended up being around seven, very close to six used by construction in Watts [Wat99] (remember that his construction for small-world graphs leaves the initial $\langle k \rangle$ for a ring unchanged). Measured average path lengths L and clustering coefficients C have expected values, given the corresponding ϕ values which, without being in the random graph regime, are nevertheless not far from it for both for initial rings and initial random graphs. In other words, the networks with good performance

Net. Type	$\langle k \rangle$	C	L	Φ	P
ring	7.906	0.053	2.649	0.654	0.823
ring	7.611	0.053	2.703	0.670	0.820
ring	7.409	0.048	2.750	0.685	0.813
ring	7.342	0.049	2.736	0.669	0.807
ring	7.450	0.057	2.730	0.679	0.807
rand.	7.798	–	2.695	0.664	0.821
rand.	7.543	–	2.736	0.585	0.812
rand.	7.355	–	2.729	0.686	0.800
rand.	7.422	0.062	2.736	0.631	0.798
rand.	6.778	–	2.858	0.748	0.797

Figure A.7: The ten best evolved networks. $\langle k \rangle$ is the mean node degree. C is the clustering coefficient. L is the characteristic path length. ϕ is the percentage of shortcuts, and P is the network performance on the density task. Left part: ring-based evolved individuals. Right part: random-based evolved individuals (a – in random-based graphs means that the clustering coefficient is not computable since those graphs are allowed to have vertices with a degree smaller than 2).

constructed by Watts as well as those artificially evolved have many links rewired. It is worth noticing that, although the evolutionary algorithm does not limit the node degree other than establishing a maximum allowed value $k_m = 50$, all the evolved networks have a much smaller $\langle k \rangle$. The operation of graph-CAs evolved from random conditions is qualitatively indistinguishable from those originated from rings (see figure A.4).

A.4.2 Evolution of automata graphs for the synchronization task

To artificially evolve automata networks for the synchronization task, we have used exactly the same genetic algorithm setting as for the density task (see section A.4), except for the fitness function, which is the same as the one used by Das *et al.* [DCMH95]. We have again two starting points for the initial population: either a population of slightly perturbed radius-two rings, or arbitrary connected random graphs with the same number of vertices.

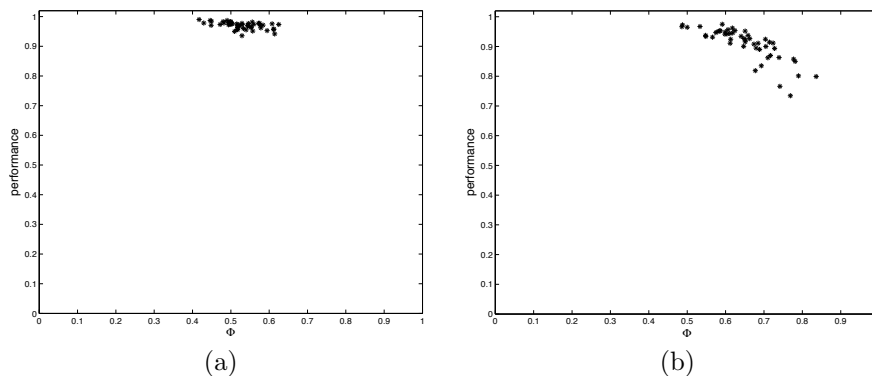


Figure A.8: The ϕ vs performance values of the 50 best individuals found in the 50 evolutionary runs on the synchronization task. (a) starting from rings, (b) starting from random graphs.

The results are perfectly in line with those obtained by Watts [Wat99]. Both the ϕ range and the performance are wholly comparable, as can be seen comparing his figure 7.10 (p.197) and figure A.8 in this section. For reasons of space, we omit the curves representing the evolution of ϕ , fitness, and performance through generations, which show behaviors very similar to those seen in the case of the density task (section A.4.1).

A.5 Limiting the number of shortcuts

The network evolutions described in the previous section lead to small-world graphs with a comparatively high proportion of shortcuts, in agreement with the automata built by Watts. Since our systems are just a paradigm for coordinated distributed task solving by simple automata, we do not take into account real-world constraints such as wire length and other engineering considerations that would be essential for the actual construction of the network. Nevertheless, it would be interesting to study the evolution of the same graphs with the added requirement that ϕ is as low as possible.

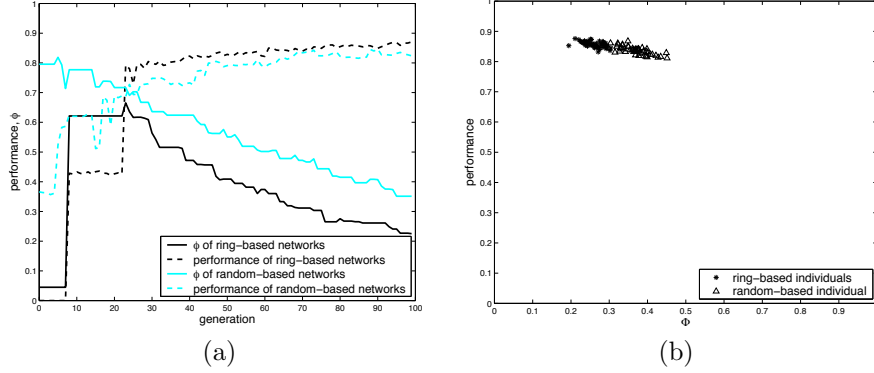


Figure A.9: Density task results. (a): ϕ and performance curves vs. generation number for an initial population of perturbed rings and a population of random graphs. (b): ϕ vs performance values of the 50 best individuals found in the 50 evolutionary runs starting from rings and starting from random graphs. Fitness function is f' .

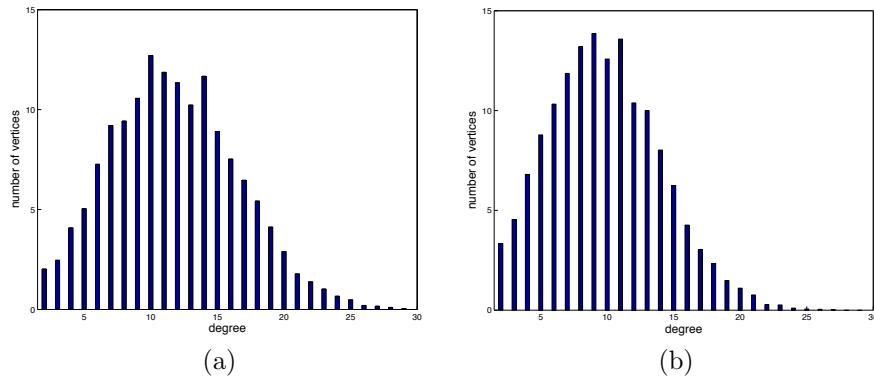


Figure A.10: Degree distribution of best evolved networks for the density task, using f' as a fitness function. Initial ring population (a); initial random graph population (b). Mean degrees $\langle k \rangle$ are 11.76 in (a) and 9.72 in (b).

An easy way to implement this criterion is to include a term in the fitness function which, for a given network fitness, favors networks having a lower ϕ value. A similar approach was used by Sipper and Ruppin [SR97] with the aim of minimizing the wire length of their inhomogeneous CAs. Obviously, the most general way to solve the problem would be to use multi-objective optimization. However, the simpler technique will prove sufficient for our exploration. The new fitness function is thus:

$$f' = f + (1 - \phi) \times w,$$

where f is the usual CA fitness, w is an empirical weight factor with $w \in [0, 1]$, and f' is the effective fitness. After experimenting with a few different w values, we finally used $w = 0.6$ in the experiments described here, although the precise w value only makes a small difference.

As depicted in figure A.9 for the density task, we see that the introduction of a selection pressure favoring networks with smaller ϕ values is effective in evolving graph-CAs that keep high performance, equal to or better than those previously found using unconstrained evolution (see figures A.3 and A.5 in section A.4.1 for comparison). Here also, starting from a population of perturbed rings or random graphs does not make a big difference although, as expected, starting from slightly perturbed rings, which have low ϕ , tends to favor slightly lower ϕ values of the evolved networks. The ϕ values are around 0.3 (see figure A.9), while they are about 0.7 in the previous case (figures A.3 and A.5). The average degrees are somewhat higher however: 11.76 and 9.72 for ring-based and random graph-based respectively. This compares favorably with Watts' hand-constructed networks (figure 7.4, p. 192 in [Wat99]), where one can see high-performance networks with ϕ around 0.3 but with average degree $\langle k \rangle$ equal to 12. It is clear thus that, to some extent, having more neighbors on the average compensates for the reduced number of shortcut links. The degree distribution for evolved networks is shown in figure A.10 and confirms that the degree distribution $P(k)$ is approximately Poissonian.

Experiments of the same type on the synchronization task (not shown here for reasons of space), give similar results, in the sense that high-performance graph-CAs are obtained easily by artificial evolution. The average values of ϕ starting from perturbed rings and random graphs are 0.19 and 0.34 respectively. The degree distribution is again approximately binomial and the mean degrees $\langle k \rangle$ are 13.06 for ring-based individuals, and 10.04 for random-based ones.

A.5.1 Task flexibility of the evolved networks

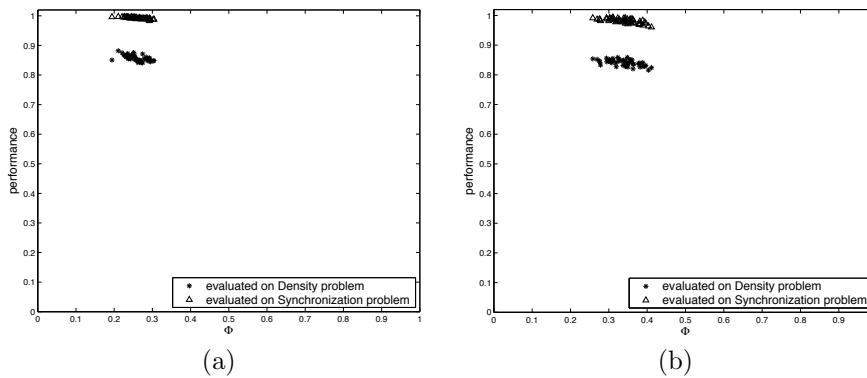


Figure A.11: Performance vs ϕ of networks evolved for the density task on both density and synchronization. Ring-based networks (a), random graph-based networks (b).

As we have seen, it is much easier to evolve small-world networks rather than regular lattices for both tasks. This is also manifest in the fact that networks evolved specifically for one task yield good performance when used for the other one. As noted by Watts [Wat99], the two tasks are nearly identical and thus this finding is not surprising. Furthermore, this remains true for the whole range of ϕ values for which automata have been evolved or generated by hand.

For instance, Figure A.11 shows how networks evolved for the density task using ϕ as a second objective (see previous section) are also well-suited for synchronization (of course, upon changing the rule). The opposite is also true: namely, that networks evolved for the synchronization task can be used for solving the density problem.

A.6 Robustness in the presence of random faults

Noisy environments are the rule in the real world. Since these automata networks are toy examples of distributed computing systems, it is interesting and legitimate to ask questions about their fault-tolerance aspects. A network of automata may fail in various ways when random noise is allowed. For instance, the cells may fail temporarily or they may die altogether; links may be cut, or both things may happen. In this section, we will compare the robustness of standard lattice-CAs and small-world CAs with respect to a specific kind of perturbation, which we call *probabilistic updating*. It is defined as follows: the CA rule may yield the incorrect output bit with probability p_f , and thus the probability of correct functioning will be $(1 - p_f)$. Furthermore, we assume that errors are uncorrelated. This implies that, for a network with N vertices, the probability $P(N, m)$ that m cells (vertices) are faulty at any given time step t is given by

$$P(N, m) = \binom{N}{m} p_f^m (1 - p_f)^{N-m}$$

i.e. it is binomially distributed. It should be noted that we do not try to correct or compensate for the errors, which is important in engineered system but very complicated and outside our scope. Instead, we focus on the “natural” fault-tolerance and self-recovering capabilities of the systems under study.

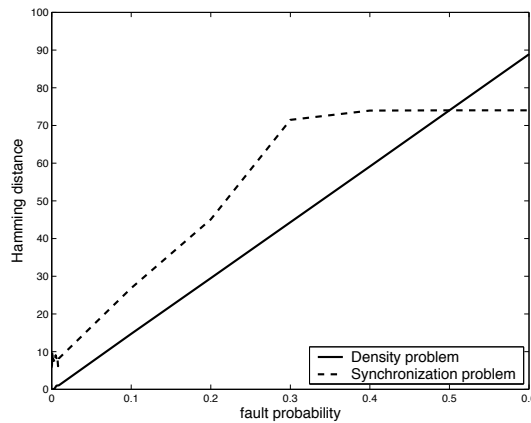


Figure A.12: Hamming distance (y-axis) vs fault probability (x-axis) for the density problem (full line), and the synchronization problem (dashed line). The curves are averages over 10^3 distinct initial configurations.

To observe the effects of probabilistic updating on the CA dynamics, two initially identical copies of the system are maintained. One evolves undisturbed with $p_f = 0$, while the second is submitted to a nonzero probability of fault. We can then measure such things as Hamming distances between unperturbed and faulty configurations, which give information on the spreading of damage (e.g., [STB96] where the case of synchronous, nonuniform CAs is examined). Figure A.12 shows that, for the density task, the amount of disorder is linearly related to the fault probability. This is an excellent result when compared with ring CAs where already at $p_f = 0.001$ the average Hamming distance is about 20 [STB96], and tends to grow exponentially. At $p_f = 0.1$ it saturates at about 95, while it is still only about 20 for the small-world CA.

This striking difference is perhaps more intuitively clear by looking at figures A.13 and A.14. The faulty CA depicted in figure A.14 is the best one obtained by artificial evolution in [MCH94, MHC93] and it is called EvCA here. It is clear that even small amounts of noise are able to perturb the lattice CA so much that either it classifies the configuration incorrectly (c), or it cannot accomplish the task any longer (d) as p_f increases further. For the same amount of noise the behavior of the small-world CA is much more robust and even for $p_f = 0.01$ the fixed point configuration is only slightly altered. Note also that the EvCA configuration has $\rho^0 = 0.416$

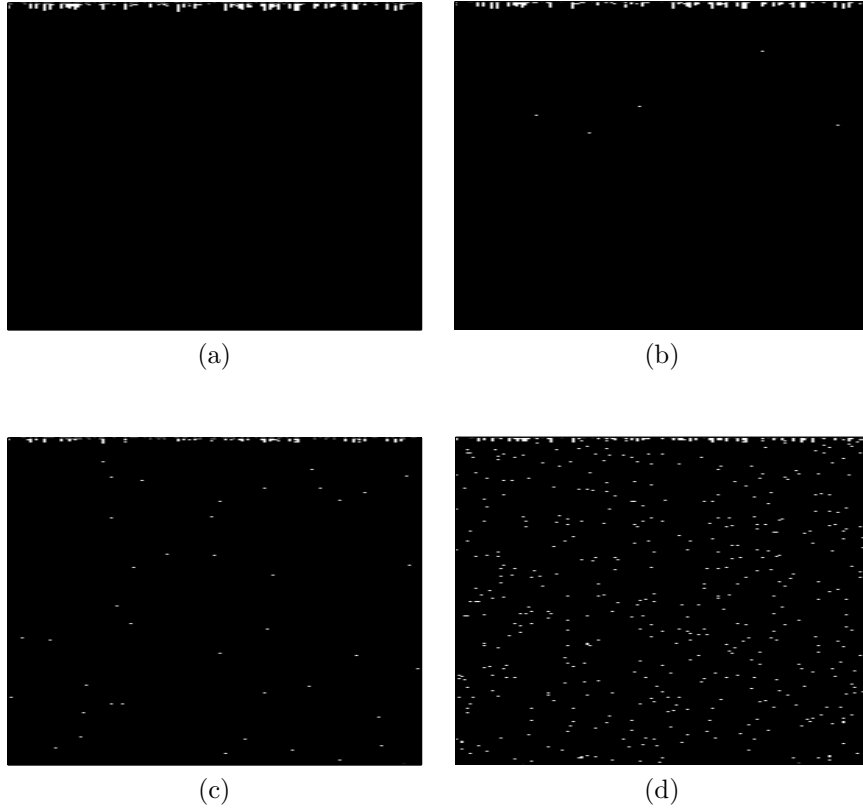


Figure A.13: Typical behavior of a small-world CA under probabilistic updating. The density ρ^0 is 0.490 and the probabilities of fault p_f in (a), (b), (c), and (d) are, respectively, 0, 0.0001, 0.001, and 0.01.

whereas the one used in the small-world CA has $\rho^0 = 0.490$, and it is thus more difficult to classify. For completeness, we note that in a previous study [TV01] we investigated the behavior of evolved *asynchronous* lattice CAs for the density task under probabilistic noise. We found that, while asynchronous CAs are much more fault-tolerant than synchronous ones, their robustness is not as good as that of small-world CAs and their performance is significantly lower.

Looking again at figure A.12 we see that the behavior of the synchronization task (dashed line) under noise is poorer. In fact, it is not possible to maintain strict synchronization in the presence of faults. The system manages to limit the damage for low fault probabilities but it goes completely out of phase over $p_f = 0.2$. For higher probabilities the distance stabilizes around 75 (i.e. half of the cells on the average are in the wrong state). In spite of this, the behavior is still much better than the one observed for ring CAs, where at $p_f = 0.01$ the Hamming distance is already about 55 [STB96], while it is only about 8 in the small-world CA.

A.7 Conclusions

Starting from the work of Watts on small-world cellular automata, we have used an evolutionary algorithm to evolve networks that have similar computational capabilities. Without including any preconceived design issue, the evolutionary algorithm has been consistently able to find high-performance automata networks in the same class of those constructed by Watts. In addition, by giving some evolutionary advantage to low- ϕ networks, the evolutionary process has been able to find networks with a low- ϕ and excellent performance for both tasks.

These results have been easy to find even though the evolutionary algorithm is an unsophis-

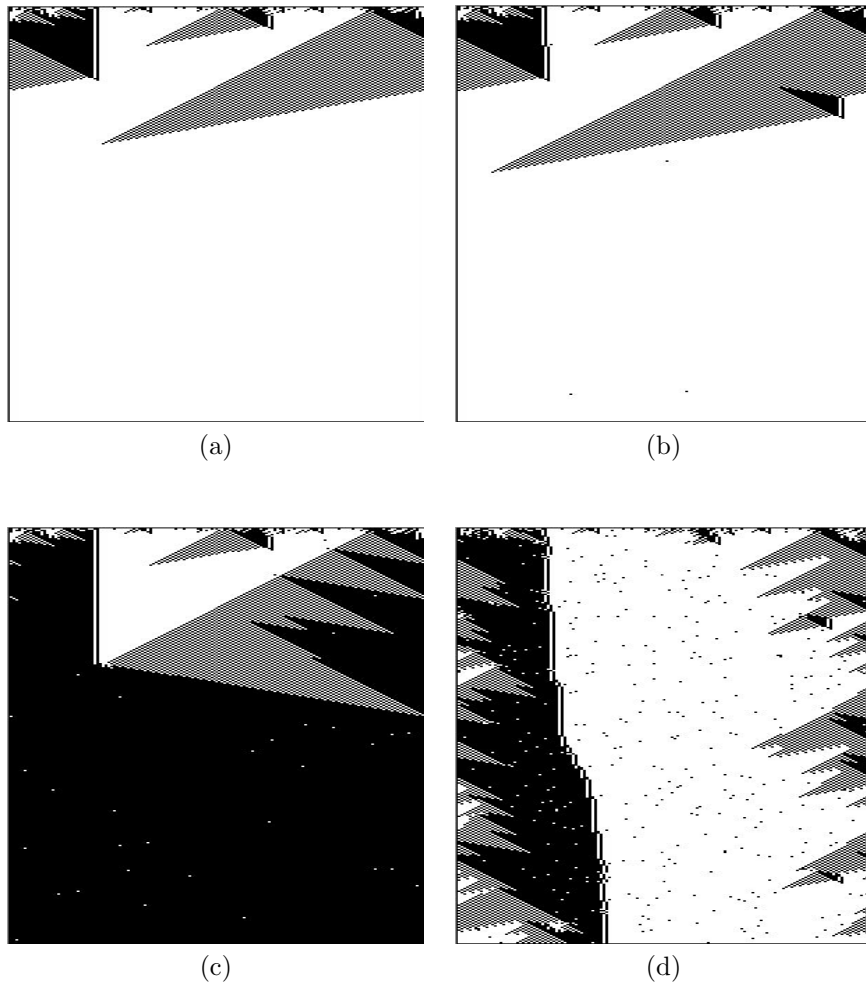


Figure A.14: Typical behavior of EvCA [MHC93] under probabilistic updating. The density ρ^0 is 0.416 and the probabilities of fault p_f in (a), (b), (c), and (d) are, respectively, 0, 0.0001, 0.001, and 0.01.

ticated one. This means that the space of small-world networks is “solutions rich”, which is the contrary of what one observes in the rule space for standard ring CA, where evolving good rules has proved difficult. The power of artificial evolution is seen in the fact that, even starting from a population of completely random graphs, the algorithm finds automata in the same class. This result is an indication that small-world network automata in this range have above average distributed computation capabilities, although we only studied two problems of this type and any generalization would be unwarranted at this stage. Not only are these networks extremely efficient, they also feature above-average robustness against transient probabilistic faults. A comparison with standard lattice CAs shows that small-world CA are much less affected by random noise. The difference is striking, and could be one of the reasons that explain the ubiquity of irregular “natural” collective computational systems with respect to regular structures.

It is also clear at this point that we have not used the power of artificial evolution at its best. In particular, we adopted the fixed rules of Watts and let the networks evolve. It would probably pay if we would let the rule evolve together with the network topology. This has been suggested by Watts [Wat99] and has previously been attempted by Sipper and Ruppin with good results [SR97]. Further work along these lines is needed. We also plan to study the collective computational capabilities of other small-world graph structures, especially scale-free networks.

Article B

Performance and Robustness of Cellular Automata Computation on Irregular Networks

Authors:	Ch. Darabos, M. Giacobini, M. Tomassini
Published in:	Advances in Complex Systems
Volume:	10
Number:	1
Pages:	85–110
Year:	2007

Abstract

We investigate the performances of collective task-solving capabilities and the robustness of complex networks of automata using the density and synchronization problems as typical cases. We show by computer simulations that evolved Watts–Strogatz small-world networks have superior performance with respect to several kinds of scale-free graphs. Besides, we show that Watts–Strogatz networks are as robust in the face of random perturbations, both transient and permanent, as configuration scale-free networks, while being widely superior to Barabási–Albert networks. This result differs from information diffusion on scale-free networks, where random faults are highly tolerated by similar topologies.

B.1 Introduction

Following the pioneering investigation of Watts and Strogatz [WS98], in recent years there has been substantial research activity in the science of networks, motivated by a number of innovative results, both theoretical and applied. For our purposes in this paper, a network is simply a connected graph $G = (V, E)$, where V is a set of vertices (nodes) and E is a set of undirected edges (links) between nodes in V . Networks are a central model for the description of countless phenomena of scientific, social and technological interest. Typical examples include the Internet, the World Wide Web, social acquaintances, electric power supply networks, biological networks, and many more. The key idea is that most real networks, both in the natural world as well as in man-made structures, have mathematical properties that set them apart from regular lattices and random graphs, which were the two main topologies studied until Watts and Strogatz’s work. Inspired by previous qualitative observations made by social scientists, in their 1998 paper [WS98], Watts and Strogatz introduced an algorithmic construction for *small-world networks* in which pairs of vertices are connected by short paths through the network. The existence of short paths between any pair of nodes has been found since then in real networks as diverse as the Internet, airline

routes, the World Wide Web, neural, genetic, and metabolic networks, citation and collaboration networks, and many others [New03]. The presence of short paths is also a property of standard random graphs as those constructed according to the Erdős-Rényi model [ER59b], but what sets real networks apart from random graphs is a larger *clustering coefficient*, a measure that reflects the locality of a structure.

The topological structure of a network has a marked influence on the dynamical processes that may take place on it. Regular and random networks have been thoroughly studied from this point of view in many disciplines. For instance, the dynamics of lattices and random networks of simple automata have received a great deal of attention [Gar95, Kau93]. On the other hand, there are very few studies of the computational properties of automata networks of the small-world type. Notable exceptions are Watts' book [Wat99], and a few recent articles on automata on scale-free and other complex networks [AC03a, ADGM⁺04, MMT05, GTRP06]. In these works the automata networks were designed by a prescribed algorithm. Recently, we have shown that evolutionary algorithms can be an effective way for obtaining high-performance computational networks in the small-world region without explicit prior design [DGT06, TGD04, TGD05].

In this work we study in detail some computational behaviors of automata networks of the small-world and scale-free types, extending the results presented in [TGD04, TGD05]. As a typical example of collective computational tasks we take the *density task* and the *synchronization task*, to be described below. As natural computational systems often have a degree of stochasticity, it is particularly important to investigate their behavior in a noisy environment. In the present work we shall thus focus on the study of the dynamical properties of those generalized automata networks when faulty behavior can arise. We shall see the behavior in the presence of perturbations strongly depends on the network topology, and we shall draw some conclusions on the suitability of these topologies for collective computation.

In the following section we give a brief account of Watts–Strogatz small-world and scale-free networks. Then an introduction to generalized cellular automata is presented, followed by a description of the computational tasks and their performance measures. Next we discuss the experimental performance of generalized automata networks on the tasks, with an emphasis on their fault tolerant aspects. Finally, we present our conclusions and ideas for future extensions.

B.2 Small-World and Scale-Free Graphs

Although the following material should be well known, we include a succinct description for the sake of completeness so as to make the paper more self-contained. For more details, the reader is referred to the original works.

The Watts–Strogatz Model. Following Watts and Strogatz [WS98], a small-world graph can be constructed starting from a regular ring of N nodes in which each node has k neighbors ($k \ll N$) by simply systematically going through successive nodes and “rewiring” each edge (also called link here) with a certain probability β . When an edge is deleted, it is replaced by an edge to a randomly chosen node. If rewiring a link would lead to a duplicate edge, it is left unchanged. This procedure will create a number of edges, called *shortcuts*, that join distant parts of the lattice. Shortcuts are the hallmark of small worlds. While the average path length, i.e. the average value of all pairs shortest paths between nodes scales logarithmically with the number of nodes in a random graph, in Watts–Strogatz graphs it scales approximately linearly for low rewiring probability but decreases very quickly and tends towards the random graph limit as β increases. This is due to the progressive appearance of shortcut edges between distant parts of the graph, which obviously contract the path lengths between many vertices. However, small world graphs typically have a higher clustering coefficient than random graphs, and a degree distribution $P(k)$ close to Poissonian. The clustering coefficient C of a node is a measure of the probability that two nodes that are its neighbors are also neighbors among themselves. The average $\langle C \rangle$ is the average of the C s of all nodes in the graph.

The Barabási–Albert Model (BA). Albert and Barabási were the first to realize that many real networks grow incrementally and that their evolving topology is determined by the way in which new nodes are added to the network. They proposed an extremely simple model based on these ideas [AB02]. One starts with a small clique, a cluster of fully connected vertices, of size m_0 . At each successive time step a new node is added such that its $m \leq m_0$ edges link it to m nodes already in the graph. When choosing the nodes to which the new nodes connect, it is assumed that the probability π that a new node will be connected to node i depends on the current degree k_i of i . This is called the *preferential attachment* rule. Nodes with already many edges are more likely to be chosen than those that have few. The probability $\pi(k_i)$ of node i to be chosen is given by $\pi(k_i) = k_i / \sum_j k_j$, where the sum is over all nodes already in the graph. The model evolves into a stationary network with power-law probability distribution for the vertex degree $P(k) \sim k^{-\gamma}$, with $\gamma \sim 3$, which justifies the name *scale-free*. As for Watts–Strogatz small-worlds, scale-free graphs have short average path length and clustering coefficients that are higher than those of the corresponding random graphs with comparable number of vertices and edges.

The Configuration Model (CM). Tackling the generation of scale-free graphs top-down, the Configuration Model defines beforehand the power-law $P(k) = ak^{-\gamma} + c$ according to which the degrees will be distributed in the graph [BC78, MSA03]. Clearly, the degree distribution now depends on the γ values set for the experiment, in our case $\gamma \in \{1.5, 2.0, 2.5, 3.0\}$. In this manner, we know before generating the graph how many vertices of each degree we will have. This model has the advantage over the Barabási–Albert model that it will tend to minimize the degree correlations between sets of nodes due to the sequential manner in which the nodes are connected in the algorithm above.

In order to compare results with the performance and fault tolerance of Watts–Strogatz small-worlds and BA scale-free graphs, these CM scale-free networks must have comparable average degrees of $\langle k \rangle \in \{6, 12\}$. This is achieved by attributing a number of *stubs* e (the end of an edge) to each vertex following the power-law degree distribution described above. A small number of stubs will be left unattributed, compared to the $\langle k \rangle \times N$ to be attributed. Then, following [MSA03], we randomly attribute the unallocated stubs to the remaining vertices, thus achieving the desired average degree. This allocation process may lead to duplicated edges, therefore it is sometimes necessary to reallocate some stubs, thus derogating significantly with the degree distribution. Figure B.1 (a) shows a typical degree distribution of scale-free graphs obtained with this algorithm. Clearly, the intended power-law distribution is partially lost in favor of the fixed average degree $\langle k \rangle$.

The Modified Configuration Model (MCM). In order for the degree distribution to remain as close as possible to the predefined power-law, we introduce here a slight modification to the previous algorithm. The first step of distributing stubs is identical, but instead of distributing the remaining ones at random, we try to increase only by one at a time the number of nodes with a degree k , making sure it remains below the number of nodes with a degree $k - 1$ according to the given power-law distribution. Figure B.1 (b) shows typical degree distribution of scale-free graphs obtained with this Modified Configuration Model algorithm. Clearly, the distribution is now closely following a power-law.

B.3 Cellular and Networked Automata

Cellular Automata (CAs) are dynamical systems in which space and time are discrete. A standard CA consists of a finite or infinite d -dimensional regular lattice of cells, each of which can take on a value from a finite, typically small, set of values. The value of each cell at time step t is a function of the values of a small local neighborhood of cells at time $t - 1$. The cells update their states simultaneously according to a given local rule.¹

¹Asynchronous CAs can also be considered, though they will not be treated in this paper.

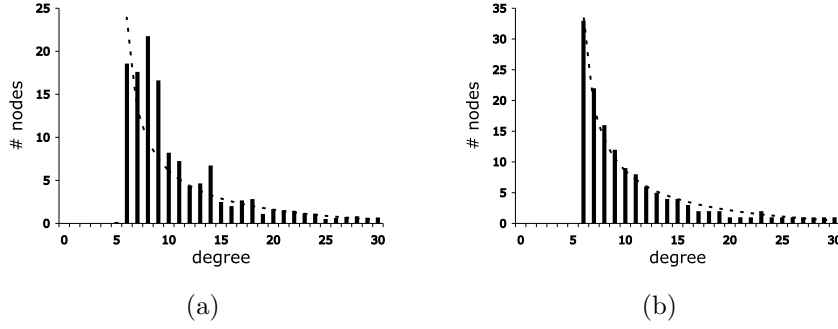


Figure B.1: Degree distribution of scale-free graphs with $\gamma = 2.5$, averaged out over 50 networks. (a) graphs built using the original Configuration Model algorithm $\langle k \rangle = 12$ and (b) graphs generated with the Modified Configuration Model algorithm $\langle k \rangle = 12$. The dashed line represents the power-law curve to be fit as closely as possible.

Formally, a cellular automaton A is a quadruple

$$A = (\Sigma, G, d, f),$$

where Σ is a finite set of states, G is the cellular neighborhood, $d \in \mathbb{Z}^+$ is the dimension of A , and f is the local cellular interaction rule, also referred to as the transition function.

Given the position of a cell, \mathbf{i} , $\mathbf{i} \in \mathbb{Z}^d$, in a regular d -dimensional uniform lattice, or grid (i.e., \mathbf{i} is an vector of integers in a d -dimensional space), its *neighborhood* G is defined by:

$$G_{\mathbf{i}} = \{\mathbf{i}, \mathbf{i} + \mathbf{r}_1, \mathbf{i} + \mathbf{r}_2, \dots, \mathbf{i} + \mathbf{r}_{n-1}\},$$

where n is a fixed parameter that determines the neighborhood size, and \mathbf{r}_j is a fixed vector in the d -dimensional space.

The *local transition rule* $f : \Sigma^n \rightarrow \Sigma$ maps the state $s_{\mathbf{i}} \in \Sigma$ of a given cell \mathbf{i} into another state from the set Σ , as a function of the states of the cells in the neighborhood $G_{\mathbf{i}}$. In uniform CAs f is identical for all cells, whereas in non-uniform ones f may differ between different cells, i.e., f depends on \mathbf{i} , $f_{\mathbf{i}}$.

For a finite-size CA of size N (such as those treated herein) a *configuration* of the grid at time t is defined as

$$C(t) = (s_{\mathbf{0}}(t), s_{\mathbf{1}}(t), \dots, s_{\mathbf{N}-1}(t)),$$

where $s_{\mathbf{i}}(t) \in S$ is the state of cell \mathbf{i} at time t . The progression of the CA in time is then given by the iteration of the *global mapping*, also called *evolution operator* F

$$F : C(t) \rightarrow C(t+1), \quad t = 0, 1, \dots$$

through the simultaneous application in each cell of the local transition rule f . The global dynamics of the CA can be described as a directed graph, referred to as the CA's *phase space*.

In this paper we focus on one-dimensional binary CAs, i.e. $\Sigma = \{0, 1\}$. In this case f is a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and the neighborhood size n is usually taken to be $n = 2r + 1$ such that:

$$s_{\mathbf{i}}(t+1) = f(s_{\mathbf{i}-r}(t), \dots, s_{\mathbf{i}}(t), \dots, s_{\mathbf{i}+r}(t)),$$

where $r \in \mathbb{Z}^+$ is a parameter, known as the *radius*, representing the standard one-dimensional cellular neighborhood. The domain of f is the set of all 2^n n-tuples. For finite-size grids, spatially periodic boundary conditions are frequently assumed, resulting in a circular grid; formally, this implies that cellular indices are computed modulus N .

To visualize the behavior of a one-dimensional CA one can use a two-dimensional space-time diagram, where the horizontal axis depicts the configuration $C(t)$ at a certain time t and the

vertical axis depicts successive time steps, with time increasing down the page (for example, see Figure B.2).

We now extend the previous concepts and introduce the term of *Generalized Automata Networks* (GANs). With respect to standard CAs, the most important change concerns the network topology: whereas in CAs this topology is a d -dimensional regular lattice, GANs can be built on any connected graph. Let $G = (V, E)$ be a graph, where V is a set of vertices and E is a set of edges. E is a binary relation on V ; it is either *symmetric* if edges are unordered pairs, as in undirected graphs, or *asymmetric* if edges are ordered pairs, as in directed graphs. Both cases arise in GANs. With these definitions, a GAN on V is a triple $(G, \Sigma, \{f_i | i \in V\})$. The only change with respect to lattice synchronous CAs is in the local transition function f which now depends on the *degree* k_i of vertex i , i.e. the number of neighbors can be different for different $i \in V$. This can be formalized as: $f_i : \Sigma^{k_i} \rightarrow \Sigma$. As in the case of CAs, non-uniform GANs can be defined by allowing f_i to depend not only on the degree k_i of vertex i , but also on the position of i in the graph G . Likewise, asynchronous GANs can be defined by explicitly stating a sequence of vertex updates, including random sequences. In this paper we deal exclusively with two-state, uniform, synchronous GANs.

B.4 Generalized Automata Networks Computations

The design, evolution, and performance evaluation of one-dimensional CAs that approximately perform the density and the synchronization tasks has a long history; an excellent review appears in [CMD03].

The density Task. The density task is a prototypical distributed computational problem for binary CAs. For a finite one-dimensional CA of size N it is defined as follows. Let $C(0)$ be the *initial configuration* of the CA, i.e. the sequence of bits that represents the state of each automaton at time 0, and let ρ_0 be the fraction of 1s in the initial configuration. The task is to determine whether ρ_0 is greater than or less than $1/2$. If $\rho_0 > 1/2$ then the CA must relax, after a number of time steps of the order of the grid size N , usually $2N$, to a fixed-point configuration of all 1's, otherwise it must relax to a fixed-point configuration of all 0's. Here N is set to 149, the value that has been customarily used in research on the density task (if N is odd one avoids the case where $\rho_0 = 0.5$ for which the problem is undefined); for a pictorial example, see Figure B.2 (a).

This computation is trivial for a computer with a central control: just scanning the cell array and adding up the number of, say, 1 bits will provide the answer in $O(N)$ time. However, it is nontrivial for a small radius one-dimensional CA since such a CA can only transfer information at finite speed relying on local information exclusively, while density is a global property of the configuration of states. It has been shown that the density task cannot be solved perfectly by a uniform, two-state CA with finite radius [LB95], although a slightly modified version of the task can be shown to admit perfect solution by such an automaton [CST96], or by a combination of automata [Fuk97].

The *performance* of a CA rule on the density task is defined as the fraction of correct classifications over $n = 10^4$ randomly chosen initial configurations (ICs). These are sampled according to a binomial distribution among the 2^N possible binary strings i.e., each bit is independently drawn with probability $1/2$ of being 1. Clearly, this distribution is strongly peaked around $\rho_0 = 1/2$ and thus making a difficult case for the CA to classify. The best CAs found to date either by evolutionary computation or by hand have performances around 0.8 [CMD03].

CAs performing the density task classify each given initial configuration (IC) in one of the following categories: correctly classified, incorrectly classified or not converged, if the CA has not relaxed to a fixed point.

Using his small-world construction, and thus relaxing the regular lattice constraint, Watts [Wat99] has been able to easily obtain GANs with performance around 0.85, with the same mean connectivity $\langle k \rangle$ as in the regular CA case. He used a majority rule for the automata: at each time step, each node will assume the state of the majority of its neighbors in the graph. In case of a tie, when the degree is even, the next state is assigned at random with equal probability. When

used in a one-dimensional CA this rule has performance $P \simeq 0$ since it gives rise to stripes of 0s and 1s that cannot mix at the borders.

In [TGD04] we showed that such high-performance small-world networks can be obtained automatically and easily with a simple evolutionary algorithm, starting from either regular or completely random graphs (see Section B.4.1).

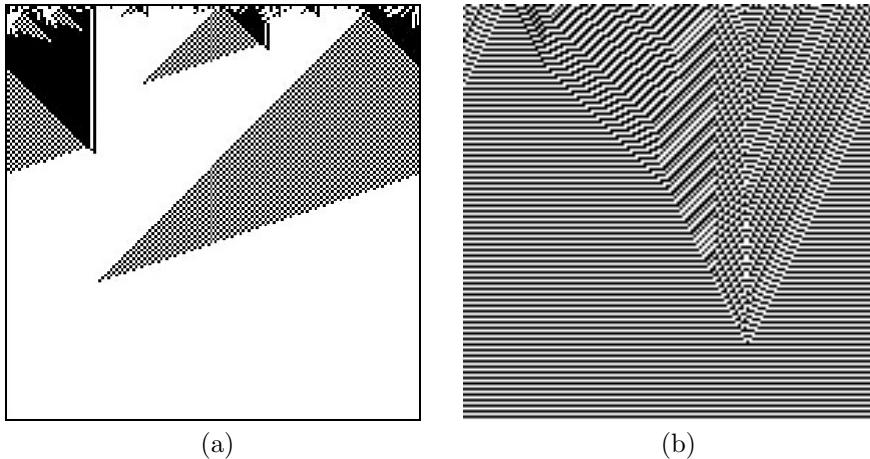


Figure B.2: Time evolution of two binary CAs: cell states are represented horizontally (black stands for 1). Time increases down the page. The initial density of ones is 0.416. (a) Solving the density task. (b) Solving the synchronization task. The size of the automata is $N = 149$.

The Synchronization Task. The one-dimensional synchronization task was introduced in [DCMH95]. In this task the CA, given an arbitrary IC, must reach, within $M \simeq 2N$ time steps, a final configuration that oscillates between all 0s and all 1s on successive time steps. Figure B.2 (b) depicts the space-time diagram of a CA that solves the task.

As with the density task, synchronization also comprises a non-trivial computation for a small-radius CA, and it is thus extremely difficult to come up with CA rules that, when applied synchronously to the whole lattice produce a stable attractor of oscillating all 0s and all 1s configurations. Das et al. [DMC94] were able to automatically evolve very good ring CA rules of radius three for the task by using genetic algorithms. Sipper [Sip97] did the same for quasi-uniform CAs, i.e. CAs with a few different rules instead of just one, attaining excellent performance for radius-one CAs. The performance of a CA on this task is evaluated by running it on randomly generated ICs, uniformly distributed over densities in the range $[0, 1]$, with the CA being run for $M \simeq 2N$ time steps. Evolved CAs for this task have performance P close to 1 (see Section B.4.1).

A simple variant of the majority rule used above for the density task works also for the synchronization task. The rule is called the “contrarian” rule [Wat99], and it operates in the same way as the majority rule, except that it gives the opposite state as output.

This time, CAs performing the synchronization task classify each given IC in one of the following 2 categories: converged or not converged. Compared to the density task, this problem is less complex because the CA only has to relax to a fixed point, there is no correct or incorrect classification.

B.4.1 Tasks Performance on Generalized Automata Networks

Tasks Performance on Evolved Small-World Networks. In previous work [TGD04, TGD05] we used an evolutionary algorithm to evolve GANs of size $N = 149$ that have similar computational capabilities as those found by Watts [Wat99]. Using the majority rule for the nodes (see above), we evolved network topologies starting from populations of slightly modified regular one-dimensional

lattices and from populations of random graphs. For the evolutions, we have used two different fitness functions in order to obtain two different classes of networks, with low and high number of shortcuts (captured by the measure of the ϕ value, which is the fraction of edges in a graph that are shortcuts), and with different average degrees. Without including any preconceived design issue, the evolutionary algorithm was consistently able to find high-performance automata networks in the same class of those constructed by Watts (see Figure B.3). For details of the evolutionary algorithm, see [TGD05]. Figure B.4 depicts the performance of Evolved Small-World CA networks

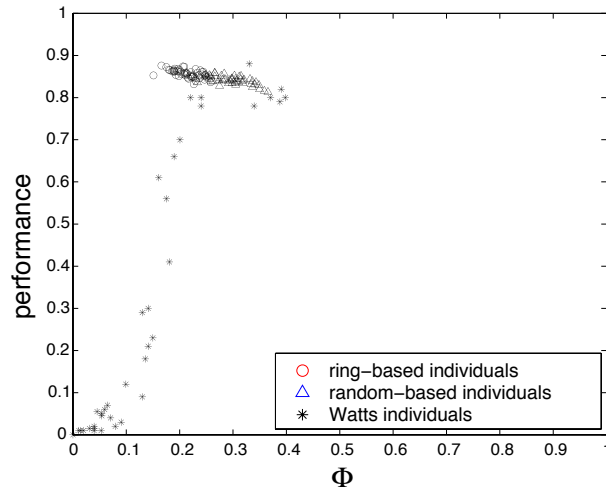


Figure B.3: The ϕ - performance values on the density task of the 50 best individuals found by evolution starting from rings and random graphs using a fitness function that favors low ϕ networks, thus obtaining networks comparable with Watts' hand-constructed networks with $\langle k \rangle = 12$.

on the density and the synchronization tasks (performance is defined in section B.4). We can see the performance of networks evolved with pressure on the ϕ . In the robustness studies of section B.5 of the present article, we only investigate the low- ϕ networks.

The difference in performance between the two tasks is explained by the fact that where the CA has reached a fixed point but with the wrong value on the density task, this case is correct for the synchronization task and is counted in the performance. Clearly, the two tasks are very similar in their rules, performance of these evolved structures is nevertheless excellent considering the fact that the rules for both tasks are extremely simple and do not allow regular lattice CAs to perform the tasks at all.

Tasks Performance on Scale-Free Networks. In accordance to the Barabási–Albert model, we constructed networks of size $N = 149$ to be used as support for CA computations. Knowing that the average degree must remain comparable to our work on the small-world graphs [TGD05], we generated scale-free graphs with $\langle k \rangle = \{6, 12\}$. Following the model, we then defined the range of m_0 values for each k using the derived equation for m . These values of m_0 must respect the constraints $m \leq m_0$ and $m \geq 1$ to ensure the graph is connected. For $\langle k \rangle = 6$, $m_0 \in [4, 25]$ and for $\langle k \rangle = 12$, $m_0 \in [7, 40]$. We assure that even though m is not always an integer, the exact global number of edges in the graph, thus $\langle k \rangle$, is respected in all cases.

Results as represented in Figure B.5 show that performance on the density task of CAs mapped on BA networks are above 0.7 for networks with a smaller m_0 . When a certain threshold is reached (m_0 about 14 for $\langle k \rangle = 6$ and about 35 for $\langle k \rangle = 12$), performances drop dramatically. This means that the more the structure of the scale-free network become star-like, with a unique oversized cluster and only small satellites weakly connected ($m \rightarrow 1$), the information circulates with more difficulties. One can conclude from these results that scale-free network topologies are less suitable than Watts–Strogatz small worlds as a substrate for the density task. The results are even worse

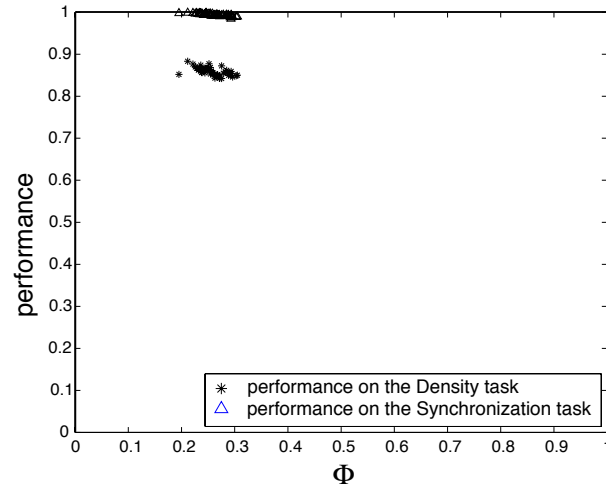


Figure B.4: The ϕ - performance values on the synchronization task of the 50 best individuals found by evolution starting from rings. (a): using a fitness function that does not favor low ϕ networks, thus obtaining networks comparable with Watts' hand-constructed networks with $\langle k \rangle = 6$. (b): using a fitness function that favors low ϕ networks, thus obtaining networks comparable with Watts' hand-constructed networks with $\langle k \rangle = 12$.

than those obtained in rings [CMD03] using specialized rules.

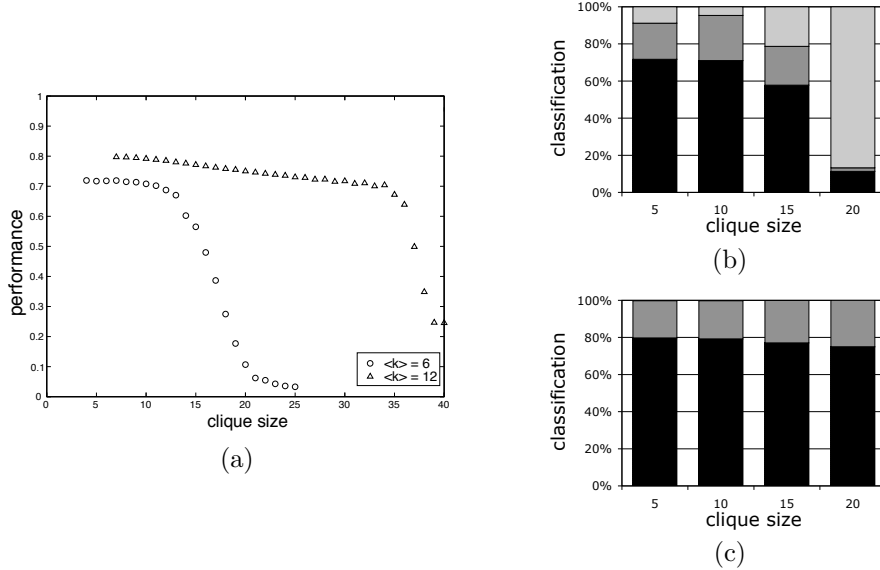


Figure B.5: (a) Performance vs *cliquesize* of scale-free networks (build on the BA) on the density task. The circles represent the performance of networks with an average connection $\langle k \rangle = 6$ and triangles $\langle k \rangle = 12$. On the right, the average percentages of correctly classified ICs (black), incorrectly classified ICs (dark gray) and unclassified ICs (light gray) for sizes of the initial clique in [5,20] by scale-free networks with (b) $\langle k \rangle = 6$ and (c) $\langle k \rangle = 12$. Results are averages over 50 graph realizations.

Figure B.6 shows the performance of the BA scale-free CA networks on the synchronization task. As expected, the performance (in black) is very close to that obtained on the density task,

it is approximately the sum of the proportion of correctly (black) and incorrectly (dark gray) classified showed in Figure B.5 (b) and (c).

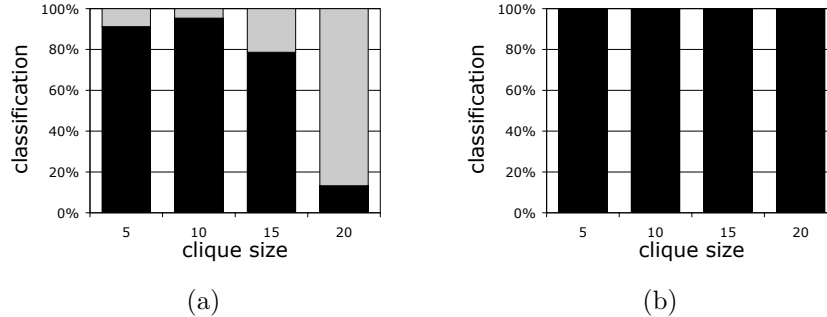


Figure B.6: BA scale-free graphs on the synchronization task. Average percentages of classified ICs (black) and unclassified ICs (light gray) for sizes of the initial clique in $[5,20]$ with (a) $\langle k \rangle = 6$ and (b) $\langle k \rangle = 12$. Results are averages over 50 graph realizations.

In the case of the BA scale-free CA networks, the performance on both tasks does not reach that of our evolved Watts–Strogatz small-world CA networks, where the performance is constantly above 0.8.

The performance on the density task of scale-free CA networks obtained using the CM algorithm with respect to different values of $\gamma \in \{1.5, 2.0, 2.5, 3.0\}$ is shown in Figure B.7. To compare results with previous small-world and scale-free networks, we constructed networks with $\langle k \rangle = 6$ and $\langle k \rangle = 12$ (see Figure B.7). Similarly to Figure B.7, Figure B.8 shows the performance of

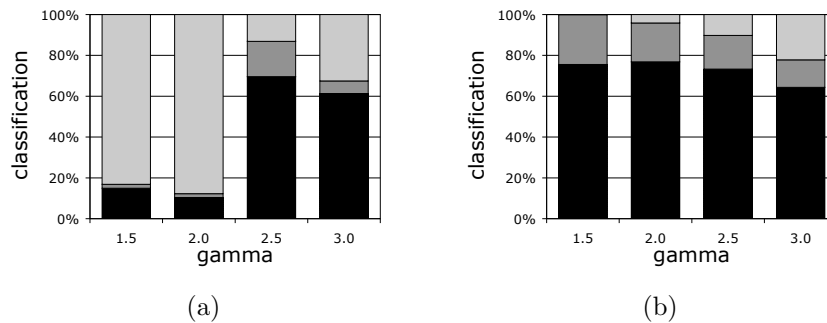


Figure B.7: Configuration Model CA networks on the density task. Average percentages of classified ICs (black), wrongly classified ICs (dark grey), and unclassified ICs (light gray) for values of $\gamma \in \{1.5, 2.0, 2.5, 3.0\}$ with (a) $\langle k \rangle = 6$ and (b) $\langle k \rangle = 12$. Results are averages over 50 graph realizations.

scale-free CA networks obtained using the MCM algorithm with respect to different values of $\gamma \in \{1.5, 2.0, 2.5, 3.0\}$ in the case of the density task. We note that in both cases of CM scale-free CA networks, performance remains below that of the Barabási–Albert scale-free networks. More importantly, we see that for an average degree of $\langle k \rangle = 6$ performance of networks with $\gamma = 1.5$ and $\gamma = 2.0$ are very poor, whereas it remains roughly constant for $\langle k \rangle = 12$ and all values of γ .

All the same performance measurements were performed on the synchronization task, and, as expected, performance in all cases are closely comparable to the sum of correctly and incorrectly classified ICs on the density task as shown in Figures B.7 and B.8. To save space, we do not show these results in this work.

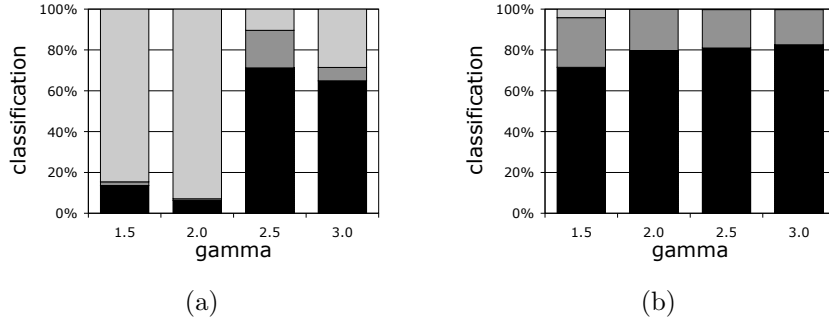


Figure B.8: Modified Configuration Model CA networks on the density task. average percentages of classified ICs (black), wrongly classified ICs (dark grey), and unclassified ICs (light gray) for values of $\gamma \in \{1.5, 2.0, 2.5, 3.0\}$ with (a) $\langle k \rangle = 6$ and (b) $\langle k \rangle = 12$. Results are averages over 50 graph realizations.

B.5 The Effects of Noise and Failures

Noisy environments are the rule in the real world. Since automata networks are toy examples of distributed computing systems, it is interesting and legitimate to ask questions about their fault tolerance aspects. A network of automata may fail in various ways when random noise is allowed. For instance, the cells may fail temporarily or they may die altogether; edges may be cut, or both things may happen. In this section, we shall compare the robustness of standard lattice-CAs to that of small-world and scale-free CAs with respect to a specific kind of transient perturbation and permanent failures, which we respectively call *probabilistic updating* and *edge failure*.

B.5.1 Probabilistic Updating Perturbations

Probabilistic faults are defined as follows: the rule of each cell of a GAN may yield the incorrect output bit with probability p_f , and thus the probability of correct functioning will be $(1 - p_f)$. Furthermore, we assume that errors are uncorrelated. This implies that, for a network with N vertices, the probability $P(N, m)$ that m cells (vertices) are faulty at any given time step t is binomially distributed. It should be noted that we do not try to correct or compensate for the errors, which is important in engineered system but very complicated and outside our scope. Instead, we focus on the “natural” fault tolerance and self-recovering capabilities of the systems under study.

To observe the effects of probabilistic updating on the CA dynamics, two initially identical copies of the system are maintained. One proceeds undisturbed with $p_f = 0$, while the second is submitted to a nonzero probability of fault. We can then measure such things as Hamming distances between unperturbed and faulty configurations, which give information on the spreading of damage [STB96].

Evolved Small-World Networks. As described in Section B.4, caption of Figure B.3, we have extracted the best individual of 50 independent evolutionary runs for both considered starting points (ring-based and random-based). Individuals had evolved towards shortcut proportions ϕ between 0.15 and 0.35. We have studied the performance (or classification abilities) variation of these networks under probabilistic updating for fault probabilities f_p evenly scattered over $[0, 0.3]$. Figure B.9 shows how percentages of initial configurations that are correctly classified (in black), incorrectly classified (in dark gray), and not classified at all (in light gray) change as the fault probability increases. These percentages are averaged out over all contributing individuals to that classification category. Figure B.9(a) depicts the behavior of networks having evolved from perturbed rings, whereas Figure B.9(b) shows that of networks having emerged from random networks.

As pictured in Figure B.9 although the fault probability increases rapidly, our evolved networks show interesting fault tolerance capabilities on the density task up to 10% of faulty outputs. Moreover we note that the proportion of correctly classified ICs compared to the incorrectly classified ones is around 10:1, and this ratio remains almost constant despite the increase in the fault probability. This is especially interesting considering that identifying unclassified ICs is trivial ($2N$ steps and no convergence to a steady state) whereas distinguishing correct from incorrect classification is impossible without knowing the solution beforehand. We conclude that although an increasing number of ICs will not reach a fixed point, the ratio between correctly classified and misclassified ICs remains comparable.

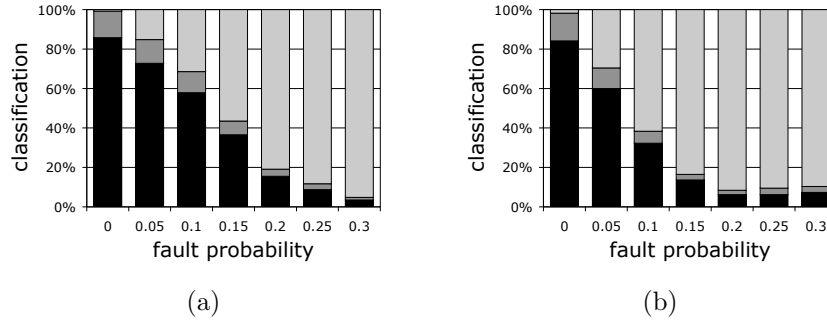


Figure B.9: Small-world CAs on the density task. Percentages of correctly classified (black), incorrectly classified (dark gray) and unclassified (light gray) ICs with increasing fault probability for (a) ring-based networks and (b) random-based networks evolved using a fitness function favoring low ϕ values ($\phi_{(a)} = 0.26$ and $\phi_{(b)} = 0.34$), thus $\langle k \rangle = 12$. Results are averages over 50 graph realizations.

p_f	Ring-based		Random-based	
	correct steps	wrong steps	correct steps	wrong steps
0.0	5.33 [0.27]	8.11 [0.41]	6.35 [0.53]	9.43 [0.73]
0.05	5.78 [0.56]	9.03 [0.92]	6.88 [0.90]	10.78 [1.25]
0.10	6.46 [0.95]	10.89 [1.73]	7.40 [1.56]	13.14 [2.82]
0.15	7.23 [1.68]	12.04 [2.79]	7.98 [2.50]	15.03 [5.63]
0.20	7.34 [2.28]	14.29 [7.52]	11.66 [14.03]	26.86 [20.55]
0.25	20.33 [33.15]	31.94 [38.75]	19.35 [23.82]	41.67 [32.49]
0.30	53.60 [66.11]	81.71 [63.57]	84.05 [70.58]	120.98 [79.07]

Figure B.10: On the density task, the number of time steps necessary to relax to a fixed point for increasing values of fault probability. Column 2 and 3 represent data for ring-based networks and column 4 and 5 for random-based networks ($\langle k \rangle = 12$). The value in square brackets is the standard deviation over the 50 networks of the evaluation.

Table B.10 shows that faultless systems converge in average to a solution in less than ten time steps. This fast convergence is mainly due to the presence of shortcuts through the network. We also see that the number of time steps necessary to reach a fixed point increases exponentially as the fault probability grows. We note in Table B.10 that correctly classified ICs are consistently found in significantly less times steps than incorrectly classified ones. As shown by the standard deviation (in square brackets), this difference is significant for values of fault probabilities up to $p_f = 0.1$, whereas for higher values the results loose their significance. In other words, it would be safe to assume for faults up to 10% that by taking all the converged solutions and discarding the 1/10 that had the slowest convergence, we would be left with almost all correctly classified ICs.

For the synchronization task, results are almost the same and, exactly as expected, the perfor-

mance of our evolved networks under probabilistic faults is approximately the sum of the correctly and incorrectly classified ICs for the density problem. Results are shown in Figure B.11. In addition, the number of steps necessary to relax to a fixed point, although it follows the same tendency as for the density task, is not relevant here to differentiate between correctly and incorrectly classified ICs, as there is no such case anymore. Lastly, we wish to point out for completeness that regular lattice CAs are extremely fragile, and show almost no fault tolerance [TGD05].

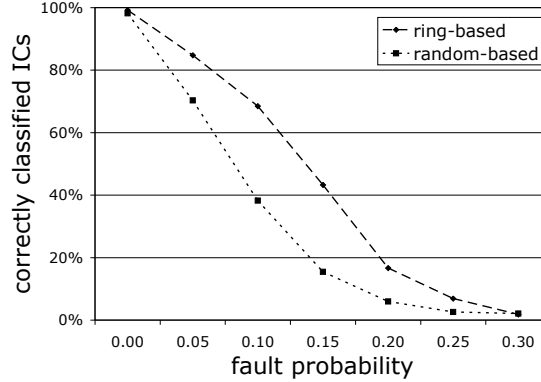


Figure B.11: Evolved small-world CAs on the synchronization task. Percentages of ICs for which the CA solved the task as the fault probability increases for networks starting from a regular structure and starting from random structures. In both cases, $\langle k \rangle = 12$. Results are averages over 50 graph realizations.

Barabási–Albert Scale-Free Networks. To investigate the effects of noise on scale-free automata networks we have used populations of 50 scale-free networks to perform the density task for 7 values of fault probability $f_p \in [0.0, 0.3]$ equally scattered over the interval.

The performance under noise of the above scale-free automata networks is shown in Figure B.12. Figure B.12(a) shows classification details for scale-free networks with an average degree of $\langle k \rangle = 6$ and an initial clique size $m_0 = 5$ and Figure B.12(b) for $m_0 = 10$ and $\langle k \rangle = 12$, for increasing fault probability. These clique size have been chosen because they offer the maximum performance in a noiseless environment (see Section B.4.1). The two cases present the same qualitative behavior. The ratio of correctly to incorrectly classified ICs is close to 1:5. Figure B.12 shows that as the fault probability increases, the ability of performing the collective task is lost. When one compares these results to the small-world case in the above section, we conclude that the fault tolerance is significantly lower in the scale-free case.

These results raise an interesting issue concerning the fault tolerance of scale-free graphs. Indeed, a recent well known result states that scale-free graphs are extremely robust against random node failures, while they have been shown to be fragile when the failures concern highly connected nodes (hubs), a fact that is particularly relevant for the Internet [AJB00]. Thus, scale-free networks are good for information dissemination, i.e. when what counts is that alternative paths remain available in spite of random noise. However, we find that the same structures do not offer such resilience when a task is to be solved in a collective, coordinated way, such as the density task studied here; the results would be even worse if the faults were unrecoverable. We have performed similar computer experiments with scale-free graphs starting from larger cliques. The results, not shown to save space, confirm the findings described above. Again, for fault of small magnitude ($p_f < 0.1$) the number of steps necessary to reach a fixed point is significantly different in the case of correctly and incorrectly classified ICs, thus this measure can help determine whether the classification is correct or not. The qualitative conclusion that we reach from this result is that certain topologies, here Watts–Strogatz networks, perform better and are more robust under fluc-

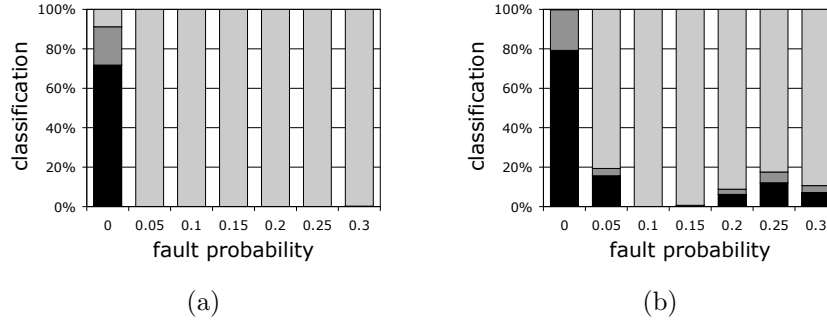


Figure B.12: BA CAs on the density task. Percentages of correctly classified (black), incorrectly classified (dark gray) and unclassified (light gray) ICs as the fault probability increases for (a) an initial clique size of 5 and $\langle k \rangle = 6$ and (b) and initial clique size of 10 and $\langle k \rangle = 12$. Results are averages over 50 graph realizations.

tuating noise than BA scale-free graphs for collective tasks such as the two problems used here. Whether this result generalizes to other similar tasks is currently unknown but would be certainly interesting to investigate.

The fault tolerance of these networks on the synchronization task is comparable to that of the density task described above, as previously the performance in this case evolves as the sum of the correctly and incorrectly classified ICs. Those results are shown in the Figure B.13 below.

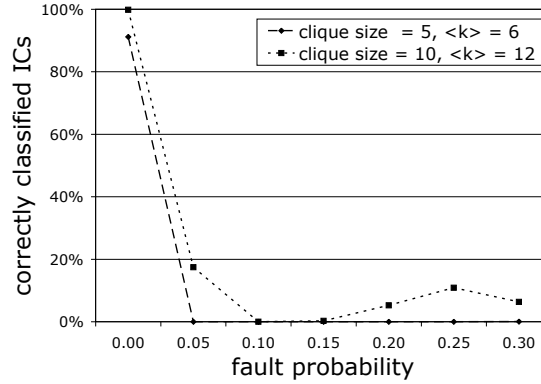


Figure B.13: BA CAs on the synchronization task. Percentages of ICs for which the CA solved the task as the fault probability increases for an initial clique size of 5 and $\langle k \rangle = 6$ and initial clique size of 10 and $\langle k \rangle = 12$. Results are averages over 50 graph realizations.

Configuration Model. As shown above, BA scale-free CA networks have a relatively poor tolerance to probabilistic updating perturbation when solving a collective task. In order to determine if this fact is due to an intrinsic property of scale-free graphs or to the algorithm used to construct them and to the correlations that are present in the BA model, we have built 50 networks using the CM algorithm described in Section B.2 with values of $\gamma \in \{1.5, 2.0, 2.5, 3.0\}$ for both $\langle k \rangle = 6$ and $\langle k \rangle = 12$. We evaluate the fault tolerance of the CA networks on both the density and the synchronization task following the scheme used for the BA scale-free networks and the evolved small-worlds networks. Although we have evaluated the complete set of γ values, topologies

obtained with a value of $\gamma = 1.5$ and 2.0 offered a performance and a tolerance to probabilistic updating perturbation that was close to null. Therefore the Figure B.14 (a) and (c) only show the performance of Configuration Model scale-free networks with values of $\gamma = 2.5$ and (b) and (d) with $\gamma = 3.0$ for both $\langle k \rangle = 6$ and $\langle k \rangle = 12$.

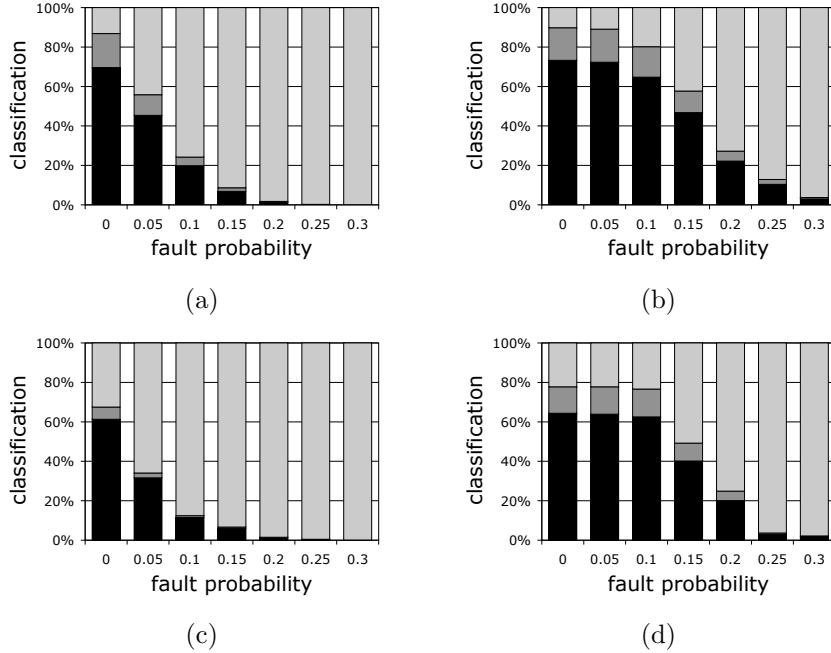


Figure B.14: Configuration model CAs on the density task. Percentages of correctly classified (black), incorrectly classified (dark gray) and unclassified (light gray) as the fault probability increases. The left hand side column (a, c) groups graphs with an average value of $\langle k \rangle = 6$, respectively $\langle k \rangle = 12$ for the right hand side column (b, d). With each pair of graphs, the γ values increases: for (a) and (b) the $\gamma_{(a),(b)}$ value is 2.5 and $\gamma_{(c),(d)} = 3.0$

Whereas we have previously underlined in Section B.4.1 the performance of CM scale-free CA networks consistently remains below that of Barabási–Albert scale-free and our evolved small-world CA networks, we note that these scale-free automata networks offer a tolerance to transient probabilistic faults that is comparable to that of evolved small-world networks and significantly higher than the tolerance of the BA scale-free networks for the computation of a collective task. Finally, we note that results for the synchronization task, not shown here to save space, are completely similar.

Modified Configuration Model. The previous section details the fault tolerance of CA networks built using the Configuration Model algorithm, and we have highlighted the fact that the degree distribution of these networks do not follow as strictly as required the predefined power-law curve because of the constraint on the average degree $\langle k \rangle$ (see Section B.2). Again, Figure B.15 shows the evolution of the performance for an increasing fault probability for $\gamma = 2.5$ only.

In Figure B.15 (a) and (b), scale-free CA networks constructed using the Modified Configuration Model algorithm with $\gamma = 2.5$ demonstrate outstanding tolerance to transient probabilistic faults when compared to all other models we have studied so far in this work. For the synchronization task results shown in Figure B.16 are very similar.

From the results in this section, we can draw the conclusions that, although performance of our evolved small-world networks on both tasks remains by far the highest, scale-free CA networks that have been carefully built by (1) minimizing the degree correlations between the nodes due to the Barabási–Albert model and (2) closely following a power-law curve for the degree distribution,

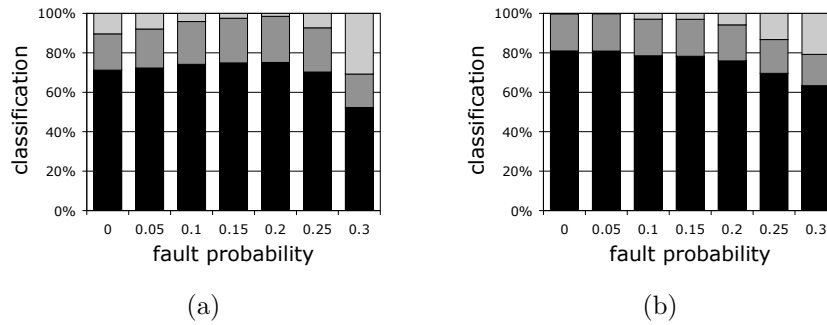


Figure B.15: Modified configuration model CAs on the density task. Percentages of correctly classified (black), incorrectly classified (dark gray) and unclassified (light gray) as the fault probability increases. (a): graphs with an average value of $\langle k \rangle = 6$; (b): $\langle k \rangle = 12$. The γ value is 2.5.

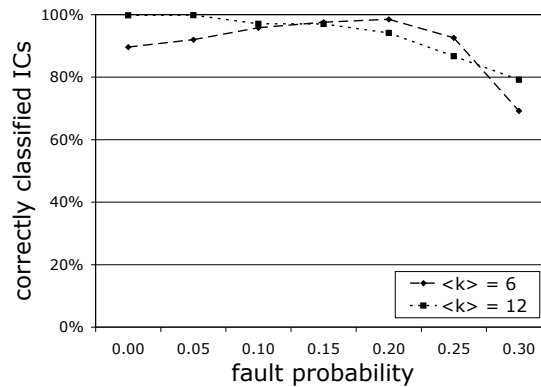


Figure B.16: Modified CM CAs on the synchronization task. Percentages of ICs for which the CA solved the task as the fault probability increases for $\langle k \rangle = 6$ and $\langle k \rangle = 12$. In both cases, $\gamma = 2.5$.

reveal remarkable tolerance to transient probabilistic updating faults for the resolution of a global task. The exact reasons for this behavior are not yet entirely clear to us.

B.5.2 Permanent Link Failures Perturbations

Failures in systems can occur in different ways. The more complex the system, the more failure types and the greater the likelihood. They can range from a one-time wrong output to a complete breakdown. The previous section dealt with transient probabilistic faults that came and went but did not impact the structure of the network in any way. This section describes experiments we have conducted with a much more drastic form of failure, that is the definitive disappearance of an edge between two vertices of the graph, called *permanent edge failure*.

To simulate this kind of irreversible system fault, we have *sabotaged* the networks used all along this work. Establishing a fault probability, we have randomly removed the corresponding number r of edges from the graph, that is for networks with an average degree of $\langle k \rangle = 6$ the values of $r \in \{25, 50, 75, 100, 125, 150, 175, 200\}$ and $r \in \{50, 100, 150, 200, 250, 300, 350, 400\}$ for networks with $\langle k \rangle = 12$ (which represent roughly 6%, 12%, 18%, 25%, 31%, 37%, 43%, and 50% in each case. This notation will be used in Figures B.18, B.20, and B.22, to compare $\langle k \rangle = 6$ with $\langle k \rangle = 12$). In other words, we delete up to almost half of the existing links between the cells of our graphs and evaluate its ability to still performed the requested tasks. To compare with previous

results, we have used an identical scheme to assess the fault tolerance of our CA networks, over 10^4 ICs, allowing a maximum of $2 \times N = 298$ time steps.

Evolved Small-World Networks. Evolved structures have so far proven to possess the best performance in a fault-free environment and have shown high fault tolerance with respect to any CA on regular structures. Figure B.17 shows the behavior of evolved small-world CA networks on the density task in the presence of permanent link failure.

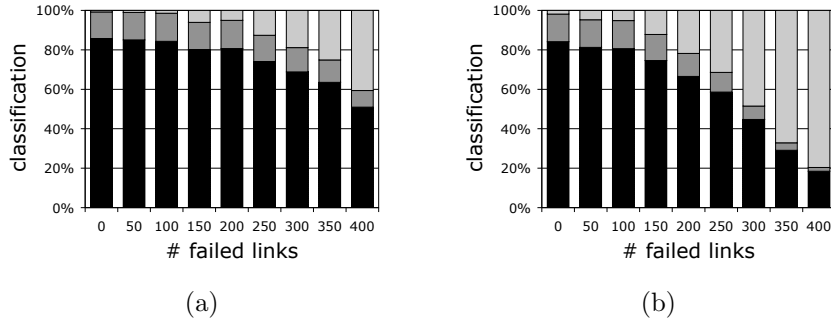


Figure B.17: Permanent faults on evolved Watts–Strogatz graphs. Percentages of correctly classified (black), incorrectly classified (dark gray) and unclassified (light gray) as the number of failed links increases. Left (a): ring based networks; Right (b): random-based networks. The mean degree $\langle k \rangle = 12$ and $\phi \approx 0.3$.

Looking at Figures B.17 (a) and (b) we notice that the ring-based evolved networks have a much higher tolerance to permanent link failures, keeping an almost constant performance until approximately a quarter of the links have failed. In this case, we can conclude that the more organized evolved networks show better robustness for small-world CA networks.

Similar results were obtained on the synchronization task, but, as there is no incorrectly classified IC in this case, the performance curve is approximately the sum of correctly and incorrectly classified ICs in the density task, results are shown in Figure B.18.

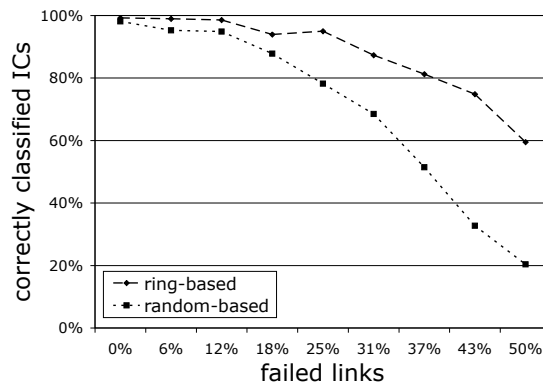


Figure B.18: Evolved Small-World CAs on the synchronization task. Percentages of ICs for which the CA solved the task as the fraction of failing links increases for networks starting from a regular structure and starting from random structures. In both cases, $\langle k \rangle = 12$. Results are averages over 50 graph realizations.

Barabási–Albert Scale-Free Networks. We have also explored the tolerance to permanent link failure of Barabási–Albert scale-free networks. In Section B.5.1 we have shown that the tolerance to transient faults of such structures is below that of other irregular CA networks. Figure B.19 depicts the evolution of the performance of scale-free CA networks as the number of failed links increases. The clique size m_0 for each average degree $\langle k \rangle$ has been chosen so as to have the highest performance in a faultless environment.

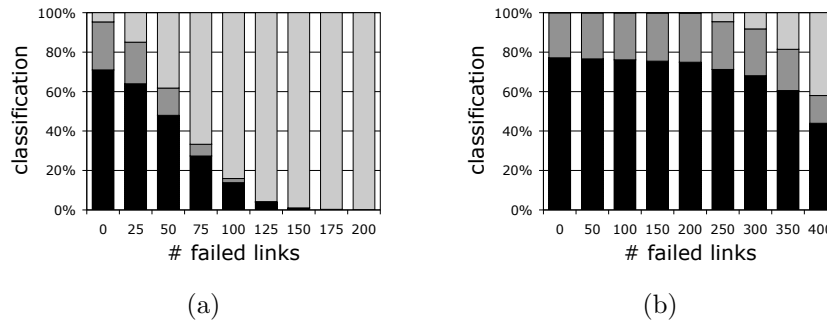


Figure B.19: Barabási–Albert scale-free graphs on the density task. Percentages of correctly classified (black), incorrectly classified (dark gray) and unclassified (light gray) as the number of failed links increases. (a) is a $\langle k \rangle = 6$ population, with a clique size $m = 5$. (b) is a $\langle k \rangle = 12$ population with $m = 10$.

Results as shown in Figure B.19 demonstrate that, if the tolerance to permanent faults of networks with a lower average degree is lower than that of evolved small-world networks, for higher average degree, the results are comparable.

We have run the full set of experiments on tolerance to permanent link failure for the synchronization task, results are shown in Figure B.20 and are very similar to those obtained for the density task.

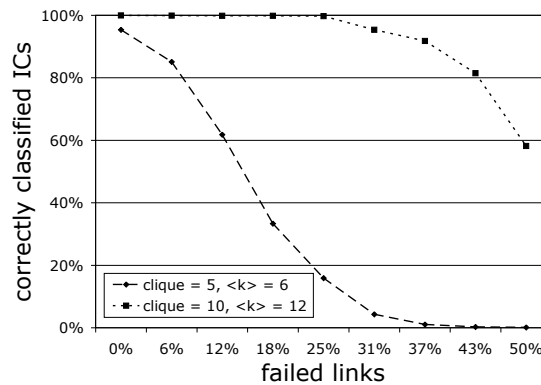


Figure B.20: BA CAs on the synchronization task. Percentages of ICs for which the CA solved the task as the fraction of failing links increases for an initial clique size of 5 and $\langle k \rangle = 6$ and and initial clique size of 10 and $\langle k \rangle = 12$. Results are averages over 50 graph realizations.

Configuration Model Scale-Free Networks. This section will detail the behavior of CA networks built by both Configuration Model algorithms described in Section B.2. We only show the results for the Modified Configuration Model because of their similarity. Although the whole

set of γ values has been numerically simulated, here we only show the case $\gamma = 3.0$, because it is closest to the Barabási–Albert γ value. Figure B.21 shows the evolution of the performance of scale-free CA networks with average degrees of $\langle k \rangle = 6$ and $\langle k \rangle = 12$.

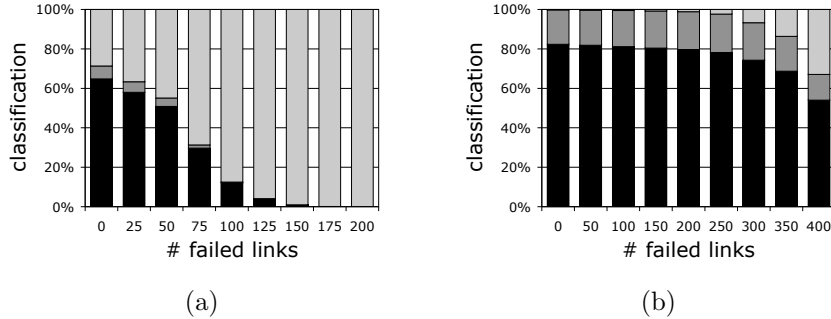


Figure B.21: Modified Configuration Model scale-free CA networks on the density task. Percentages of correctly classified (black), incorrectly classified (dark gray) and unclassified (light gray) as the number of failed links increases. (a) with an average value of $\langle k \rangle = 6$, and (b) $\langle k \rangle = 12$. $\gamma_{(a),(b)}$ value is 3.0

In this case, Figure B.21 (a) shows that Configuration Model scale-free CA networks with a low average degree have an inferior tolerance to permanent faults when compared to other complex structures. Figure B.21 (b) shows that for higher average degrees, Configuration Model scale-free CA networks and other structures display comparable fault tolerance capabilities. The results for the synchronization task, shown in Figure B.22, are analogous.

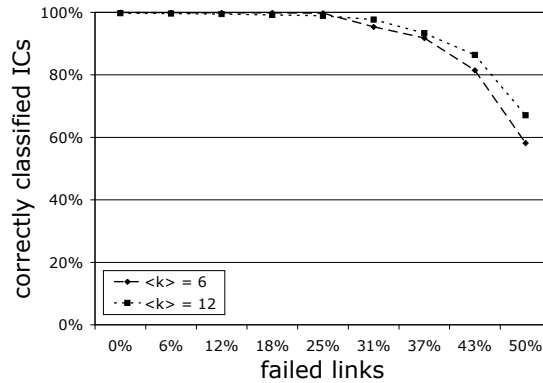


Figure B.22: Modified CM CAs on the synchronization task. Percentages of ICs for which the CA solved the task as the proportion of failing links increases for $\langle k \rangle = 6$ and $\langle k \rangle = 12$. In both cases, $\gamma = 3.0$.

B.6 Conclusions

In this work we have empirically investigated the performances and collective task-solving capabilities of complex networks of automata using the density and synchronization problems as typical cases. We have shown by computer simulations that previously evolved small-world networks have superior performance with respect to scale-free graphs of the Barabási–Albert type, and those built according to the configuration model.

Then we have investigated in detail the fault tolerance capabilities of these network families against, on the one hand, transient uniformly random errors and, on the other hand, permanent link failures. Several interesting conclusions can be drawn. With respect to probabilistic transient faults, the best results are those obtained with evolved small-world networks and Modified Configuration Model scale-free graphs. The latter is even slightly better, except for faultless systems. The worst results are those of Barabási–Albert graphs, which offer virtually no resistance to that kind of perturbation. This is relatively surprising, as it is well known that scale-free networks similar to the Barabási–Albert type offer remarkable fault tolerance against random errors in information transmission contexts, e.g. the Internet. Turning now to permanent link failures, we observe that again, our evolved small-world graphs show outstanding robustness against this more drastic system failure. Configuration model scale-free networks are also good, especially with the higher average degree $\langle k \rangle = 12$. In this case Barabási–Albert graphs show comparable performance degradation. This is understandable as it is related to random link suppression in a context where hubs play a major role. Finally, we point out that all the irregular networked automata studied in this paper are much more tolerant to all kinds of faults than standard regular lattice CAs for both the density and synchronization tasks.

Although, strictly speaking, one cannot generalize these results to other collective problem solving automata tasks, our conclusions should apply, at least in their general qualitative aspects, to other similar collective computational systems. In future work we shall address the problem of trying to explain in more detail the reasons that are behind the widely different behaviors observed here, reasons that are at present not all well understood. We also intend to numerically explore other problems and extend the evolutionary search to the automata rules themselves in the hope that systems with even higher computational performance and robustness will self-organize and emerge.

Part III

Improving Models of Genetic Regulatory Networks

Chapter 6

The Biology Behind It

God, our genes, our environment, or some stupid programmer keying in code at an ancient terminal - there's no way free will can ever exist if we as individuals are the result of some external cause.

Orson Scott Card

Life's information is encoded in *genes* [Pea06, Pen07]. In modern biology, a gene is defined as the basic unit component encoding the heredity in living organisms. Genes contain the genetic information to build and maintain the cells of an organism. A gene is made of a sequence of *nucleic acid* (usually *Deoxyribonucleic acid* or *DNA*). Nucleic acids are macromolecules formed by the basic *nucleotides*: *adenine* (A), *cytosine* (C), *guanine* (G), and *thymine* (T). This sequence of DNA is composed of both *coding* parts, i.e. the actual information on what the gene does, and *non-coding* parts that determine whether the gene is *active*, i.e. *expressed* or not. This second part is also called the *regulating* part. Finally, there is a certain amount of extra DNA that is neither coding nor regulating.

To clarify, a *chromosome* is an organized form of DNA encoding several genes, along with regulating information and other non-coding sequences of *nucleotides*. Figure 6.1 gives an idea as to which elements of the genetic material is a building block of the next.

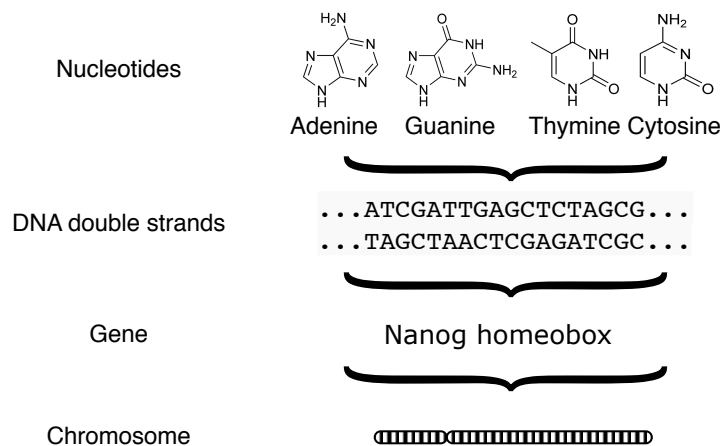


Figure 6.1: Genetic material: how they all fit together.

In living organisms' cells, when a gene is active or activated, both the coding and the non-coding sequences of the gene are copied during a process called *transcription*. The copy resulting from transcription is *ribonucleic acids* (RNA); more precisely messenger ribonucleic acid (mRNA). RNA is very similar to DNA, but differs in a few important structural details: in the cell, RNA is usually single-stranded, while DNA is usually double-stranded; RNA nucleotides also differ slightly with respect to nucleotides of DNA. In *eukaryotic* cells mRNA is then transported outside the *nucleus*, which is the central control of the cell containing the genetic information. In both *prokaryotic* cells (i.e. cells without a nucleus) and *eukaryotic* cells, mRNA directs *enzymes* during the production of *proteins*.

The molecules resulting from gene expression, whether RNA or protein, are known as gene products, and are responsible for the development and functioning of all living things. They are also a central element of the regulation of downstream genes later during the life of the cell (see Figure 6.2). The physical development and phenotype of organisms can be thought of as a product of genes interacting with each other and with the environment [Now06]. Some proteins serve only to activate other genes, and these are the *transcription factors*. By binding to the promoter region at the start of other genes transcription factors activate a gene. On the other hand, some transcription factors are inhibitory.

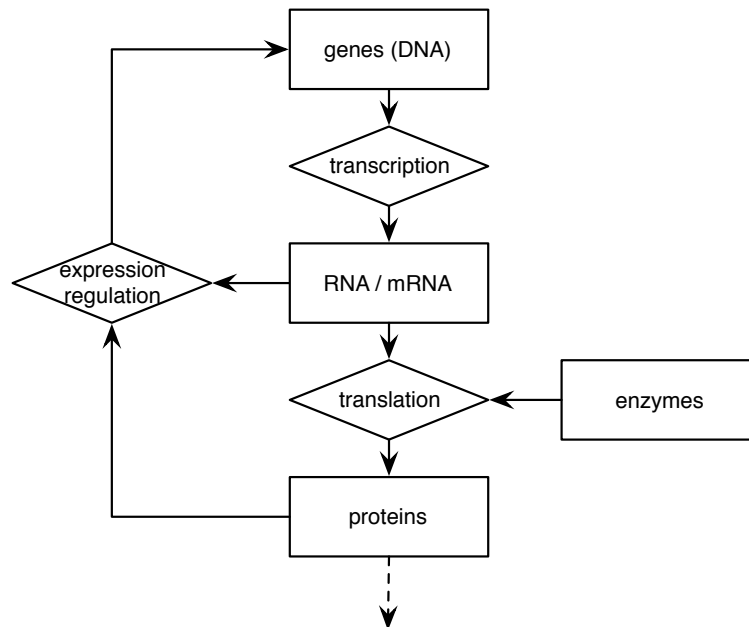


Figure 6.2: Gene regulation: how they regulate each other and themselves.

6.1 Genetic Regulatory Networks

In recent years, high throughput sequencing techniques, such as microarrays, have allowed biologists not only to sequence the entire genome of living organisms, but also to shed some light on the interactions between these genes and how they regulate each other. Nevertheless, the regulation aspect between genes can oftentimes only be inferred by gene *co-expression* data, but not directly witnessed. In a few words, co-expression and co-regulation data compare the expression level of genes at consecutive time intervals, and these time-series are analyzed to find emerging patterns in expressions of genes, which in turn reveals possible regulatory interactions amongst genes.

Gene regulatory networks are formed by genes, messenger RNA (mRNA), and proteins. The interactions between these elements include transcription, translation, and transcriptional regu-

lation [Alb04]. Usually, vertices represent genes and directed edges are the regulating influence (promotion or inhibition) of a gene on another via a protein or an mRNA sequence, as described above. These networks have particular structural topologies that help maintain the stability of the system, while allowing it to evolve. Figures D.1 in the second article below shows two genetic regulatory subnetworks of biological organisms: yeast and mouse stem cell.

The dynamic processes taking place in regulatory networks are extremely complex and we are just beginning understanding them in detail. However, it is possible, and useful, to abstract many details of the particular kinetic equations in the cell and focus on the system-level properties of the whole network dynamics. This complex systems biology approach, although usually not strictly applicable to any given particular case, may still provide interesting general insight.

Chapter 7

Modeling Genetic Regulatory Networks

Essentially, all models are wrong,
but some are useful.

George E. P. Box

Models, as mentioned earlier are simplified representations of the reality submitted to constraints and conditions. In the case of genetic regulatory networks there are essentially four types of models [BB01]. Boolean networks that we will describe in details in the next section, and three more:

- *differential equations systems* used to describe the reaction kinetics of the constituent parts. These models usually involve an in-depth understanding of the temporal variation of the concentration of the network's substances. The functions are ultimately derived from basic principles of chemical or enzymatic kinetics [EG06];
- *continuous networks* are an extension of the Boolean model described below, only this time the genes expression level is assumed to be a continuous function of time. It has been argued that using a continuous representation captures several properties of gene regulatory networks not present in the Boolean model [Voh01];
- *stochastic gene networks* are GRNs models reflecting recent experimental results [BKCC03, ELSS02] hinting that gene expressions are stochastic processes. Some formalisms of this phenomenon have been proposed [ARM98], and several works on single genes and small synthetic networks have been conducted [GCC00, EL00, RO05].

The structural topology of the GRNs, and consequently of models thereof, is still open to discussion, but one property has been agreed by all parties that GRNs are sparse networks (see Section 2.2.1) and that the upstream-regulator per gene is less than two [Lec08]. In the early random Boolean network model [Kau69], the network topology was random. Later, more complex network topology with different input and output degree distributions were proposed [Ald03] (see next section).

Another subject of discord in the community is the timing of events in the models, more specifically, the fact that most models assume that the regulation, activation, and expression of the genes is taking place instantaneously and that these phenomena take the same time for all genes. Although the order of magnitude is comparable amongst, biologists feel this approximation might be a real weakness of the models. Alternate models were proposed where these reactions were delayed reactions in order to account for the time it takes for the entire process to be complete [RZ06].

7.1 Random Boolean Networks

Introduced by Kauffman in the late 60's, Random Boolean Networks (RBNs) are early models of genetic regulatory networks [Kau69]. RBNs have been studied in detail by analysis and by computer simulations of statistical ensembles of networks and they have been shown capable of surprising dynamical behavior. An excellent review on the topic can be found in [Dro08].

In Kauffman's RBNs (known today as Classical RBNs) with N nodes, a node represents a non-discriminate gene and is modeled as an Boolean *on/off* device, meaning that a gene can only either be expressed if it is on (1), and it is not expressed otherwise (0). Each gene receives exactly K randomly chosen inputs from other genes (see Figure 7.1).

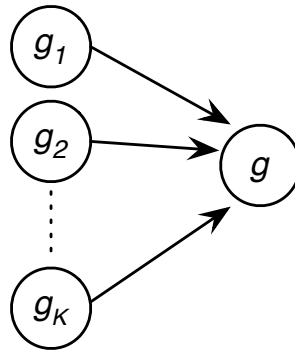


Figure 7.1: Gene interaction in RBN: the activation state of genes g_1, g_2, \dots, g_K have an influence on gene g .

From a simplistic viewpoint, the combined effect of proteins produced by genes g_1 to g_K attaching to an mRNA binding site, thus either promoting or repressing the activity of the target gene g , can be seen as a direct effect of a function $f(g_1, \dots, g_K, g, t) \rightarrow g^{t+1}$. In this case, we allow g to be one of the arguments of the gene update function f , thus permitting self-regulation. If we assume all genes are Boolean nodes, we can define the activity of any gene at time $t + 1$ as the result of a Boolean function of each of the gene's entries at time t . The *bias*, or probability p for a node to be expressed at the next time-step is the only variable parameter of the Boolean function.

Each gene's Boolean function is chosen arbitrarily and can be seen as a randomly generated lookup table with 2^{K+1} entries. The model evolves through time by discrete time-steps and changes to the genes state of activations are instantaneous. Every gene updates its state at every time-step.

The state of a gene g_i at any given time t is $g_i(t) \in \{0, 1\}$. For a RBN with N genes, the *state* or *configuration* $C(t)$ of the entire RBN at time t is defined by the binary string $C(t) = (s_0(t), s_1(t), \dots, s_{N-1}(t))$. A finite size RBN with N genes has a total of 2^N possible configurations called the *state/phase space*. As states are binary strings, they can be identified by a unique integer i represented by the binary string: $i = \sum_{j=0}^{N-1} s_j 2^j$.

The time-evolution of RBN is fully deterministic. Once the Boolean functions have been attributed to the genes, an *initial configuration/state* is set to the RBN at time $t = 0$. In this initial state, each gene is given a random expression value, either *on* or *off*. At each time-step, each gene potentially updates its own state according to its Boolean function and the RBN finds itself in a potentially different state $C(t) \rightarrow C(t + 1)$. Independently of its initial condition, a RBN will travel through a number of states before relaxing into a subset of configurations called an *attractor* and cycle through the states of the attractor. The number of states forming an attractor is called the *attractor's cycle length* l . The length of an attractor varies in: $1 \leq l \leq 2^N$. The state-space of a RBN can contain more, but no less, than one attractor. The ensemble of configurations leading to an attractor is called the *basin* of this attractor.

A basin of attraction is made of three types of configurations: *garden-of-Eden*, *transitory*, and *attractor* states. Garden-of-Eden states cannot be reached by the dynamics of the systems itself,

it can only be set as an initial configuration. Transitory states are traversed only once. Finally, attractor states are part of the cycle the system goes through once it has relaxed to stability.

Figure 7.2 shows the possible time evolution of an example of RBN of size $N = 4$ and $K = 3$. This system has therefore $2^4 = 16$ possible configurations. In this example, we do not specify the Boolean functions, but instead show how the RBN transitions from any possible state. Garden-of-Eden states are a darker shade of grey, transitory states are light gray and double lines delimit the attractor states. In this particular example, the entire state-space belongs to the same basin of attraction leading to the unique attractor of this specific system. One can assume that if the Boolean function were different, the entire state-space transitions would be completely different as well.

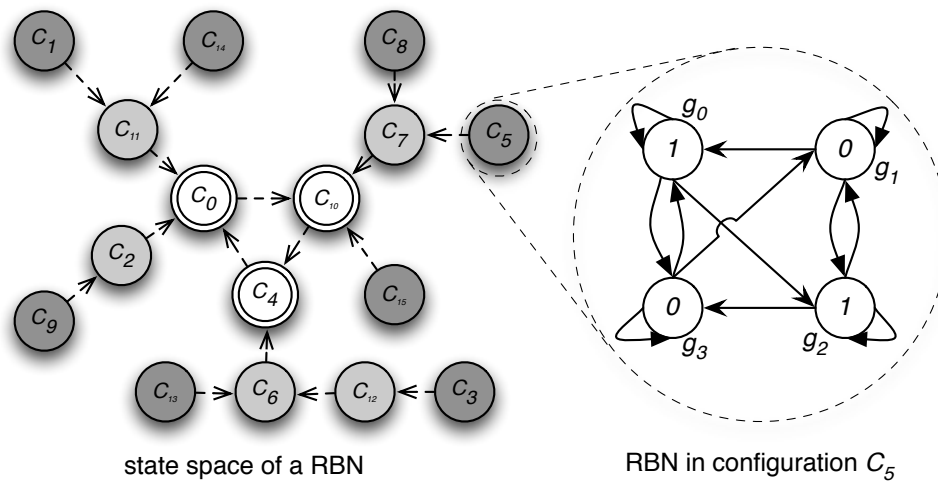


Figure 7.2: state-space of a RBN. An example with $N = 4$, and all $2^4 = 16$ states of the state-space belong to the same attractor of length 3.

RBN systems evolve dynamically over time in either the *ordered regime* or *ordered phase* or the *chaotic regime/phase*. The regime in which the RBN has been set can be identified according to the proportion of nodes that are actively participating in an attractor by flipping their states “often”. In other words, assume that we can define two categories for the nodes of a system in an attractor: *frozen* and *twinkling* [Kau00]. Frozen nodes are those whose state remains unchanged for a long time, say fifty time steps. On the contrary, twinkling ones change their state frequently. In the ordered regime, the proportion of frozen nodes grows linearly with the network’s size N , and a vast majority of the nodes are frozen. In the chaotic regime a majority remain twinkling. Finally at the critical regime, or so-called edge-of-chaos, the number of twinkling and frozen nodes is comparable. Another critical feature distinguishing the ordered from the chaotic regime is that in the first one, the length of the attractors scales as a polynomial or superlinear [Dro08] function with the size of the network, whereas in the chaotic regime, it grows exponentially.

In Kauffman’s original RBN systems, the edge between order and chaos is achieved when $K = 2$ and the bias $p = 0.5$. Kauffman speculates that living organisms operate at this *critical regime*, at the edge between the ordered and the chaotic phase. This condition helps organisms to achieve a tradeoff between the stability of order and the robustness of chaos. See Section C.2 for more details.

7.2 Extension of Random Boolean Networks

With the considerable advances in molecular biology that we have witnessed within recent years, some aspects of the original RBN have been questioned. For each of these aspects, one or more new models, or adaptations of the RBN model, have been proposed. Here, we mention a few:

- *the topological structure*: as more and more details are unveiled about GRNs, the certitude is that their structure is not that of a random graph. Recent findings hint to a network topology with asymmetrical input and output degree distributions. Indeed, Aldana proposes a RBN model with scale-free topologies, that is, with a *power-law* (see Section 2.2.2) output degree distribution and a *Poisson* or *normal* input degree distribution [Ald03, SVA04]. This new model shows singular robustness that can directly be attributed to the topological properties of the underlying network [AC03b].
- *the synchronicity*: when considering biological phenomena, the synchronicity of the events taking place in organisms is a questionable assumption. Nevertheless, it was an understandable simplification of Kauffman's original model. More recently, asynchronous models have been proposed and display dynamical behaviors that are in agreement with previous results, yet more biologically interesting [MT03, Ger04].
- *the instantaneity*: in real life, gene activation time is in the order of seconds or minutes, protein decay time is between minutes and hours, and those time are dependent on the gene or protein in question. Approximating those times to instantaneous happenings is a gross oversight of biological reality. On the other hand, in order for time delays to actually be an asset to the model, a deeper understanding of the biochemical properties of genes, proteins and mRNA is crucial, and we only now begin to have access to these data. Several models that account for the time it takes biochemical reactions to occur, notably by taking an arbitrary chosen number of time-steps to refresh a gene's expression state [RZ06].
- *the Boolean state of expression*: continuous models have been mentioned at the beginning of this chapter.
- *the random Boolean update function*: although the exact combined effect of the proteins and/or mRNA is still unclear, and thus the actual Boolean update function is unknown, it is safe to assume that these function are not random. We already have mentioned stochastic networks as an alternative. Another interesting scheme was suggested that takes into account the actual promoting or repressing effect of genes known today, and combining them in an additive way [LLL⁺04].

These shortcomings of the original RBN model have been at the center of our work on Boolean model for genetic regulatory networks. The detail of this work will be presented in Articles C and D.

Chapter 8

Biologically Inspired Faults

All sorts of computer errors are now turning up. You'd be surprised to know the number of doctors who claim they are treating pregnant men.

Isaac Asimov

Living organisms are robust to a great variety of genetic changes, and since RBNs are simple models of the dynamics of biological interactions, it is interesting and legitimate to ask questions about their fault tolerance aspects.

Kauffman defines a one type of perturbation to RBNs as “gene damage” [Kau00], that is the transient reversal of a single gene in the network. These temporary changes in the expression of a gene are extremely common in the normal development of an organism. The effect of a single hormone can transiently modify the activity of a gene, resulting in a growing cascade of alternations in the expression of genes influencing each other. This is believed to be at the origin of the cell differentiation process and guides the development.

The effect of a damaged gene can be measured by the size of the avalanche resulting from that single gene changing its behavior from active to inactive or vice-versa. The size of an avalanche is defined as the number of genes that have changed their own behavior at least once after the perturbation happened. Naturally, this change of behavior is compared to an unperturbed version of the system that would be running in parallel. The size of the avalanche is directly related to the regime in which the RBN is; in the ordered regime, the cascades tend to be significantly smaller than in the chaotic regime. In real cells, where the regime is believed to lie on the edge of chaos, the cascades tend to be small too. Moreover, the distribution of the avalanche sizes in the ordered regime follows a power law curve [Kau00], with many small and few large avalanches. In the chaotic regime, in addition to the power law distribution, 30-50 % of all avalanches are huge. The distribution of avalanches size of RBNs in the ordered regime roughly fits the expectations of biologists, where most of the genes, if perturbed, are only capable of initiating a very small avalanche, if any. Fewer genes could cause bigger cascades, and only a handful can unleash massive ones.

Another measurement of the effect of transient gene reversal, is to compare the change in the configuration of the RBN between two consecutive time steps on an unperturbed system and on one where a single gene has been perturbed. The difference between two consecutive states s_t and s_{t+1} of the system is measured in terms of Hamming distance, that is the number of genes that have changed their expression between s_t and s_{t+1} , normalized over the network size.

Naturally, one can imagine more sophisticated failure schemes on models of genetic regulatory networks such as RBNs. These failures are usually inspired by scientific experiments conducted on biological organisms. For example the gene knockout experiment measures the expression level of all genes, in cells which a knockout gene and in normal cells, using cDNA microarray data.

Serra and coauthors [SVS04, SVGK07] used this type of failure on RBNs to predict the size of real avalanches on microarray data. They showed that a very simple model with few inputs and random topologies can approximate the distribution of perturbation in gene expression levels with respect to microarray data. Moreover, they present a theoretical study showing that this simple model is actually valid in a particular type of network topologies.

Another notable perturbation inspired by real biological regulatory networks applied to RBNs is the gene duplication phenomenon suggested in by Aldana and coauthors [ABKR07]. They study the robustness of genetic regulatory networks using RBNs and explore their behavior when exposed to nature-inspired genetic perturbations: gene duplications. They show that an intrinsic property of such networks is to tend to preserve and multiply previous phenotypes, encoded in the attractor state-space of the network.

Chapter 9

Articles in this Part

Life emerged, I suggest, not simple, but complex and whole, and has remained complex and whole ever since.

Stuart Kauffman

Articles C and D deal with improving Boolean models of genetic regulatory networks inspired by Kauffman's random Boolean networks. We focused on finding modern biological discoveries to include in the model in order to overcome some of the shortcomings that we draw attention to at the end of Section 7.1. We worked on the topological aspects, the functional aspects and on the dynamical aspects of Boolean models. We have included recent biological findings at each step and each iteration of our new models, and compared them with original RBN both in terms of performance (i.e. qualities of the attractors) and robustness in the face of small perturbations.

In Article C

In this work we focus on the structural and dynamical aspects of Boolean models for GRNs. Indeed, we propose the use *generalized Boolean networks*, a broadening of the random Boolean model, where the topology does not have to be based on a random graph. Instead, we apply the concepts coined by Aldana and generate networks of the scale-free type as substrate for our Boolean models. These scale-free Boolean networks (SFBNs) have a long-tailed power-law output degree distribution and a Poisson-like input degree distribution, close to that of random graphs. These topologies are believed to be much closer to biological reality, where a handful of genes (i.e. hubs) produce proteins that regulate a large number of genes, and most of the genes only have an extremely limited regulating effect (i.e. leaves). Nevertheless, in order for a network to be scale-free, its degree distribution ought to cover several orders of magnitude, and Aldana's original ones were limited to a maximum of 19 genes. In our case, we build SFBNs with a maximum of 200 genes. This tenfold scaling offers interesting insight on the scaling capabilities of the model, it however makes the model very demanding in computational resources. Therefore, we had to resort to statistical sampling.

As a second improvement we have fitted the model with is a more biologically plausible timing of events. Indeed, as argued in the previous chapter, the full synchronicity is clearly a biological improbability. Thanks to microarray experiment data, biologists believe that the activation of a gene induces a change of activation status in a subset of genes, and so on. Therefore, the regulating effect of genes could tend to impact regions of the genetic regulatory network, instead of the whole system simultaneously. This can be seen as a cascading effect, and therefore we introduce a novel update timing we call Activated Cascade Update (ACU). This update scheme is explained in details in the article, and mimics the phenomenon witnessed in real-life. The dynamical behavior and robustness of Boolean networks with both random and scale-free topologies using ACU are

compared to those of synchronous classical RBNs and SFBNs. We run extensive simulations, and thoroughly study all different scenarios. Results detailed in the article are generally favorable to our new, more biologically relevant model.

In Article D

In the final chapter of this work, we have studied the possibility of using sub-networks of GRNs of biological organisms. We faced several challenges when selecting candidate GRNs. The portions we needed had to be as self-standing as possible, with minimal external output, and the confidence in the gene interaction and their kind had to be acceptable. We finally opted for two portions of GRN, one of the yeast cell-cycle and one of mouse embryonic stem cells, details are in Section D.2. The straightforward use of these networks is to replace the nodes by Boolean models of gene expression values, and use them as Boolean model. Although, using real-life biological GRN structures overcomes in a new way the topological flaw of the original RBN model, it raises a new problem. Until now, the critical regime was achieved by tuning the gene expression probability p in the Boolean function, the network average degree, or the power-law exponent. This latter parameter is now a constant fixed by the network. Therefore, we have an empirical method to derive the regime from the systems dynamics using *Derrida plots* (described in Section D.3.1) and we propose a new numerical metric that measures the system's distance to criticality: the *criticality distance* in Section D.3.2. This new measure has, in turn, allowed us to tune the p value so that our systems operate indeed in the critical regime.

Additionally, we worked on closing another gap of Kauffman's original model by proposing an alternative Boolean function that is closer to current biological knowledge. From the literature and our collaboration with biologists, we were able to leverage the extra information in the studied GRNs presented above. Indeed, the yeast and stem cell partial GRNs we used not only provided us with the existence of gene on gene influence, but also their kind: *promoting* or *repressing*. We propose a Boolean function that adds the combined effect of the genes, and uses a threshold T value for gene activation or deactivation, the Activator Driven Additive function detailed in Section D.4. In every case we were able to determine a T value that makes our systems critical. Results analyzed in the article are very encouraging, both in terms of the kind of attractors found, that agree with our expectations according to the regime, and in terms of robustness, which has not suffered from the biological input to the model.

Finally, we use a third GRN sub-networks (in Section D.5.1), this one from plant biology, where the actual function of each gene model in the network has been established. This new case study clearly shows it operates in the critical regime. We use this more complete case study to validate our ADA model. The results are excellent, and prove that in this particular case, ADA functions are much closer to real-life than random Boolean function with an overlap of approximately 92%.

In this last part of the work, we took some liberties with the term "real-life". The three genetic regulatory networks we use in this article are parts, or sub-networks, of biological organisms. These sub-networks are composed of genes that have been identified as playing a key role in a specific process within the entire organism: pluripotency in the case of mouse embryonic stem cells, cell-cycle in yeast, and finally plant guard cell abscisic acid (ABA) signaling. The systems studied suffer some obvious limitations, mainly that none of the networks is actually completely self-sufficient, thus some degree of interaction with the rest of the organism has been omitted. If we believe the methodologies can be generalized to bigger/different systems, we are aware that the results are limited to the studied organisms, under the strict conditions specified in the work, and do not reflect the complexity of "real-life" in every aspects.

Article C

Dynamics of Unperturbed and Noisy Generalized Boolean Networks

Authors: Ch. Darabos, M. Tomassini, M. Giacobini
Published in: Journal of Theoretical Biology
Volume: 260
Pages: 531–544
Year: 2009

Abstract

For years, we have been building models of gene regulatory networks, where recent advances in molecular biology shed some light on new structural and dynamical properties of such highly complex systems. In this work, we propose a novel timing of updates in Random and Scale-Free Boolean Networks, inspired by recent findings in molecular biology. This update sequence is neither fully synchronous nor asynchronous, but rather takes into account the sequence in which genes affect each other. We have used both Kauffman's original model and Aldana's extension, which takes into account the structural properties about known parts of actual GRNs, where the degree distribution is right-skewed and long-tailed. The computer simulations of the dynamics of the new model compare favorably to the original ones and show biologically plausible results both in terms of attractors number and length. We have complemented this study with a complete analysis of our systems' stability under transient perturbations, which is one of biological networks defining attribute. Results are encouraging, as our model shows comparable and usually even better behavior than preceding ones without losing Boolean networks attractive simplicity.

Keyword

Random Boolean Networks, Complex Networks, Boolean Dynamics, Scale-Free Networks, Genetic Regulatory Networks, Perturbations

C.1 Introduction

Gene regulatory networks comprising genes, proteins and other interacting molecules, are extremely complex systems and we are just beginning understanding them in detail. However, it is possible, and useful, to abstract many details of the particular kinetic equations in the cell and focus on the system-level properties of the whole network dynamics. This Complex Systems Biology approach, although not strictly applicable to any given particular case, may still provide interesting general insight.

Random Boolean Networks (RBNs) have been introduced by Kauffman more than thirty years ago [Kau69] as a highly simplified model of gene regulatory networks (GRNs). RBNs have been studied in detail by analysis and by computer simulations of statistical ensembles of networks and they have been shown to be capable of surprising dynamical behavior. We summarize the main results in the next section.

In the last decade, a host of new findings and the increased availability of biological data have changed our understanding of the structure and functioning of GRNs. In spite of this, the original view of Kauffman has been used to predict gene expression patterns observed experimentally [BS01, ABCA⁺08]. Today, this model is still valid provided that it is updated to take into account the new knowledge about the topological structure and the timing of events of real-life gene regulatory networks without losing its attractive simplicity. Following these guidelines, our aim in this work is to describe and test a new model that we call Generalized Boolean Networks (GBNs), which includes, at a high level of abstraction, structures and mechanisms that are hopefully closer to the observed data. Adhering to the original Kauffman's view that attractors of the dynamics of RBNs are the important feature and that they roughly correspond to cell types, we will discuss the results of the systems ability to relax into stable cycles or fixed points, and their tolerance to local perturbation.

The organization of this work is the following. In the next section we briefly review the main assumption implied in Kauffman's RBNs and their possible limitations. Changes to both the randomness and the synchrony assumptions will be proposed in section C.3 leading to generalized boolean networks. In sections C.4 and C.5 the new model is studied by statistical sampling using numerical simulation. Then we introduce the concept of perturbation in section C.6 and we investigate numerically the stability properties of GBNs. Finally, in section C.7 we present our conclusions and discuss possible future work.

C.2 Classical Random Boolean Networks

Random Boolean Networks (RBNs) have been introduced by Kauffman [Kau69] as a highly simplified model of gene regulatory networks. In Kauffman's RBNs with N nodes, a node represents a gene and is modeled as an on-off device, meaning that a gene is expressed if it is on (1), and it is not otherwise (0). Each gene receives K randomly chosen inputs from other genes. Initially, one of the 2^{2^K} possible Boolean functions of K inputs is assigned at random to each gene. The network dynamics is discrete and synchronous: at each time step all nodes simultaneously examine their inputs, evaluate their Boolean functions, and find themselves in their new states at the next time step.

More precisely, the *local transition rule* ϕ is one of the $2^{2^{K+1}}$ possible Boolean functions of K inputs from the neighboring nodes plus that of the node itself, thus possibly implementing a biological situation where a gene regulates itself:

$$\phi : \Sigma^{K+1} \rightarrow \Sigma.$$

This function maps the state $s_i \in \Sigma = \{0, 1\}$ of a given node i into another state from the set Σ , as a function of the state of the node itself and of the states of the nodes that send inputs to i . For a finite-size system of size N (such as those treated herein) a *configuration* $C(t)$ of the RBN at time t is defined by the binary string:

$$C(t) = (s_0(t), s_1(t), \dots, s_{N-1}(t)),$$

where $s_i(t) \in \Sigma$ is the state of node i at time t . The progression of the RBN in time is then given by the iteration of the *global mapping*, also called *evolution operator* Φ :

$$\Phi : C(t) \rightarrow C(t+1), \quad t = 0, 1, \dots$$

through the simultaneous application at each node of the non-uniform local transition rule ϕ . The global dynamics of the RBN can be described as a directed graph, referred to as the RBN's

phase space. Over time, the system travels through its phase space, until a point or cyclic attractor is reached whence either it will remain in that point attractor forever, or it will cycle through the states of the periodic attractor. Since the system is finite and deterministic, this will happen at most after 2^N time steps.

This extremely simple and abstract model has been studied in detail by analysis and by computer simulations of statistical ensembles of networks and it has been shown to be capable of surprising dynamical behavior. Complete descriptions can be found in [Kau93, ACK03]. We summarize the main results here.

First of all, it has been found that, as some parameters are varied such as K , or the probability p of expressing a gene, i.e. of switching on the corresponding node's state, the RBN can go through a phase transition. Indeed, for every value of p , there is a critical value of connectivity:

$$K_c(p) = [2p(1-p)]^{-1}$$

such that for values of K below this critical value $K_c(p)$ the system is in the ordered regime, while for values of K above this limit the system is said to be in the chaotic regime. In classical RBNs $K_c(0.5) = 2$ corresponds to the edge between the ordered and the chaotic regime, systems where $K < 2$ are in the ordered regime, and $K > 2$ means that the system is in the chaotic phase for $p = 0.5$.

In his original work, Kauffman discovered that the mean cycle length scales are at most linear with N for $K = 2$. He also believed that the number of attractors scales with the square root of the number of genes in the system, which has an interesting analogy with the number of different cell types for genomes in multicellular organisms. In fact, this last hypothesis has been proven to be an artifact of under-sampling by Bilke and Sjunnesson [BS01] who showed that the number of attractors scales linearly with N . In addition, Kauffman found that for $K = 2$ the size distribution of perturbations in the networks is a power-law with finite cutoff that scales as the square root of N . Thus perturbations remain localized and do not percolate through the system. Kauffman's suggestion was that cell types correspond to attractors in the RBN phase space, and only those attractors that are short (between one and a few tens or hundreds of states) and stable under perturbations will be of biological interest. Thus, according to Kauffman, $K = 2$ RBNs lying at the edge between the ordered phase and the chaotic phase can be seen as abstract models of genetic regulatory networks. RBNs are interesting in their own as complex dynamical systems and have been thoroughly studied as such using the concepts and tools of statistical mechanics (see [DP86, ACK03]).

For the sake of completeness, let us mention that the "discrete" approach to the high-level description of genetic regulatory networks is not the only possible one. A more realistic description is obtained through the use of a "continuous-state" model. In the latter, the levels of messenger RNA and proteins are assumed to be continuous functions of time instead of on/off variables. The system evolution is thus represented by sets of differential equations modeling the continuous variation of the components concentration. Here we focus on the discrete approach, but the interested reader can find more information on the continuous models in [EG06], for instance.

C.3 From Random to Generalized Boolean Networks

In this section we describe and comment on the main assumptions implied in Kauffman's RBNs. Following this, we propose some modifications that, in our opinion, should bring the model closer to known facts about genetic regulatory networks, without losing the simplicity of classical RBNs. Kauffman's RBN model rests on three main assumptions:

1. Discreteness: the nodes implement Boolean functions and their state is either on or off;
2. Randomness: the nodes that affect a given node in the network are randomly chosen and are a fixed number;
3. Timing: the dynamics of the network is synchronous in time.

C.3.1 Discrete State Approach

The binary state simplification could seem extreme but actually it represents quite well “threshold phenomena” in which variables of interest suddenly change their state, such as neurons firing or genes being switched on or off. This can be understood since the sigmoidal functions one finds in the continuous differential equation approach [EG06] actually do reduce to threshold gates in the limit, and it is well known that Boolean functions can be constructed from one or more threshold gates [Has95]. So, in the interest of simplicity, our choice is to keep the discrete Boolean model for the states of the nodes and the functions implemented at each node.

C.3.2 Random vs Scale-Free Networks

RBNs are directed random networks. The edges have an orientation because they represent a chemical influence from one gene to another, and the topologies of the graphs are random because any node is as likely to be connected to any other node in an independent manner. There are two main types of RBNs, one in which the connections are random but the degree of each node is fixed, and a more general one in which only the average connectivity is fixed. Random graphs with fixed connectivity degree were a logical generic choice in the beginning, since the exact couplings in actual genetic regulatory networks were largely unknown. Today it is more open to criticism since it does not correspond to what we know about the topology of biological networks. In fact, many biological networks, including genetic regulatory networks, seem to have a scale-free type or hierarchical output distribution (see, for example, [VDS⁺04, Alb05, CGMA07]) but not random, according to present data, as far as the *output* degree distribution is concerned¹. The *input* degree distributions seem to be close to normal or exponential instead. A scale-free distribution for the degree means that $p(k)$ follows a power-law $p(k) \sim k^{-\gamma}$, with γ usually but not always between 2 and 3. In contrast, random graphs have a Poisson degree distribution $p(k) \simeq \bar{k}^k e^{-\bar{k}}/k!$, where \bar{k} is the mean degree, or a delta distribution as in a classical fixed-degree RBN. Thus the low fixed connectivity suggested by Kauffman ($K \sim 2$) for candidate stable systems is not found in such degree-heterogeneous networks, where a wide connectivity range is observed instead. The consequences for the dynamics may be important, since in scale-free graphs there are many nodes with low degree and a small, but not vanishing, number of highly connected nodes (see, for instance, [AB02, New03]).

For the sake of completeness, we also wish to point out that the degree distribution is only one statistical aspect of a given network and the attribution of a scale-free nature to some genetic regulatory networks has been challenged [DBC05, TYD05]. Indeed, it has been recently shown that a random sample of networks with different degree distributions may give subgraphs with similar degree distributions. Conversely, networks with identical degree distributions may have different topologies [HBNP07, DBC05]. The issue is still far from being settled due to the insufficient amount of analyses. However, we believe that it doesn’t fundamentally change the nature of high-level models such as those discussed here. In particular, everybody seems to agree on the fact that the distributions are, if not scale-free, at least broad-scale, i.e they have a longer tail to the right for the output degree distribution.

The first work that we are aware of, using the scale-free topology for modeling Boolean networks dynamics is [OS02]. Oosawa and Savageau took *Escherichia coli* as a model for their scale-free nets with an average input degree \bar{k} of two. Interesting in this particular case, the model is a little too specialized as most other known networks or network fragments have higher connectivity levels. What is needed are models that span the range of observed connectivities.

Along this line, Aldana presented the first detailed analysis of a model Boolean network with scale-free topology [Ald03, AC03b]. Using the power-law exponent γ as a critical parameter instead of the mean degree, he has been able to define a phase space diagram for scale-free boolean networks, including the phase transition from ordered to chaotic dynamics, as a function of γ where, if

¹The *degree distribution function* $p(k)$ of a graph represents the probability that a randomly chosen node has degree k [New03]. For directed graphs, two distributions may be defined, one for the outgoing edges $p_{out}(k)$ and another for the incoming edges $p_{in}(k)$.

$p = 0.5$, then $\gamma_c(p) = 2.5$ is the critical value for which systems rest on the edge between order and chaos, if $\gamma_c(p) > 2.5$ and the system is in the ordered regime and if $\gamma_c(p) < 2.5$ it lies in the chaotic phase. He also made exhaustive simulations for several small values of the network size N ($N \leq 20$). The scale-free distribution was the input distribution $p_{in}(k)$ while $p_{out}(k)$ was Poissonian. We now know these distributions are actually inverted when compared to known GRNs. In our model we have thus adopted networks with a scale-free output distribution, and a Poissonian input distribution, as this seems to be at least close to the actual topologies. However, from the mathematical point of view the results in terms of different regimes as a function of γ are the same in both cases [Ald03, AC03b].

One problem with Aldana's networks was their small size since he wanted to explore the phase space exhaustively, and this can only be done for small N . However, scale-free network statistics cannot be accurate unless the network size is large enough and k ranges, which should span at least a few orders of magnitude, are suitably binned or the cumulative distribution function is used instead of $p(k)$ [DM03]. In another recent work, RBNs of various topologies and of larger size have been studied using statistical sampling and numerical simulation by Iguchi et al. [IKY07]. They used standard synchronous updating of network nodes and various graph topologies: random with Poisson distribution, exponential, and scale-free. Iguchi et. al focused on the distribution of phase space attractors and on their lengths and as such their work is closely related to the one presented here. However, most of their results concern the directed networks in which the input and output distributions are the same ($p_{in}(k) = p_{out}(k)$) and, as said, above the timing of node update is synchronous. They focused their analysis on the mean degree \bar{k} . While \bar{k} is a significant parameter for random and exponential degree-distributed graphs, it is much less meaningful for graphs having a scale-free degree distribution. For a continuous power-law distribution defined in $(0, +\infty)$ the mean becomes infinite for $\gamma \leq 2$ and the variance diverges for $\gamma \leq 3$ [Sor03]. Although \bar{k} can always be computed given a finite arbitrary degree sequence $\{k_j\}, j = 1, \dots, N$, it still loses its meaning when the distribution is such that a non-negligible number of extreme values exist, as in scale-free networks which are highly degree-heterogeneous. In this case, the average is controlled by the few largest degrees and not by the numerous small ones. These differences make it difficult to directly compare their results with ours but we shall nevertheless comment on our respective findings as their study is related in many ways to the present one.

Construction of input and output degree networks distributions

Here we present the methodology for constructing our model networks, starting with the input and output degree distributions. As said above, Kauffman's RBNs are directed graphs with connectivity K . In fact, as anticipated in the preceding section, according to present data many biological networks, including GRNs, suggest an inhomogeneous output distribution and a Poissonian or exponential input distribution [VDS⁺04, CGMA07]. Whether $p_{in}(k)$ is Poissonian or exponential both distributions have a tail that decays quickly, although the Poissonian distribution does so even faster than the exponential, and thus both have a clear scale for the degree. On the other hand, $p_{out}(k)$ is very different, with a fat tail to the right, meaning that there are some nodes in the network that influence many other nodes.

In our model we have thus adopted networks with a scale-free output distribution, where $p_{out}(k)$ is the probability a node n will have a degree k :

$$p_{out}(k) = \frac{1}{Z} k^{-\gamma}$$

where the normalization constant $Z(\gamma) = \sum_{k=1}^{k_{max}} k^{-\gamma}$ coincides with Riemann's Zeta function for $k_{max} \rightarrow \infty$. The input distribution approximates a normal function centered around \bar{k} . We call our model scale-free boolean networks (SFBNs). Figure C.1 offers a taste of what such distributions look like. Naturally, $p(k)$ being defined over the positive integers only, an approximation is necessary to define how many nodes of the network have a given input degree. Namely, in a first pass, we use the integer value $\lfloor p(k) \rfloor$ as the number of nodes that will have a degree k , for each degree $k \in \{k_{min}, k_{max}\}$. In a second pass, we use the decimal value $p(k) - \lfloor p(k) \rfloor$ as the probability that one more node will have a degree k until the degree of all nodes has been

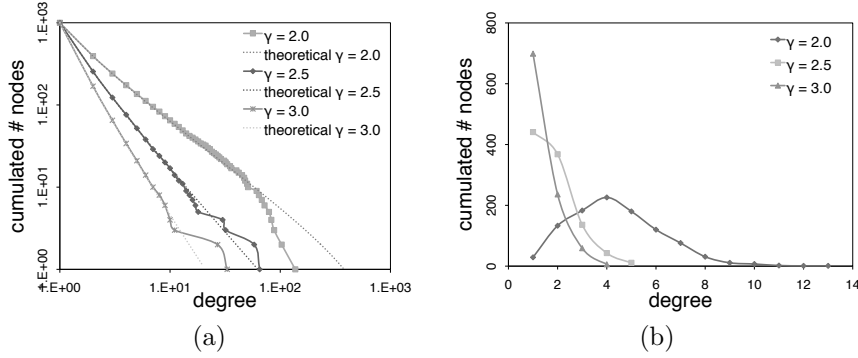


Figure C.1: Networks degrees distributions. Actual output degrees distributions on a log-log scale (a) and input degrees distributions on a lin-lin scale (b) of a sample generated networks of size $N = 1000$ and $\gamma = 2.0$, $\gamma = 2.5$ and, $\gamma = 3.0$. Distributions are discrete and finite; the continuous lines are just a guide for the eye.

specified. This non-deterministic process causes slight differences in the distributions, which is especially important around the critical regime to explore the solution space since the previously mentioned approximation leaves each scale-free distributions slightly off the power-law. Once the exact output degree distribution of a given network is known, we use the average connectivity \bar{k} to produce a matching discrete Poisson input distribution. Finally, each node i of the system is assigned an input degree k_{in}^i and an output degree k_{out}^i , and nodes are randomly connected according to these, avoiding edge repetitions.

In Table C.1 below, we show the average input and output degrees \bar{k} over 100 different networks, including their standard deviation. This is only given as an information, because, as it has been mentioned in section C.3.2 of this work and according to Aldana's model [Ald03], the regime of a SFRBN cannot be defined by its average connectivity but by the γ exponent of its output degree distribution function.

	\bar{k}_{order}	$\bar{k}_{critical}$	\bar{k}_{chaos}
$N = 100$	1.36 ± 0.06	1.82 ± 0.14	3.25 ± 0.35
$N = 150$	1.39 ± 0.05	1.81 ± 0.11	3.42 ± 0.42
$N = 200$	1.38 ± 0.04	1.81 ± 0.11	3.68 ± 0.40
$N = 500$	1.37 ± 0.02	1.84 ± 0.07	3.78 ± 0.29
$N = 750$	1.37 ± 0.02	1.84 ± 0.06	3.85 ± 0.24
$N = 1000$	1.37 ± 0.02	1.83 ± 0.05	3.98 ± 0.23

Table C.1: Networks mean degrees. Average input and output degrees \bar{k} over 100 networks including the standard deviation in all three different regimes.

Iguchi et al. [IKY07] have explored Boolean networks where both the output and the input distribution are of the scale-free type and used the average degree as an indicator of differentiation. Although an interesting metric, the average degree allows one to distinguish regimes only in Kauffman's classical RBNs with random topologies. Instead, they have used a modified *Barabási-Albert preferential attachment* model which allows one to tune the networks average degree \bar{k} . However, the \bar{k} factor has no effect on the regime, as the preferential attachment model [AB02] produces a single value of $\gamma \sim 3$ well into the chaotic regime. When dealing with the attractors cycle lengths, they have used systems where the average degree was either $\bar{k} = 2$ or $\bar{k} = 4$ which, according to our calculations, would essentially place all of their systems more or less deeply in the chaotic regime. In addition, they show examples of SFRBNs with an average degree of $\bar{k} = 1$, which does not seem possible if all nodes are connected. Thus, a direct comparison of our results with those of [IKY07] is hardly meaningful for SFRBNs.

C.3.3 Timing of Events

Standard RBNs update their state synchronously (SU). This assumption simplifies the analysis, but does not agree with results on gene activation experiments if the network has to be biologically plausible [EG06]. Rather, genes seem to be expressed in different parts of the network at different times, according to a strict sequence (see, for instance, [De02]). Thus a kind of serial, asynchronous update sequence seems to be needed. Asynchronous dynamics must nevertheless be further qualified, since there are many ways for serially updating the nodes of the network.

Two types of asynchronous updates are commonly used. In the first, a random permutation (RPU) of the nodes is drawn and the nodes are updated one at a time in that order. At the next update cycle, a fresh permutation is drawn and the cycle is repeated. In a second often used policy, the next cell to be updated is chosen at random with uniform probability and with replacement. This is a good approximation of a continuous-time Poisson process, and it will be called *Uniform Update* (UU).

Several researchers have investigated the effect of asynchronous updating on classical RBN dynamics in recent years [HB97, MT03, Ger04]. Harvey and Bossomayer studied the effect of random asynchronous updating on some statistical properties of network ensembles, such as cycle length and number of cycles, using both RPU and UU [HB97]. They found that many features that arise in synchronous RBN do not exist, or are different in non-deterministic asynchronous RBN. Thus, while point attractors do persist, there are no true cyclic attractors, only so-called “loose” ones and states can be in more than one basin of attraction. Therefore attractor lengths, which is one of the main features in RBNs, are not well defined in the asynchronous case. Also, the average number of attractors is very different from the synchronous case: even for $K = 2$ or $K = 3$, which are the values that characterize systems at the edge of chaos, there is no correspondence between the two dynamics.

Mesot and Teuscher [MT03] studied the critical behavior of asynchronous RBNs and concluded that they do not have a critical connectivity value analogous to synchronous RBNs and they behave, in general, very differently from the latter, thus confirming in another way the findings of [HB97].

Gershenson [Ger04] extended the analysis and simulation of asynchronous RBNs by introducing additional update policies in which specific groups of nodes are updated deterministically. He found that all types of networks have the same point attractors but other properties, such as the size of the attractor basins and the cyclic attractors do change.

Considering the above results and what is known experimentally about the timing of events in genetic networks we conclude, with [MT03], that neither fully synchronous nor completely random asynchronous network dynamics are suitable models. Synchronous update is implausible because events do not happen all at once, while completely random dynamics does not agree with experimental data on gene activation sequences and the model does not show stable cyclic attractors of the right size. For this reason, in the following section C.3.3 we propose a new quasi-synchronous node update scheme, which is closer to that observed in natural systems [De02, OD04].

Semi-Synchronous Update Scheme

As we have seen above, in GRNs, the expression of a gene depends on some transcription factors, whose synthesis appears to be neither fully synchronous nor instantaneous. Moreover, in some cases like the gene regulatory network controlling embryonic specification in the sea urchin [De02, OD04], the presence of an activation sequence of genes can be clearly seen. We concluded that neither fully synchronous nor completely random asynchronous network dynamics are suitable models. Thus the activation/update sequence in a RBN should be in some way related to the topology of the network, i.e. on the mutual chemical interaction structure of proteins, RNA, genes, and other molecules which is abstracted in the network.

Aiming at remaining faithful to biologically plausible timing of events without introducing unnecessary complexity into the model, we considered the influence of one node on another as biological *activating* or *repressing* factors: only when the state of the node is turned or stays *on* has this node an effect on the subsequent nodes in the activation sequence. In contrast, nodes changing their

state to or remaining *off* have no impact on nodes they are linked to, thus breaking the cascade. In other words, only the activation of an activator or a repressor will have a repercussion on the list of nodes to be updated at the next time-step. This update scheme, which has been briefly described previously in [DGT07] is called the Activated Cascade Update (ACU) . As a consequence of this

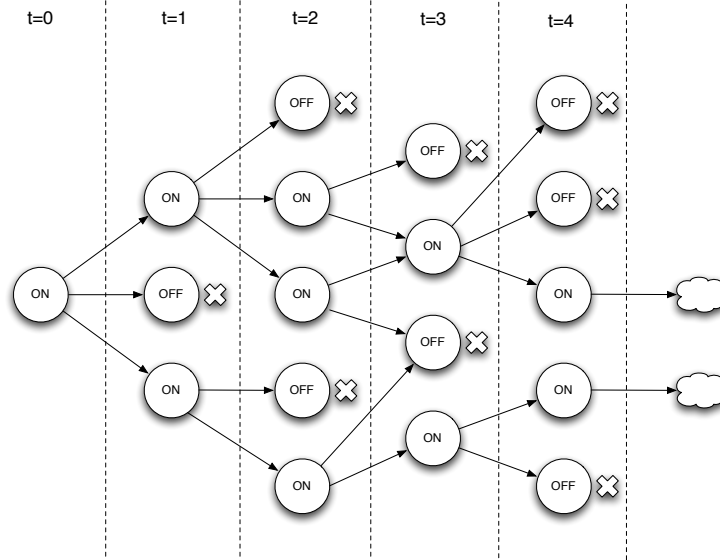


Figure C.2: A possible Activated Cascade Update sequence. At time $t = 0$ a node N_0 is chosen at random and updated according to its inputs, if the new state of N_0 is inactive, another starting node N_0 is chosen at random. At time $t = 1$ all the nodes receiving an input from N_0 are updated according to their own inputs, those becoming or remaining active (state *on*) decide which node will be updated at the next time step. The cascade continues according to this scheme.

novel update procedure, the definition of point or cyclic attractors changes slightly, because the state of a network at any give time t is, from now on, not only determined by the individual state $s_i^t \in \{on, off\}$ of each node but also by the list l^{t+1} of nodes to be updated at the next time step. The concept of loose attractor has, in this context, no relevance.

C.4 Methodology and Simulations

In this work we investigate the effect of the new ACU update scheme presented in Section C.3.3 vs. the previous SU on SFBNs for a set of γ exponents of the scale-free distribution $\gamma \in \{2.0, 2.5, 3.0\}$. The results will be compared to classical RBNs and all three sizes mentioned above will be studied. In order to explore their behavior in the three different regimes, we propose to vary $\bar{k} \in \{1.8, 2.0, 2.2\}$, thus keeping the probability p of the node update functions to $p = 0.5$. In an effort to probe the network scaling properties, we have simulated ensembles of graphs with $N \in \{100, 150, 200\}$ which, although still comparatively small, is closer to the observed GRNs sizes and still computationally feasible.

For each combination of topology, update and size, we produce 50 networks. To each network, we associate 20 randomly generated sets of Boolean update functions. A network-function pair is called a realization. Subsequently, for each realization we create 500 different initial configurations (ICs) with equal probability for each gene to be expressed or not. Starting from each IC, we let each realization run over a number of initial steps depending on the size N of the network (10000 for $N = 100$, 20000 for $N = 150$, and 30000 for $N = 200$). This allows the system to possibly stabilize after a transient period, reaching the basin of an attractor. After this primary period, we determine over another 1'000 time steps if the system has relaxed to an attractor. If so, we

define the length of that attractor as the minimum number of steps necessary to cycle through the attractor’s configuration. In other words, we run $50 \text{ networks} \times 20 \text{ update functions} \times 500 \text{ ICs} = 5 \times 10^5$ simulations for all combination of 3 sizes, 3 regimes and 2 updates for a total of 9×10^6 simulations. Very often in this work, we will omit to show figures of all three different sizes as this parameter does not always have an impact on the results, that are in turn very similar for all sizes. Nevertheless, all sizes and cases have been thoroughly simulated and studied.

C.5 Finding Attractors

During the simulations, we have analyzed for each IC of each realization whether the system has relaxed to a single state (point attractor) or cycled through the configurations of a periodic attractor. According to Kauffman’s estimate [Kau93], the median lengths of attractors $l \propto \sqrt{N}$ or linear at most for \bar{k} critical. For \bar{k} well into the chaotic regime, the median length grows exponentially with N . Biologically speaking, very long attractors are unlikely to have any meaning due to the actual gene expression time which is in the order of seconds to minutes. Therefore we investigate in depth only attractors with lengths ranging from 1 to 100 states. Admittedly, the maximum length is arbitrary but remember that, according to Kauffmann, we are mostly interested in attractors that are short and stable in the “critical” regime (or “edge of chaos”). In natural systems, point and periodic attractors may have different significations. As an example periodic attractors can be interpreted as a model of the genetic regulatory system during the cell cycle, whereas point attractors can refer to the end of the differentiation cycle of a stem cell. Although it has been shown that point attractors may play a fundamental role outside the stem cell context, as in the works of Albert *et al.* [AO03] and more recently of Álvarez-Buylla and coauthors [ABCA⁺08], we will often present simulation results and statistics both with and without point attractors. The reason for this is that in some instances under ACU, the sheer number of point attractors tend to bias the statistics and to make the results more difficult to interpret (see Fig. C.4).

C.5.1 Number of Attractors

In Fig. C.3 we show the frequencies at which networks of size (a) $N = 100$ and (b) $N = 200$ find attractors of any length. Since the simulations for networks with $N = 150$ nodes behave similarly to larger and smaller ones, we do not show them here. Fig. C.3 shows that almost all instances

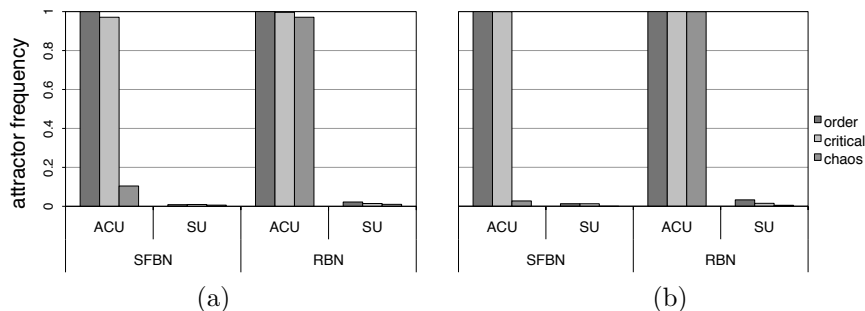


Figure C.3: Number of attractors of any length. Comparing the frequency at which simulations have found an attractor of any length for realizations with (a) $N = 100$ and (b) $N = 200$ genes. We compare all network topologies and update schemes.

under ACU we find an attractor, except for scale-free systems in a chaotic regime, which tend to produce 10 to 50 times less attractors. On the contrary, GRNs under SU struggle to relax to an attractor. In both RBNs and GBNs, we observe that the number of attractors does not seem to be impacted by the scaling.

Frequencies and length concerning simulations of shorter and more biologically plausible attractors are shown below in Fig. C.4 and in Fig. C.5 respectively. On the right-hand sides, point attractors

have been removed from statistics. When comparing Figs. C.3(a) with C.4(a) and C.3(b) with

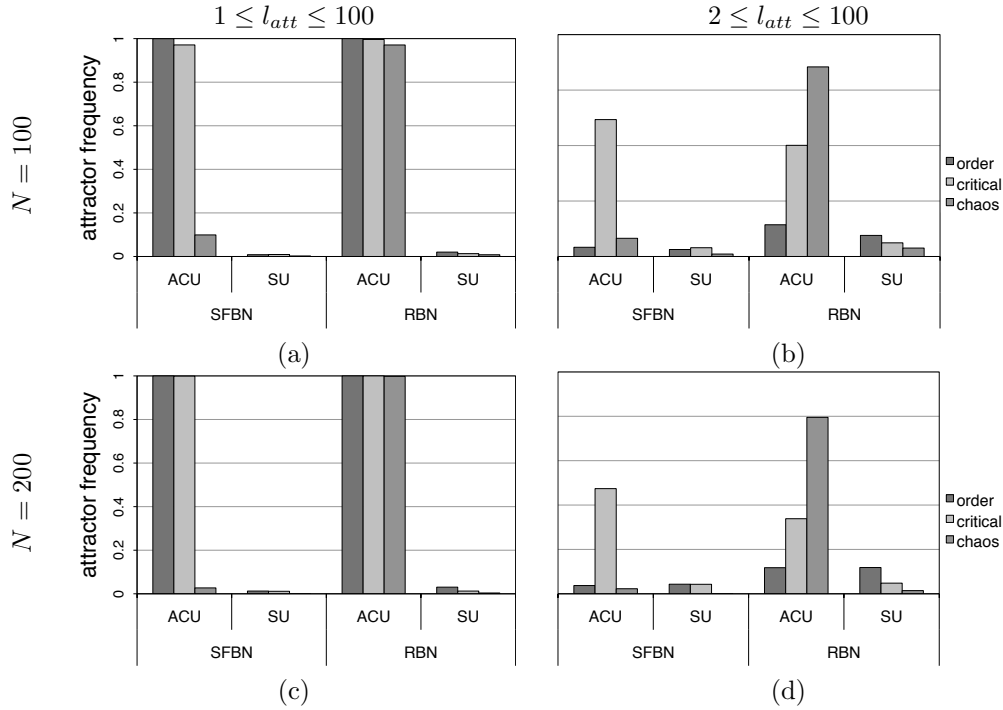


Figure C.4: Number of attractors of biologically plausible lengths. Comparing the frequency at which simulations have found attractors of length (a)(c) between 1 and 100 and (b)(d) between 2 and 100. We consider networks of size (a)(b) $N = 100$ and (c)(d) $N = 200$.

C.4(c) respectively, there is virtually no difference, as over 95% of the attractors are in fact below a length of 100 states. As for attractors of length between 2 and 100 in Fig. C.4(b) and (d), we see that ACU systems, whether scale-free or random, find more attractors than those under SU. We note that SFBNs in a critical regime under ACU have a peak in finding attractors, compared with other regimes, which are exactly the attractors we are interested in. In RBNs and GBNs, we observe that the number of attractors does not seem to be impacted by the scaling either. Using ACU almost every IC of every realization leads to an attractor, no matter what the regime is. On the contrary, under SU the overall number of attractors tends to decrease as the system goes from order to chaos.

C.5.2 Variety of the Attractors

Table C.2 shows how many times on average the same attractor has been found for each update scheme, regime and topology over the 500 ICs the system has been submitted to. We divided the results for attractors including and excluding point attractors (PA). We can summarize in Table C.2 a few observations as follows: the topology type does not seem to have a drastic effect on how often the system relaxes to the same attractor. On the other hand, the update scheme affects the total number of times the same attractor is found, and so does the regime, but in a much milder manner. In fact, we can see that SU tends to find much more often the same attractor than ACU does. In addition, we see in Figures C.3 and C.4 that this SU also tends to find many fewer attractors overall. Alternatively, systems under ACU find a greater number of different attractors, only in the chaotic regime, where the overall number of distinct attractors is already very small compared with the other ones. We witness an increase in the average number of the times the same attractor is found. Note that in the chaotic regime for systems where $N = 200$ under SU, the low repetition value is due to the fact that very few attractors are found.

			N=100		N=200	
			w PA	w/o PA	w PA	w/o PA
SFBN	ACU	order	1.01	1	1	1
		critical	1.03	1.13	1.01	1.01
		chaos	4.42	10.44	2.16	1.68
	SU	order	121.16	73.37	46.55	78.77
		critical	104.26	95.82	60.54	65.46
		chaos	5.08	43.60	1	1
RBN	ACU	order	1.01	1	1	1
		critical	1.01	1.01	1	1
		chaos	1.03	1.07	1.01	1.01
	SU	order	44.90	39.39	22.24	26.33
		critical	58.01	53.26	30.34	33.93
		chaos	61.01	58.75	24.69	30.04

Table C.2: Attractors diversity. The average time each attractor has been found over 500 ICs. Cases with (w PA) and without (w/o PA) point attractors are segregated. In this case, $N = 100$ and 200, and the attractors length is limited to a maximum of 100 states.

C.5.3 Length of the Attractors

Fig. C.5 shows statistics on the length of attractors. We exclude point attractors for figures on the right-hand side (figures (b) and (d)). The bar at the center of the box is the median of the attractors lengths, the upper and lower box delimiters are the third and first quartile respectively. The whiskers show extreme minimal and maximal values. Results are shown only for the case where the networks size $N = 200$, as results for smaller sizes are very similar. Once more we see that scaling does not have a significant impact on the length of the attractors that are found by the systems. It is mostly the regime the system evolves in and, in a lesser manner, the update scheme that defines the attractors average length. We note in Fig. C.5(a) and (c) that, although under-represented, attractors under SU seem to be the longest, especially in the chaotic regime. When focusing on the more interesting part of the attractors population in Fig. C.5(b) and (d), we see that the lengths remain comparable, though slightly shorter when considering systems under ACU. We also know from the section C.5.1 above that those attractors are much more frequent in systems under ACU. A global conclusion concerning the attractors distribution is that the update model has a prominent effect on the number and length of attractors over the networks topologies. Fig. C.6 shows the distribution of the number of attractors according to their length on a *log-log* scale. Interestingly, the distribution of attractors lengths for SFBNs shows a *long-tail* for both updates, which is especially marked for systems in the chaotic regime. But this distribution does show comparable tendencies on SFBNs under any update, whereas for RBNs, the tail under ACU is much less pronounced than it is under SU. So we see that now, regime has a greater influence on attractors length for SFBNs and not the update scheme. It is the opposite for RBNs, where the timing of update has a greater impact. All cases compare favorably with Aldana's work [Ald03] where he clearly shows that SFBN systems, although he had the input and the output distribution swapped, exhibit a long tailed distribution of the attractors lengths in the chaotic regime. This phenomenon is much less pronounced in the ordered and critical regime. Although we observe a difference in the case of RBNs, the different regimes are difficult to tell apart. Nevertheless, the distributions also show a power-law-like curve for all regimes, with a tail longer than that of SFBNs in the ordered and critical regime. This second observation is in line with Iguchi et al. [IKY07] where, in the case of smaller RBNs, both in the critical and chaotic regime, attractors lengths distributions show a long tail. We also notice that ACU has the unexpected effect to help tell apart regimes, as distribution is much easier to distinguish than under SU.

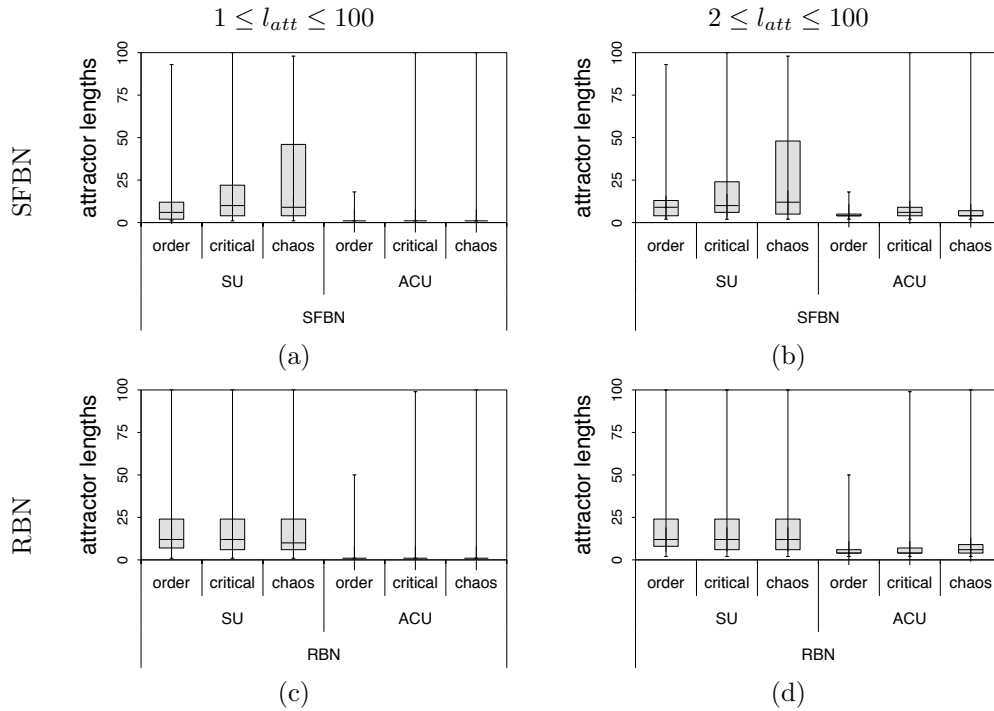


Figure C.5: Attractors lengths. Comparing the length of the attractors found by simulations. In the left-hand side column (a)(c) we show statistics of attractors of size up to 100 states and the right hand-one (b)(d) also excludes point attractors. Horizontally, the upper row (a)(b) corresponds to SFBNs and the lower row (c)(d) to RBNs. All systems have $N = 200$ nodes.

C.5.4 Scaling

Modern high throughput technologies for genetic analysis have tremendously contributed to the unveiling of ever bigger parts of GRNs in living organisms. Present sub-networks sizes range from a few tens to a few hundreds of genes. In the section above, we thoroughly investigated the attractor's dynamics of systems of sizes ranging from 100 to 200 nodes, and have noticed the size of the system mainly affects the number of attractors that are found. This fact was expected as the state space grows with the number of nodes N as 2^N , making it harder for the system to relax in a cycle. For other properties such as the length distribution or mean length, although slightly different, the general tendencies are not impacted by the scaling.

In order to study the effect of scaling on Boolean systems and its effect on both different topologies and both updates, we have extended the above analysis to networks of size $N = 25, 50, 75, 100, 125, 150, 175, 200$. Due to extreme computational resources necessary, we have unfortunately not been able to increase N to greater sizes and obtain sufficiently reliable data. Indeed, as the number of node grows, the transient period before the system reaches an attractor and length of the attractors themselves increases dramatically, especially in the chaotic phase. Aldana [Ald03] shows the increase in the transient time and also shows trends on the expected length of the attractors as N grows.

Fig. C.7 shows the trends followed by the attractors lengths as the size of the systems grows for both topologies, updates and all three regimes. The size of the attractors for SFBN systems under both update strategies, scales as expected form Aldana's work [Ald03]. He exhaustively studied SFBNs under SU of sizes $N \in \{8, \dots, 20\}$. We witness, both for SU and ACU, a similar and expected trend, where only the length of attractors found by systems in the chaotic phase increase significantly with N . The mean attractor length of systems under ACU is much shorter than that of systems under SU, which is in line with the thorough analysis conducted in Section

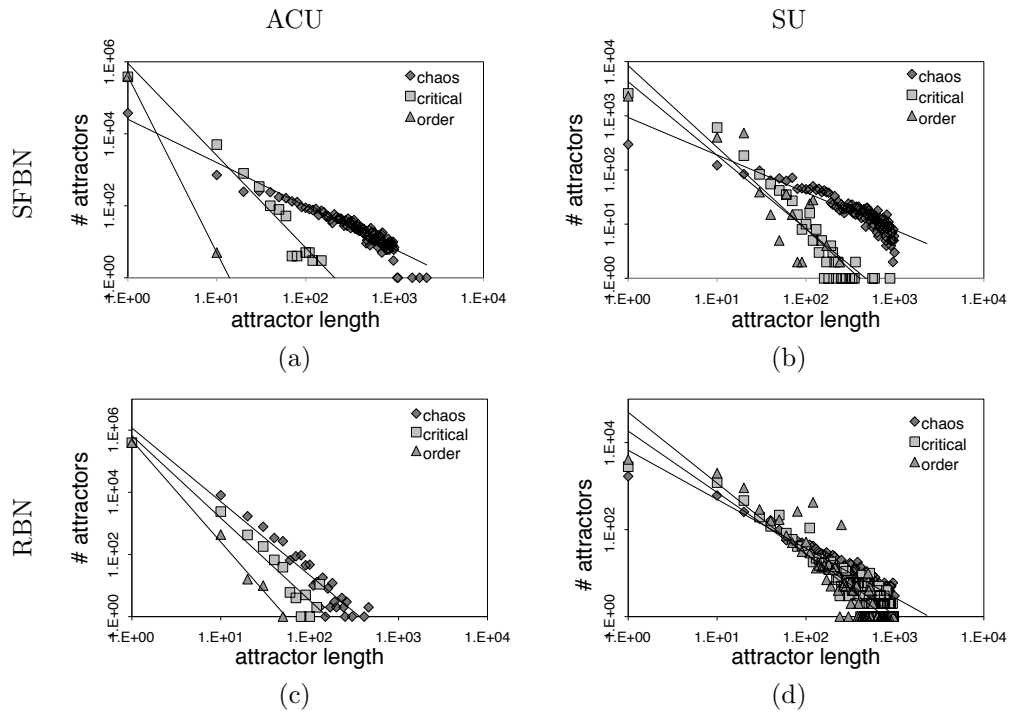


Figure C.6: Attractors lengths distribution. Distribution of the attractors' lengths between 1 and 100 states. The left-hand side column represents systems under (a)(c) Activated Cascade Update and the right-hand side on (b)(d) Synchronous Update. Figures in the upper row (a)(c) show results for SFBN, the lower one (b)(d) for RBN networks of size $N=100$. Note that the vertical axis in the four figures have different scales. The continuous lines are power regressions to be used as a guide for the eye.

C.5.3. In the case of classical RBNs, the differences in size between the regimes, although existing, is much less pronounced. The range average size is yet again much greater with SU. Under both updates, lengths for ordered and critical regimes remain relatively close whereas for the chaotic regime, the difference with the other regimes augments significantly. We have unfortunately not been able to compare the number of attractors to Aldana's work because we are only sampling much bigger systems, up to ten times larger, that cannot be exhaustively analysed in a reasonable amount of time. Nevertheless, this comforts us in the idea that our model, while in our eyes is more realistic, still it has behaviors that are in accordance with our predecessors validated work. Iguchi et al. [IKY07] have conducted similar experiment on a limited sample of scale-free input and output distribution networks of very large size under SU. Though their results seem in agreement with our own findings, their model is too different to draw direct parallels.

C.6 Fault Tolerance of Random Boolean Networks

Failures in systems can occur in various ways, and the probability of some kind of error increases dramatically with the complexity of the systems. They can range from a one-time wrong output to a complete breakdown and can be system-related or due to external factors. Living organisms are robust to a great variety of genetic changes, and since RBNs are simple models of the dynamics of biological interactions, it is interesting and legitimate to ask questions about their fault tolerance aspects.

Kauffman [Kau00] defines one type of perturbation to RBNs as "gene damage", that is the transient reversal of a single gene in the network. These temporary changes in the expression of a gene are

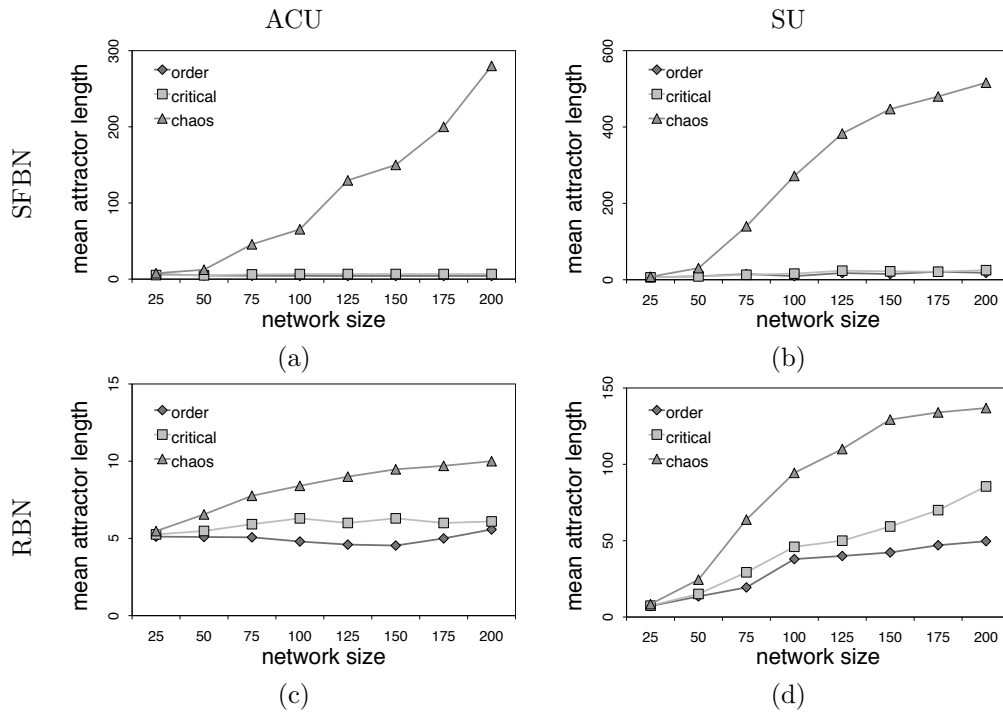


Figure C.7: Attractors average cycle length with respect to the network size. Scaling of the average length of attractors compared to the networks size N for (a)(b) SFBNs and (c)(d) RBNs under (a)(c) ACU and (b)(d) SU. Note that the vertical axis in the four figures have different scales. Continuous lines are only added as a guide for the eye.

extremely common in the normal development of an organism. The effect of a single hormone can transiently modify the activity of a gene, resulting in a growing cascade of alternations in the expression of genes influencing each other. This is believed to be at the origin of the cell differentiation process and guides the development.

The effect of a gene damage can be measured by the size of the avalanche resulting from that single gene changing its behavior from active to inactive or vice-versa. The size of an avalanche is defined as the number of genes that have changed their own behavior at least once after the perturbation happened. Naturally, this change of behavior is compared to an unperturbed version of the system that would be running in parallel. The size of the avalanche is directly related to the regime in which the RBN is; in the ordered regime, the cascades tend to be significantly smaller than in the chaotic regime. In real cells, where the regime is believed to lie on the edge of chaos, the cascades tend to be small also. Moreover, the distribution of the avalanche sizes in the ordered regime follows a power-law curve [Kau00], with many small and few large avalanches. In the chaotic regime, in addition to the power-law distribution, 30-50 percent of the avalanches are huge. The distribution of avalanche sizes of RBNs in the ordered regime roughly fits the expectations of biologists, where most of the genes, if perturbed, are only capable of initiating a very small avalanche, if any. Fewer genes could cause bigger cascades, and only a handful can unleash massive ones. Perturbing an arbitrary gene is reasonable in RBNs where all genes have the same average number of interactions. In scale-free nets however, this is no longer true due to the presence of a high degree inhomogeneity. Even for values of γ around 3.5 there will be nodes that have many more output connections than the average value. A transient perturbation of a gene that has few interactions will have moderate or no effect, while perturbing a highly connected node will have larger consequences. Several studies dealing with various kinds of system perturbation have been recently published. Aldana's approach [Ald03] is similar to the one taken here except that he deals with small scale-free networks in which N , the number of nodes, is

19. Ribeiro and Kauffman [RK07] exhaustively studied the state space of small ($N < 20$) RBNs under probabilistic errors in gene state searching for ergodic sets, i.e. sets of states such that once the system is in one of them, it cannot leave it subject to internal noise. They find that if noise may affect all nodes of an attractor then multiple ergodic sets are unlikely. However, when noise is limited, multiple ergodic sets do exist which means that attractors are stable. Serra et al [SVGK07] present a study of the distribution of avalanches in unperturbed RBNs and in RBNs in which one gene has been “knocked-out”, i.e. a state 0 has been permanently changed to 1. They show that the standard model readily explains the distribution of the resulting avalanches. They also examined the influence of a scale-free topology for the outgoing links on the system. Aldana et al. [ABKR07] examine the effect of more complex and biologically plausible perturbations of the attractor landscape of both standard RBNs and scale-free RBNs. Genes undergo duplication and mutation which cause topological changes that in general maintain the original attractors and may create new ones. Near the critical regime robustness and evolvability are found to be maximum.

C.6.1 The Effect of Perturbation

In this work we have submitted all systems that have reached biologically plausible attractors to “gene damage”, the simplest failure amongst those previously described. That is, when the system is cycling through the configurations of the attractor, the whole system is duplicated. The original will continue unperturbed. On the other hand, a node of the copy is chosen at random and will give the opposite output value a single time step. This usually knocks the system out of the course of its attractor. Now we let both systems evolve over time and record at each time step how many more nodes have a different value in the copy compared to the original. This value usually reaches a maximum that represents the number of nodes that have ever had a behavior different than those of the original. This number is the size of the *avalanche*. There are only three possible scenarios for the copy: it will return to the same attractor as the original, it will reach a different attractor or diverge and reach no attractor within the maximum number of configurations allowed (1000). Each system in an attractor is copied 10 times, and each copy will have a different avalanche starting point. We record separately these informations in order to compare the re-convergence capabilities of systems in each regime, with different topologies and update schemes.

Fig. C.8 shows the frequency at which systems that have already converged to an attractor re-converge. We show separately whether systems re-converge to any attractor or to the same one as before the perturbation. In particular, Fig. C.8 depicts results for attractors before perturbation (original attractors) of sizes between 2 and 100. We show systems that used networks of size $N = 200$. Results for smaller systems are comparable and are not shown in this work. Re-

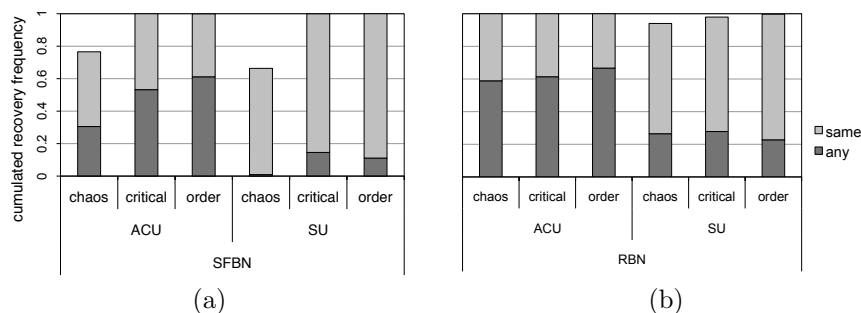


Figure C.8: Re-convergence frequency. Frequency at which perturbed systems re-converge to either the same attractor (light grey) or another one (dark grey). Left-hand (a) side figure shows results for SFBNs and right-hand (b) side figure shows results for RBNs. All systems have $N = 200$ nodes. We purposefully omit point attractors.

convergence seems to mostly depend on the regime the system evolves in, rather than its degree distributions, update scheme, or size. On Fig. C.8 we note that only networks in the chaotic

regime do not re-converge to an attractor in every case. It also seems that ACU performs a little better at helping systems to find a stable state. However, this tendency seems inverted when taking into account only cases where the same attractors are found. In this case, under ACU, the same attractor as the original one is found about half of the time. Under SU, the same one is found about 75% of the time. This could be explained by the fact that the number of attractor lying in the state spaces of systems under ACU is much greater.

As expected when dealing with random failure, the information traveling through a structure with regular output distribution is more vulnerable to faults compared to structures with *hubs* and *leaves*. This fact is well known in various examples such as computer networks which are very resistant to random failure as long as they are failures and not targeted attacks on highly interconnected nodes. Especially under SU, SFBNs tend to re-converge to the same attractor more than RBNs, although overall, both topologies perform well. The chaotic case will be explained below in details. Under ACU, critical and ordered SFBNs systems are again performing as able as or better than their counter parts in RBNs, recovering as often to any attractor but more often to the same as the original one. The counter-performance of chaotic systems, especially SFBNs, can be explained by the “spike of huge avalanches” described by Kauffman [Kau00] and visible in Fig. C.9. Indeed, SFBN systems and, in a lesser manner, RBN under SU have a surge of very long avalanches when in the chaotic regime. This characteristic explains why these systems are not as well able to re-converge to an attractor, let alone the same one.

Fig. C.9 shows the distribution of the avalanches’ size. Again we distinguish networks that have re-converged at all in Fig. C.9(a) and those that have re-converged to the original attractor Fig. C.9(b). For readability reasons and, since results are very similar, we show only results for systems of size $N = 100$ and $N = 200$. As mentioned in Section C.6, the size of the avalanche varies

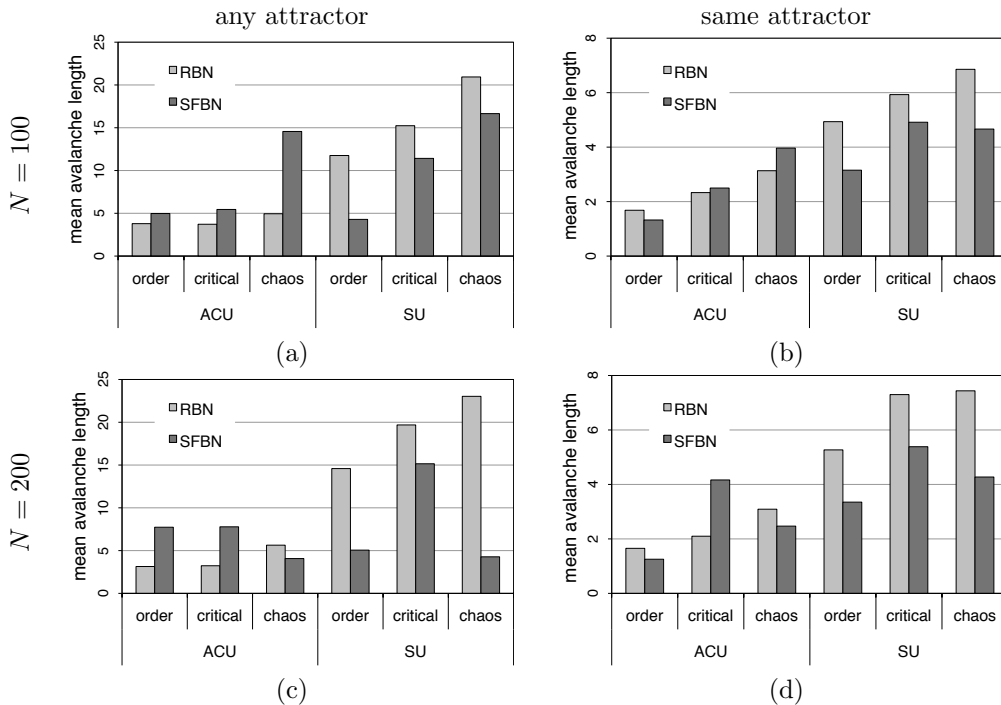


Figure C.9: Mean avalanche lengths. Average avalanches length of cases where systems re-converge to (a)(c) any attractor and (b)(d) the same attractor.

mainly due to the regime. Smaller systems with $N = 100$ react as expected, with the size of their avalanches increasing as the systems grows chaotic. However, this does not seem to always be the case, and this relationship between avalanche size and regime is changed in bigger networks. Under ACU networks where $N = 150$ or $N = 200$, it is the systems that evolve in the critical regime that

clearly show the longest avalanches. This is true for ACU only, SU systems still corroborate Kauffman's conjecture. Although in the case where systems return to the original attractor, avalanche sizes are much smaller, the tendencies observed in the more general case stand. This time we observe an obvious impact of the networks size on the systems dynamics. Further investigations are necessary as to define why larger systems in critical regime under ACU are more impacted by perturbations.

Fig. C.10 shows the distribution of the avalanches' sizes for different systems. Although values are discrete, we used continuous lines as a guide for the eye. In Fig. C.10, we see that tendencies

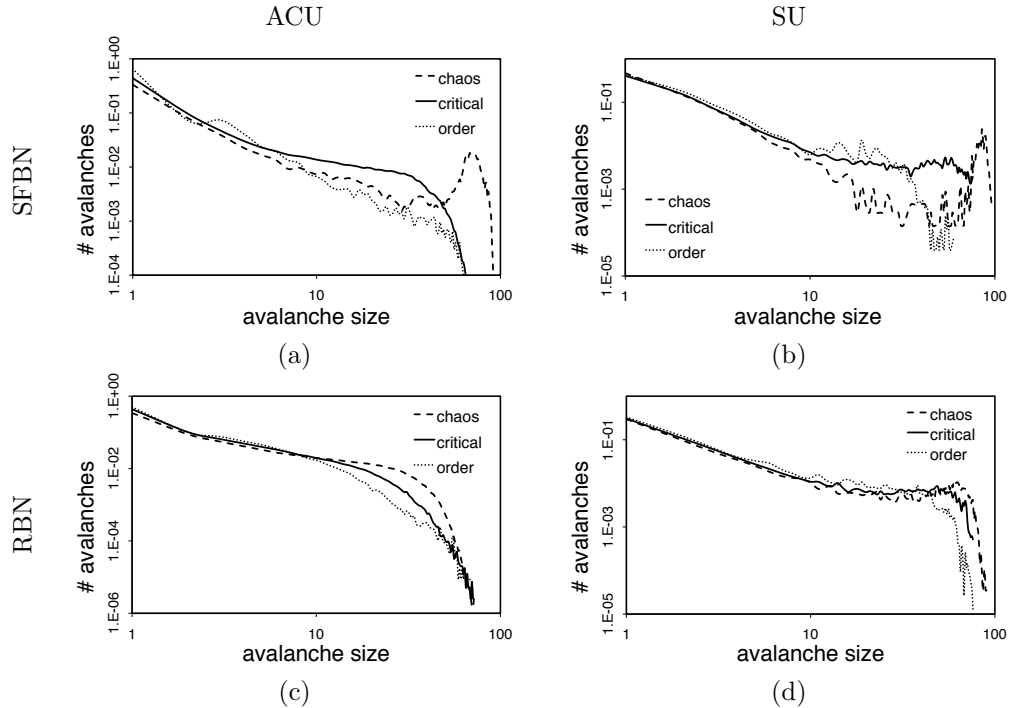


Figure C.10: Avalanche length distributions. Distribution of the avalanche lengths for all three regimes and $N = 100$. Upper row (a)(b): For SFBNs. Lower row (c)(d): For RBNs. Left-hand side column (a)(c): Systems under ACU. Right-hand side column (b)(d): Systems under SU.

are the same and are anticipated from Kauffman's work [Kau00]. SFBNs under both (a) SU and (b) ACU exhibit a steady long tailed decrease in the number of avalanches as their length grows for ordered and critical regime, and there is an increase for long avalanches in the case of chaotic systems. This tendency is the same for synchronous RBNs in (d). Interestingly, this does not seem to apply to RBNs under ACU, where no increment is to be noted.

Lastly, Fig. C.11 illustrates the average output degree of the node that represents the damaged gene. For clarity reasons, we show results only for bigger systems as they are similar when networks are scaled down. Although predictable, we clearly see the effect of the hubs in SFBNs, where failing nodes in systems that do not re-converge had a much higher output degree on average than those of systems that did recover. Another interesting observation, is that there seems to be a direct relationship between the degree of the wrongful node and the regime, the more ordered the system, the higher the degree to allow the system to recover. This difference is toned down in ACU systems. Naturally this does not hold for classical RBNs, where all nodes have comparable output degrees.

As a general conclusion on gene-damage failures of Boolean Networks, we can highlight the prominent effect of the topology on distribution of the lengths of the avalanches and its ability to re-converge to an attractor over the networks update and regime.

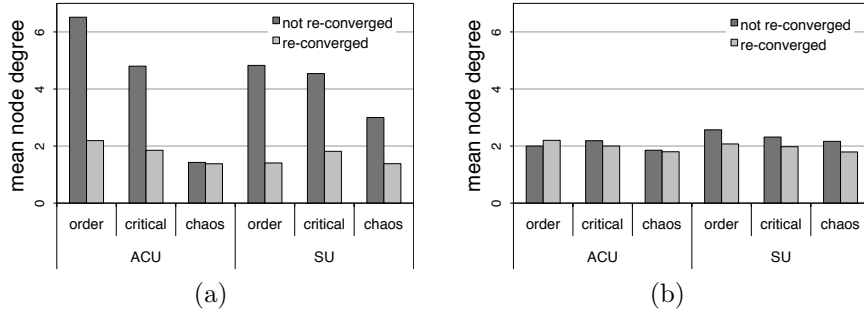


Figure C.11: Faulty nodes average degree. Average degree of the nodes that have failed for both re-converged and not re-converged avalanches. RBNs are shown on the right hand side (b) and SFBNs left one (a). Size $N = 200$.

C.6.2 Derrida Plots

In this section we compare Derrida plots of our models with those of Kauffman [Kau00] and Iguchi et al. [IKY07]. These representations are meant to illustrate a convergence versus a divergence in state space that can in turn help characterize the different regimes. These plots show the average Hamming distance $^2 H(t)$ between any two states s_a and s_b and the Hamming distance $H(t + 1)$ of their respective consecutive state s'_a and s'_b at the next time step. Derrida plots of systems in the chaotic regime will remain above the main diagonal $H(t) = H(t + 1)$ longer, crossing the main diagonal earlier and remaining closer to it as the systems near the critical regime. Systems in the critical regime remain on the main diagonal before diverging beneath it. Ordered systems remain under the main diagonal at all times. These results are already known for RBNs under SU and, to some extent as the regimes are not defined explicitly by Iguchi et al. [IKY07], for SFBNs under SU.

Figure C.12 shows the Derrida plots for systems under ACU on the right and SU on the left. This plot concerns networks of size $N = 100$. The system under SU on the right-hand side has a fairly

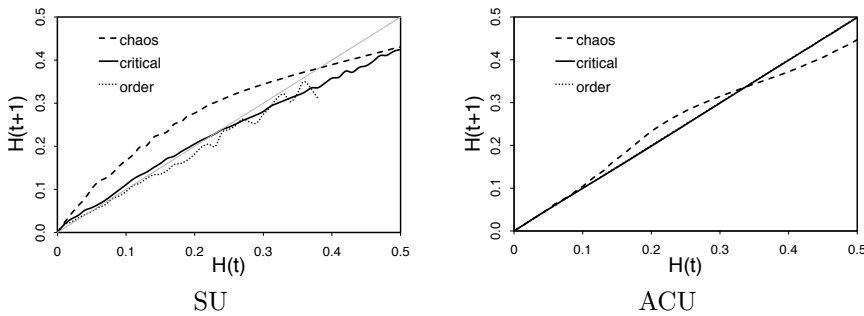


Figure C.12: Derrida plot of the Hamming Distance H at time t vs. $t+1$. The right-hand side figure represent systems under Activated Cascade Update and the left-hand side one under Synchronous Update. All systems are SFBNs of size $N = 100$.

typical behavior, where the chaotic systems remain clearly above the main diagonal, critical ones remain close to and then diverge below the main diagonal. The ordered curve, although clearly remaining below the main diagonal, is somewhat irregular. This is probably due to the fact that the number of attractors is the lowest of all in the ordered SFBN systems under SU, thus making the curve less smooth. In the case of SFBNs under ACU, the chaotic curve shows the expected behavior, though it remain closer to and crosses the main diagonal earlier than in the SU case.

²The normalized number of positions that are not identical when comparing two (binary) strings.

It can still be considered a reliable indication that the system is indeed in a chaotic regime. In the ordered and critical regimes, the curves are literally on the main diagonal all the way through, and show no sign of convergence or divergence whatsoever. So in this case, the update method has a major impact on the Derrida plots, making the ordered and critical systems impossible to distinguish under ACU.

C.7 Conclusions and Future Work

Although a long way from a fully functional model of GRNs, we are moving closer to one by aggregating modern findings obtained with recent high throughput techniques. These refinements to the original RBN model by Kauffman and the subsequent ones by Aldana help us understand some key details of the complex interactions that are taking place between the different components and the role that the topological structure plays in the dynamics. In this paper, we have made some progress towards an understanding of what structural and dynamical properties make GRNs highly stable and adaptable to mutation, yet resistant to perturbation.

This work suggests one structural property, namely the scale-free output distribution, and a dynamical one, the semi-synchronous updating, to try to improve the standard RBN model and to account in an abstract way for recent findings in system-level biology. We have used computer simulations to reflect the impact of these changes on original RBN models. Results are encouraging, as our SFBNs model shows comparable or better performance than the original one with more attractors and smaller avalanches. This leads us to believe that the models are pointing in the right direction. Nevertheless, from the results of this analysis, we also see that neither model is the absolute optimum in this problem. Indeed, if we focus on maximizing the number of attractors, the prominent effect is that of the update, with ACU combined with original RBNs achieving the best results in finding the most attractors with a biologically relevant cycle length. On the other hand, when considering maximizing the fault tolerance, we witness the highest resilience with SFBNs under SU, that achieve the highest rate of re-converging to the same attractor as observed originally. This demonstrates that no combination is optimal on all problems and that compromise is necessary if we are looking to build a model that will perform well in a realistic situation.

In the future, we intend to expand the range of analysis conducted on perturbed systems, in the hope of shedding some light on GRNs. Also, we would like to explore different degree distribution types and combinations, including the use of some actual GRNs as high-throughput molecular genetics methods make real-life data available like never before.

Acknowledgements

The authors thank F. Di Cunto and P. Provero of the University of Torino (Italy) for the useful discussions and suggestions on biological regulatory networks and C. Damiani, M. Villani and R. Serra for their insightful suggestions about RBN models. M. Tomassini and Ch. Darabos gratefully acknowledge financial support by the Swiss National Science Foundation under contract 200021-107419/1. M. Giacobini acknowledge funding (60% grant) by the Ministero dell'Università e della Ricerca Scientifica e Tecnologica.

Article D

Additive functions in Boolean networks: where synchronicity meets topology-driven update

A case study on three real-life regulatory networks

Authors: Ch. Darabos, M. Tomassini, F. Di Cunto,
P. Provero, M. Giacobini
Submitted to: BMC Systems Biology
Date: January 29, 2010

Abstract

Background: The gene-on-gene regulations are key components of living organisms. Dynamical abstract models of genetic regulatory networks, such as Kauffman's *Random Boolean Networks*, may explain the genome's evolvability and resilience to faults by the structural topology of the graph formed by genes, as vertices, and regulatory interactions, as edges. We use two real-life instances of sub-networks as support for Boolean network models, and propose a novel threshold-based dynamic update function. In this new update function the inductive or repressive effect of each edge is taken into consideration and makes it more biologically plausible than the original random update function.

Results: In order to investigate the dynamical behavior of this new model, we visualized the phase transition between order and chaos into the critical regime using Derrida plots. We also proposed a new measure, the criticality distance, that allows to discriminate between different regimes in a quantitative way. Moreover, the state spaces of the two real-life GRNs is portrayed using RBN-specific statistical measurements. Simulation results on two real-life genetic regulatory networks, the yeast cell-cycle and the mouse embryonic stem cell, show that there exist parameter settings in both update functions that allow the systems to operate in the critical region, and that these values are comparable in the two case studies. Finally, we use a third real-life regulatory network that comes with actual update functions to validate the results obtained.

Conclusions: Our new Boolean models for genetic regulatory networks includes a great deal of experimentally derived biological information, which we believe makes it more biologically relevant and potentially useful to guide experimental research. The new update function confers additional realism to the model, while actually reducing the solution space, thus making it easier to investigate.

D.1 Introduction

Genes are the central pillar of biological evolution, and therefore of life as we know it. The various genome projects provided us with lists of genes which for many organisms, including humans, are thought to be fairly complete, at least for protein-coding genes. Much less is known about genes as part of a dynamical biological system, that is about the complex regulatory interactions among genes allowing the genome to shape the organism and its interaction with the environment.

These interactions can be represented as genetic regulatory networks (GRNs) representing the regulatory effects of a gene on the others. However interactions within these networks are very subtle, intricate, and ill understood. While GRN sections of a few tens to a few hundreds of genes are known in detail for several organisms, the quality of the data drops dramatically as the network size grows.

Nevertheless, GRNs are the next big thing, and are at the center of tremendous research efforts from the biological community. The quantity and quality of results in the field, thanks to modern high-throughput molecular genetics methods, are bound to follow the same exponential trend as the gene sequencing did in its time. In the meantime, however, it is possible, and useful, to abstract many details of the individual GRNs in the cell and focus on the system-level properties of the whole network dynamics. This Complex Systems Biology approach, although not immediately applicable to any given particular case, still provide interesting general insight.

An early dynamical model for GRNs was proposed in the late 60's by Kauffman [Kau69] and is known as Random Boolean Networks (RBNs). This abstraction is very attractive due to its simplicity, yet unveils interesting dynamical phenomena about how the network structure and the gene-gene interactions are at the center of the resilience to transcriptional errors, and yet evolvability of GRNs. The dynamics of RBNs can be discriminated in two regimes: the *ordered* regime, where the system tends towards less changes in the gene activations, thus more stability to transient faults, and the *chaotic* regime, where gene activation changes frequently, thus less stability and more evolvability. It has been suggested, that biological cells operate at the border between order and chaos, a regime called *critical* or *edge of chaos* [Lan90, BTW88, Kau93]. Systems in this regime are capable of exceptional behavior: they show robustness to small perturbations, and yet remain flexible enough to integrate external signals allowing the system to adapt to new conditions in its environment. This is true for both organic [LT03, Kitnt] and non-organic systems [SDMnt] and it is a signature feature of Complex Systems in general. A way to visualize this phase transition into the critical regime makes use of *Derrida plots* [DP86], which provide a visual way of classifying RBN systems according to their dynamical behavior.

In a previous works [DGT07, DTG09], we highlighted the main weaknesses of Kauffman's original assumptions, that is, the random topology of the networks and the total synchronicity of events. Then, we proposed a novel update scheme based on gene activation and we used scale-free topologies as proposed by Aldana [Ald03]. That new model, although more faithful to present knowledge about biological networks, still suffered from one of the flaws of the original one, that is, that gene-gene interactions are drawn at random according to a scale-free degree distribution.

In the present work, we remedy this situation by using three subnetworks of real-life GRNs. In the cases studied here the interactions between the genes are known with a good level of confidence. In a related work, Ballenza *et al.* [BABC⁺08] use microarray data and canalizing functions to infer the nature of the gene regulatory network interactions in several organisms. In this work, we take advantage of extra information contained in real-life GRNs, that is the actual activating or repressing effect of the genes on one another, to propose an extension to the RBN update function proposed by Li *et al.* [LLL⁺04]. This novel, more biologically sound update function, along with real-life network topologies, fills another gap of the original model where the nodes' update function are completely random.

Some very preliminary results along this line have been presented at the conference [DGT⁺09]. Here we deepen and complete the investigation of the new model, propose a numerical way of discriminating the system's regimes that is more accurate than straightforward Derrida plots. We also conduct a full study of the systems' attractor space, and we validate the model using a third real-life instance of regulatory network proposed by Li *et al.* [LAA06].

This work is structured as follows: first, we describe the two real-life regulatory networks tackled in our model in the next section. Then in section D.3, we give a short introduction to RBN models, with particular attention to the identification of their dynamical regime. We also propose a new measure, the criticality distance, that allows to numerically discriminate between systems' regimes by analytically capturing the visual-only Derrida plots. The last part of this section is devoted to the description of the different measures that have been proposed in the literature to characterize the state space of RBNs. Section D.4 presents the new Activator Driven Additive node function, an extension of the RBN update function propose by Li *et al.* [LLL⁺04]. The regime characterization of this update function applied to the two real-life regulatory network substrates is discussed in sections D.5 and D.6. Section D.5.1 focuses on validating the new update function using a regulatory network where the actual Boolean update functions are known. The state space of new RBN models dynamical behavior is described in section D.7. In section D.8 the resilience of the two systems to small perturbations is investigated. Finally, the last section discusses the results obtained and outlines possible future lines of research.

D.2 Yeast and Embryonic Stem Cells Regulatory Networks

In this section, we give details on the two cases of real-life regulatory networks used in our model. The first one, proposed by Chen *et al.* [CXY⁺08], is a part of the transcriptional regulatory network of embryonic stem (ES) cells inferred from ChIP-seq binding assays and from gene expression changes during differentiation. We added activating informations (+ and - signs on the figure D.1(a)) inferred from gene co-expression and slightly compacted this network. The second one, described by Li *et al.* [LLL⁺04] and as used by Stoll *et al.* [SRN07], is the regulatory network underlying cell cycle in yeast. Both networks have eleven genes. Figure D.1 shows these networks, while Table D.1 shows some of their statistical properties.

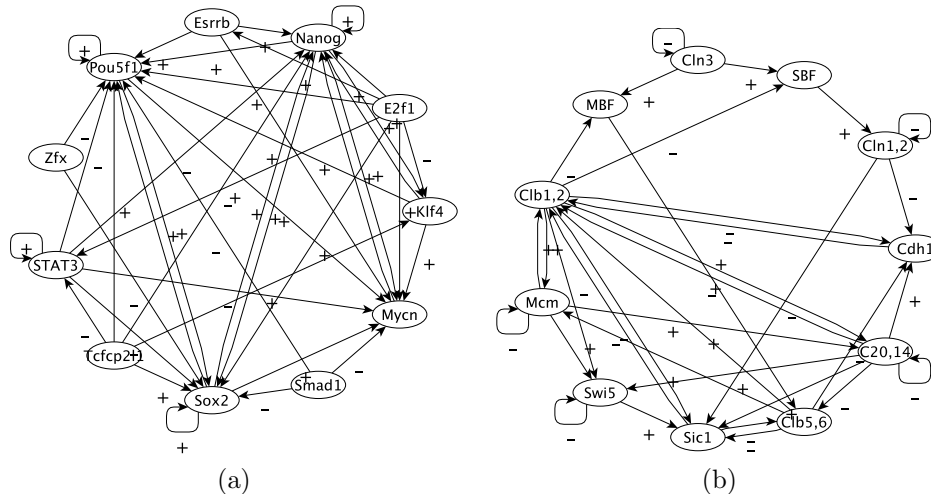


Figure D.1: Genetic Regulatory Networks. A representation of (a) the transcriptional regulatory network in ES cells and (b) the yeast cell-cycle regulatory network. Arrows point from transcription factor to the target gene. Signs + (respectively -) represent activating (respectively repressing) links.

Although the networks have too few nodes for a reliable statistical study of their degree distribution, we concluded that neither the input nor the output degree distributions show any similarities with original RBN's random topologies, where the connectivity $K = 2$ was fixed, or with Aldana's scale-free input, Poisson output distributions. To reliably establish these degree distributions, one would need to sample at least several tens of nodes for random graphs and many more over several orders of magnitude for scale-free ones, due the long tail of the distribution. However, these data

	ES cell	Yeast
N	11	11
mean degree	3.72	3.09
enhancer proportion	0.71	0.44

Table D.1: Properties of real life gene regulatory network used in this study.

are not currently available. Thus the need to use Derrida plots to determine the regime of our models. In this work, we abstract details of the genes themselves, as their individual properties do not have any consequences on the systems dynamics, beyond their activating or repressing effect.

D.3 Random Boolean Networks Modeling

Random Boolean Networks were introduced by Kauffman [Kau69, Kau93] more than thirty years ago as a highly simplified model of GRNs. Over the years, numerous other different models have been introduced [BB01, SBB00, HMICnt], but RBNs remain very attractive in their simplicity and ability to include novel concepts. In RBNs, each node represents a gene whose state is a Boolean variable S_i and each directed edge, the influence of a gene on another.

The interconnection topology is considered to be a regular random graph with exactly K incoming and K outgoing edges for each gene. A distinct function is given to each node in order to decide state changes according to the state of all in-neighboring genes (i.e. those nodes having an edge directed to the considered target gene). The lookup table describing the update function is randomly generated according to a parameter p capturing the probability that a gene's state at the next time-step is *active*. The state $S(t)$ of the system at time t is defined as the ensemble of all the nodes states $\{S_i(t)\}_{i=1}^N$. The state changes are fully deterministic, synchronous and instantaneous.

Therefore, these systems, when starting from an arbitrary state S at time $t = 0$, will go through a set of transient states before eventually cycling in a subset of one or more states called an *attractor*. According to Kauffman [Kau69, Kau93], only attractors that are short and stable to perturbations are of biological interest. In our research, we aspire to inject modern knowledge into the original RBN model, making it more biologically plausible.

A questionable assumption of the original RBNs model is the totally random interaction amongst genes with a fixed connectivity K [Kau69] or following a predefined degree distribution, such as scale-free or Poisson [Ald03]. In this work, we will take advantage of the real-life topologies defined in the previous section and use them as the substrate for our Boolean networks. Each gene of the GRN will be replaced by a Boolean variable which specifies whether or not the gene is expressed. In addition, we attach an update function to each node as described above, but in this work we compare the effect and performance of two different update function types, the random update function of the original RBN model, and a biologically inspired additive function described in the section D.4.

D.3.1 Regimes of RBNs and how to identify them

Original RBNs go through a phase transition at certain values of the fixed degree K of the nodes and the probability p of expressing a gene in the random update function. The critical regime can be achieved by satisfying the equation: $K_c(p) = [2p(1-p)]^{-1}$. Thus, when the parameter p is set to 0.5, the critical connectivity is achieved as $K_c(0.5) = 2$. If $K_c > [2p(1-p)]^{-1}$, the system will tend to be chaotic, and ordered otherwise. Considering current knowledge about GRNs, some of Kauffman's properties of the model are now subject to criticism.

In Aldana's scale-free model [Ald03], where the output degree distribution follows a power-law $p(k) \sim k^{-\gamma}$, where k is the variable node degree, this phase transition is obtained by setting γ around 2.5. In our case, we use real-life networks and not hand-made ones, and thus, we cannot tune any property of the network topologies to obtain the desired critical regime, or even to identify the regime of one of our network. Instead we use a dynamical property of the whole system which

is graphically represented by Derrida plots, proposed by Derrida *et al.* [DP86], used by Kauffman [Kau03], and widely accepted as a visual way of discriminating the regime in which RBN-like dynamical systems evolve.

This representation is meant to illustrate a convergence versus a divergence in state space that can in turn help characterizing the different regimes. It uses the Hamming distance H , defined as the normalized number of positions that differ when comparing two (binary) strings. These plots show the average Hamming distance $H(t)$ between any two states $S_a(t)$ and $S_b(t)$ and the Hamming distance $H(t+1)$ of their respective subsequent states $S_a(t+1)$ and $S_b(t+1)$. Figure D.2 shows a typical instance of Derrida plots and curves for all three regimes. Derrida plots of systems in the chaotic regime will remain above the main diagonal $H(t) = H(t+1)$, i.e. their distance tends to increase during a certain time, then cross the main diagonal from above. Systems in the critical regime remain *on* the main diagonal at the beginning and then stay below the main diagonal.

Ordered systems remain under the main diagonal at all times. In other words, systems in the critical regime we are interested in, which lies in the ordered regime at the edge of chaos, are characterized by Derrida curves that remain as close as possible to the main diagonal before diverging.

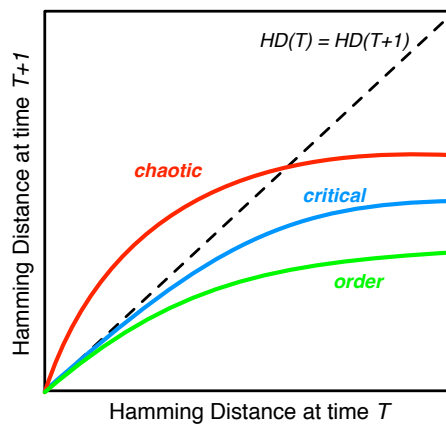


Figure D.2: Derrida plot of the original RBN model (see text).

For the two real-life regulatory network models in this work, it is obviously not possible to tune the connectivity parameter K , since the interconnection topology is fixed by experimental data. However, when the systems' dynamics are driven by the original nodes' random update functions (RUFs), the probability p could still allow the two models to be in different regimes. The number of all possible states for a given RBNs, i.e. with a single set of RUFs, is 2^N , where N is the number of genes in the system. In our case, $N = 11$, therefore there are $2^{11} = 2048$ possible states. The set of possible RUFs, even for a reasonably small subset of genes like the present, makes exhaustive enumeration impossible for original RBNs. Therefore, we resorted to statistical sampling by performing numerical simulation of 100 different sets of RUFs for each value of p . At first, p varies in the interval $[0.1, 0.9]$ by steps of 0.1. Having identified the values of interest p_i , we narrowed the interval to $[p_i - 0.05, p_i + 0.05]$ with a finer step of 0.01 to identify the values p_c that are closest to the critical region.

Figures D.3 and D.4 show Derrida plots with steps of 0.1 (a) and the finer version (b), where we adapted the scale to best show the regions of interest with a step of 0.01. As there are only 2^{11} possible states, we computed average Hamming distances over exhaustive enumeration of all possible states. In other words, we identified all pairs of states $\{S_a; S_b\}$ that are at a distance $H(S_a, S_b) = 1$ and computed the average Hamming distance of their subsequent states $H(S'_a; S'_b)$, and then moved on to a distance $H = 2, H = 3, \dots, H = 11$.

For the two regulatory network models, figures D.3 and D.4, depict the Derrida curves according to their values of p , as the RUF functions are symmetrical for values of $p \equiv 1 - p$. If not for sampling reasons, pairs of curves would superimpose, and therefore, to facilitate the interpretation of the results, we only plot curves for values of $p = \{0.5, 0.6, 0.7, 0.8, 0.9\}$. As shown in the figure D.3(a)

for transcriptional regulatory network in ES cell, the interesting values of $p_i \approx 0.8 - 0.9$, and symmetrically, $p_i \approx 0.1 - 0.2$. These are the values we chose to investigate with finer steps in figure D.3(b), revealing that in the case of ES cells the critical threshold value is close to $p_c \approx 0.87$, and symmetrically $p_c \approx 0.13$.

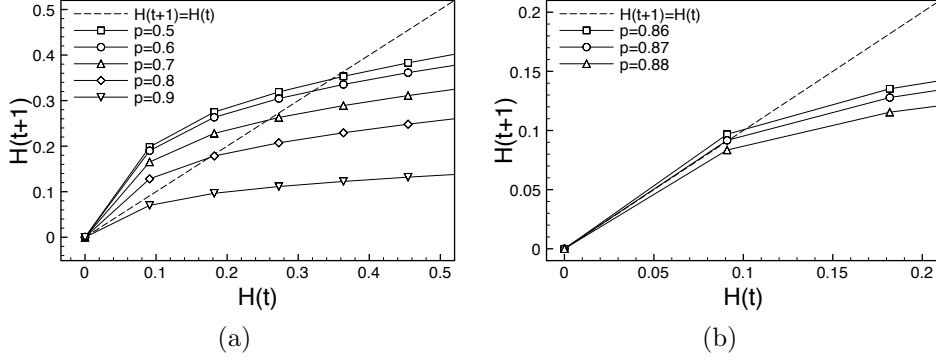


Figure D.3: Derrida plots of RUFs for ES cell, (a) $p \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$ (curves for $p \in \{0.1, 0.2, 0.3, 0.4\}$ are not reported as RUF rules are symmetrical), and (b) only values close to the critical gene expression value p_c are investigated with refinement steps of 0.01. Please note the two different scales in the axes.

Also for the case of the yeast cell-cycle regulatory network in figure D.4(a), we identified p_i to approximately the same values, more finely investigated in Figure D.4(b), where again $p_c \approx 0.83$, symmetrically $p_c \approx 0.17$.

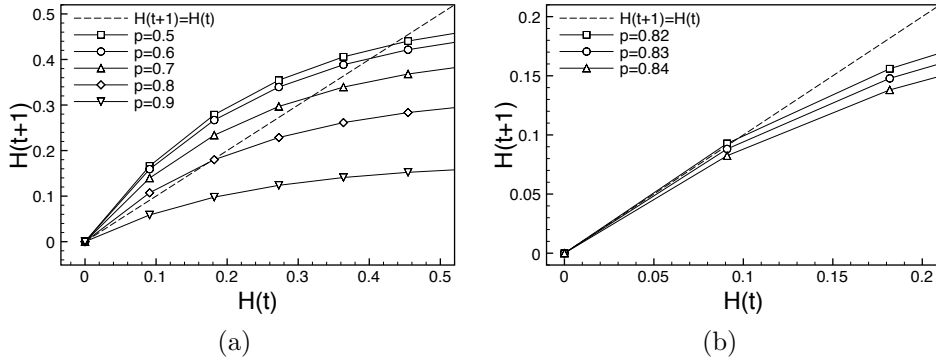


Figure D.4: Derrida plots of RUFs for yeast cell, (a) $p \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$ (curves for $p \in \{0.1, 0.2, 0.3, 0.4\}$ are not reported as RUF rules are symmetrical), and (b) only values close to the critical gene expression value p_c are investigated with refinement steps of 0.01. Please note the two different axes scales in the figures.

D.3.2 Criticality Distance

Originally, Derrida plots are a visual-only tool to define a system's regime. Nevertheless, as they are obtained as averages of experimental values of Hamming distance H' between two states s'_1 and s'_2 over a finite set of discrete values of Hamming distances H of their previous steps s_1 and s_2 , we propose a single numerical value that characterizes the distance of a system's Derrida plot to the main diagonal. This new *normalized criticality distance* (D) takes into account that the closeness to the main diagonal is more important for smaller values of H . The normalized criticality distance is obtained as follows:

$$D = \left(\sum_{n=1}^N \frac{1}{H_n} \right)^{-1} \sum_{n=1}^N \left| \frac{H_n - H'_n}{H_n^2} \right|$$

The closer D is to zero, the closer our system is to the critical regime, and therefore, the more interesting it is for in the context of this work. We use this new metric, in addition to the visual Derrida plot, to determine for which parameter sets the investigated systems are in the *critical* regime. Figure D.5 below shows how the minimal value D evolves with respect to the probability of gene expression p of each nodes' internal update function.

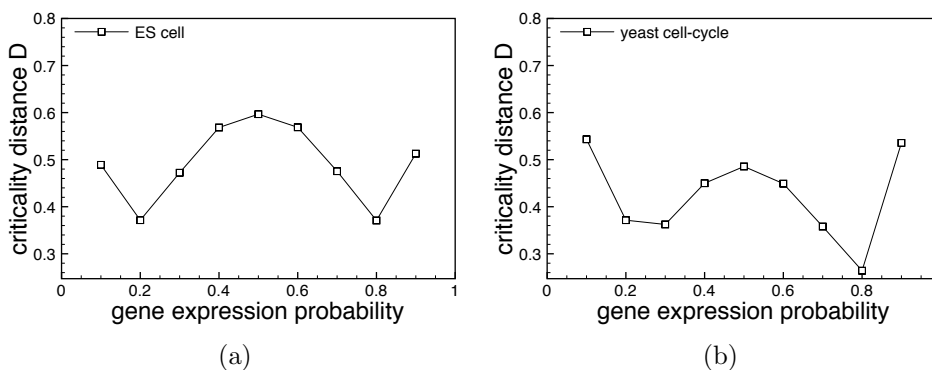


Figure D.5: Minimal Criticality Distances of Random Update functions. Criticality distances computed for each gene expression probability of RUF for both ES cells and Yeast from the Derrida plot/HD data.

For both networks the results obtained for p_c using the Derrida plots agrees with that found using the criticality distance.

D.4 Modeling the Yeast and the Embryonic Stem Cells Regulatory Networks

In the original RBN model, each node was assigned a deterministic distinct random update function (RUF). Even if their exact values are unknown, it is clear that gene update functions should not be random. Recent results suggest that genes expression rests on the combined effect of regulatory inputs that can have either an *activating* (+) or *repressing* (−) action on their target genes. Nowadays, we believe that the Boolean function should be closer to an additive function to better match real-life regulation mechanisms, where the influence of the genes upstream of the target, along with its own current activity state, could be summed in a way that takes into account the *activating* or *repressing* effect of each influencing node.

Li *et al.* [LLL⁺04] proposed a simple additive dynamical rule that characterizes the temporal evolution of the state variable. They consider that both the activating and repressing factors have the same weight, and thus, the state of a target gene at the next time-step $S_i(t + 1)$ will be: *active* (1) if it receives a majority of *activating* components from already active genes, *inactive* (0) if it receives a majority of *repressing* components, or the state of the target gene will remain unchanged in the case the number of *activating* and *repressing* inputs are equal. Inspired by their work, we propose an update function shared by all genes that takes into account the fact that *activating* and *repressing* components could have uneven effects. In this case, a gene could require a majority by more than one active input to switch states. Therefore we introduce a *threshold* value T which has to be reached in order for a gene to become active. A gene's activation state at the next time-step $t + 1$ is now given by:

$$S_i(t+1) = \begin{cases} \text{active (1)} & \text{if } \sum_j S_j^+ > T \times (\sum_j S_j^+ + \sum_j S_j^-) \\ \text{inactive (0)} & \text{if } \sum_j S_j^+ < T \times (\sum_j S_j^+ + \sum_j S_j^-) \\ S(t) & \text{otherwise} \end{cases}$$

Where S_j^+ (S_j^-) is the state of an activator (repressor) of the target gene. Moreover, as some genes of our model might not have any repressors, and, if activated, should not remain in that state permanently, we add a *decay* component. In the case where an active gene has no repressor at all, we switch it to inactive manually at the next time-step. This update function is equivalent to Li's in the case where $T = 0.5$. We call our model for update function the Activator Driven Additive function (ADA).

It can be easily proven that all rules in this class correspond to a subset of the RUFs [Has95]. In fact, once given, for each node, the *activating* and *repressing* effects of its neighbors, for each possible configuration of the neighborhood, the lookup table of the corresponding additive rule of the node can be constructed. In this form it can be easily recognized as an instance of the RUFs in the original Kauffman's RBN model. Therefore, by using ADA functions with different T -parameter values in a RBN model, we are exploring the behaviors of a subset of classical RBNs.

Another interesting implication of this update function is that under this assumption the synchronous timing of the events coincides with the semi-synchronous topology driven update scheme we recently investigated in [DTG09]. This update sequence is neither fully synchronous nor asynchronous, but rather takes into account the sequence in which genes affect each other. In this scheme, only the activation of an activator or a repressor will have an effect on the list of nodes to be updated at the next time-step. Thus, the set of all nodes that have to be updated in each time step is formed by those genes that have at least an in-neighboring active gene, even when a RUF is used to evolve the model. On the other side, when an ADA function is employed, in a synchronous timing of the update events, a node is actually updated only if it has at least an in-neighboring active gene.

D.5 Regimes Characterization in Real-Life Networks

Just as the probability p can change Kauffman's original systems' regime from chaotic to ordered for a given connectivity K and set of RUFs, the T -parameter in our ADA model can change its regime. In the following section, we show for which values of T our model of real-life topology based Boolean networks using ADA exhibit a phase transition, and compare the dynamics of the two update functions.

As discussed in section D.3.1, the space of all possible states for a given RBNs is 2^N , where N is the number of genes in the system. In our case, $N = 11$, therefore there are $2^{11} = 2048$ possible states. In the case of ADA, where all nodes share the same Boolean update function, exhaustive enumeration is possible. At first, we let the threshold T parameter vary in the interval $[0.1, 0.9]$ by steps of 0.1. After identifying the values of interest T_i , we narrowed the interval to $[T_i - 0.05, T_i + 0.05]$ with a finer step of 0.01 to identify as precisely as possible the values T_c that are closest to the critical region. As there are only 2^{11} possible states, and thus the maximum Hamming distance $H_{max} = 11$, we computed average Hamming distances over exhaustive enumeration of all states. In other words, we identified all pairs of states $\{S_a; S_b\}$ that are at a distance $H(S_a, S_b) = 1$ and computed the average Hamming distance of their subsequent states $H(S'_a, S'_b)$, and then moved on to a distance $H = 2, H = 3, \dots, H = 11$.

The left-hand sides of figures D.6 and D.7 show Derrida plots with steps of 0.1. The regions of interest for the T values are $T_i \approx 0.7$ for ES cells. Let us note that in the ADA case, contrary to RUF, update rules are not symmetrical with respect to T and $1 - T$. For yeast cell-cycle, we see two regions worth investigating $T_i \approx 0.2 - 0.3$ and $T_i \approx 0.7$. The in-depth examination of ADA simulation results for values of the T_i parameter demonstrate that results become undistinguishable (thus, figures are not shown) when the step between T values becomes small. This is due to the fact that the ADA function is less sensitive to T for genes with a low input degree. In the case of ES cells, $T_c \approx 0.68$ and in the case of yeast $T_c \approx 0.25$ or $T_c \approx 0.6$. In this last value of $T_c = 0.6$,

	RUF p values			ADA T values		
	order	critical	chaos	order	critical	chaos
ES cell	0.1, 0.9	0.13, 0.87	0.5	0.2	0.68	N/A
yeast cell	0.1, 0.9	0.17, 0.83	0.5	0.9	0.25, 0.6	0.4

Table D.2: Real life network critical values. For systems under RUF, we show the function's gene expression probability p values for all three regimes, for both ES cells and Yeast cell-cycle. In the case of ADA, we give threshold values T also for all three regimes and both studied networks.

curves for several very close values of T coincide. This observation, derived from visual analysis of Derrida plots (figures D.6(a) and D.7(a)), is supported by the measurement of the criticality distances. These are reported on the right-hand sides of the two figures D.6(b) and D.7(b).

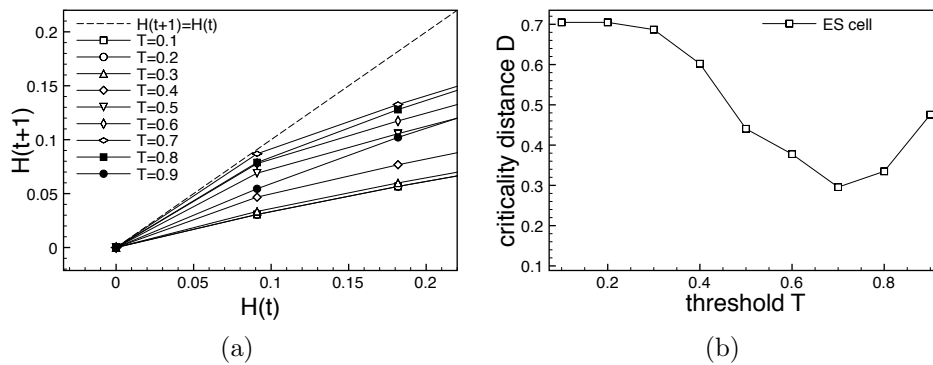


Figure D.6: Derrida plots and Criticality Distances of Activator Driven Additive functions for the mouse embryonic stem cell regulatory network.

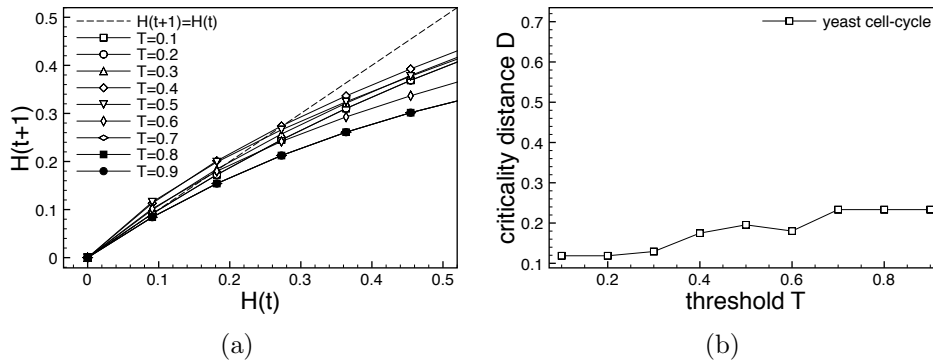


Figure D.7: Derrida plots and Criticality Distances of Activator Driven Additive functions for the yeast cell-cycle regulatory network.

We summarize the results described above in the table D.2. From these results, we observe that the ADA-thresholds T have comparable values in the two GRNs studied in this paper: $T_c^{ES} \approx T_c^{yeast}$. The same applies to the probabilities p_c of gene expression in RUF.

D.5.1 Validation of the Model on a Network with Known Update Rules

The two partial GRNs presented in section D.2 are practical RBN models of dynamical regulatory interaction networks as they are small enough to study exhaustively all 2^{11} possible states of the system. Therefore, we can fully define the update functions for each node and every possible input

combination. Yet, in these particular cases, we have nothing to compare these functions against. In order to validate ADA update functions, we used another regulatory network presented in [LAA06].

In this work, Li *et al.* define a dynamic Boolean model of plant guard cell abscisic acid (ABA) signaling. This hormone allows plants to adjust water conservation within the organism. In the original model presented, the regulatory network is made of 42 cellular components. For each of these components, in addition to their connections, the authors defined the Boolean function that decides the state of each component at the next time-step. This new information can help us assess the validity of the ADA update function.

ABA Network Reduction

Another helpful feature of the ABA regulation network is that 4 of the components have a predefined Boolean value and those do not have an update function attached to them. This allows us to replace these *constant* components (i.e. that are assigned a boolean value) in the update function of the 38 remaining ones, and then to replace some more that become constant. For example:

```
ABA = ABH1 = ERA1 = AGB1 = True
SphK = ABA
S1P = SphK
GPA1 = (S1P or not GCR1) and AGB1
```

becomes after simplification:

```
ABA = ABH1 = ERA1 = AGB1 = True
SphK = True
S1P = True
GPA1 = (True or not GCR1) and True = True
```

Following this logic, the fully simplified ABA network becomes:

```
NOS = Ca2+c
NO = NOS
GC = NO
ADPRc = NO
cADPR = ADPRc
cGMP = GC
PLC = Ca2+c
InsP3 = PLC
CIS = (cGMP and cADPR) or InsP3
Ca2+ATPase = Ca2+c
Ca2+c = CIS and (not Ca2+ATPase)
KAP = not Ca2+c
KEV = Ca2+c
```

This new simplified network is reduced to only 13 components and it is therefore possible to enumerate all possible 2^{13} states.

Determining the ADA model's regime

In order to determine the regime in which the ABA model evolves, we used the Derrida plots, shown in figure D.8. The criticality distance is not as useful in this instance, as there is no comparison to be made.

The Derrida plot of the ABA model with real-life update functions depicted in figure D.8 clearly shows that the system evolves near the critical regime. Therefore, we use the Derrida plots and criticality distance D to determine the critical values p_c of RUF, respectively T_c for ADA, when

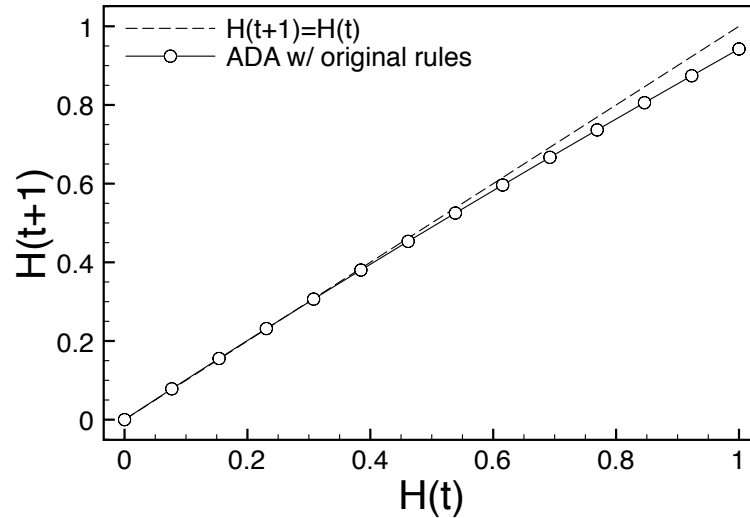


Figure D.8: Derrida plot of the simplified ADA model with the original real-life update functions.

each of these function families is substituted in the simplified ABA model. In the case of RUF, we average out the results over 100 sets of different update functions. Derrida plots for ABA system with ADA update over the full scope of $T \in [0.1, 0.9]$ values together with the corresponding evolution of the criticality distance are shown in figures D.9(a) and D.9(b). The plot for RUF's over the same range of p values is depicted in figures D.9(c) and D.9(d).

From the analysis of the figures above, we observe that in the case of ADA (figure D.9(b)), there are only two values of the criticality distance, one for $T = 0.5$ and a larger one for $T \neq 0.5$. The first one is the closest to the original ABA model critical regime when $T_c = 0.5$. In the case of RUF (figures D.9(c)-(d)), the closest gene expression value to the regime of interest is $p_c = 0.6$.

Comparing ADA, RUF and real-life functions

Using the ABA network described above, we have fully defined each node's lookup table according to its real-life function. Subsequently, we have replaced the original update functions of each node with ADA functions and $T_c = 0.5$ to define the new lookup tables. This allows us to compare in a very straight forward manner how close ADA is to this specific case real-life activations. In addition, we have also replaced the set of node functions by a sample of 100 RUFs and $p_c = 0.6$, and averaged out the results. In order to keep the measurements simple, we have computed the normalized Hamming distance between the real-life ABA function and ADA, or respectively RUF. Each node's lookup table size is $2^{k_{in}+1}$, where k_{in} is the node's incoming degree. Therefore, the added size of all nodes lookup tables is $11 \times 2^2 + 1^3 + 1^4 = 68$.

The Hamming distance $H_{(ABA,ADA)} = 6/68 = 0.088$ and $\bar{H}_{(ABA,ADA)} = 17.1/68 = 0.251$. These results show that in this particular case, ADA is significantly closer to the real-life ABA function than a random function. Although this finding cannot be generalized at this time, it suggests that at least in some cases, the ADA function ought to be closer to the real-life update function of a regulatory network system.

D.6 Enhancer Proportion vs. Threshold of Critical ADA Networks

One legitimate question one could ask is whether the distribution of the enhancer and repressor over the gene interaction links has an effect on the critical threshold T of the ADA update function,

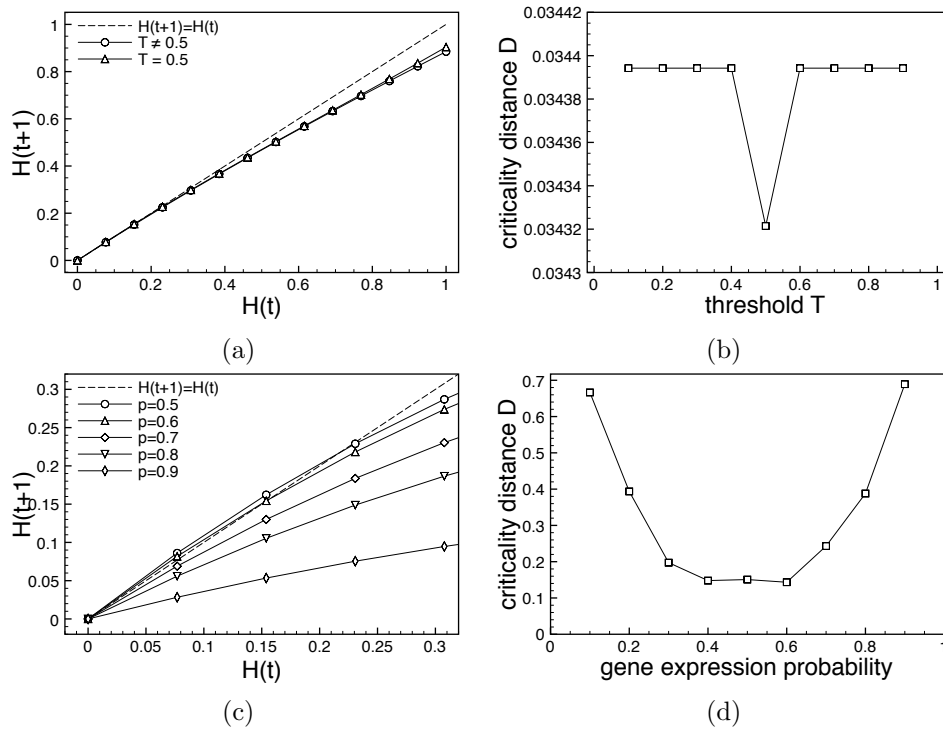


Figure D.9: Derrida plots (a)(c) and criticality distance vs. the threshold (b), respectively gene expression probability (d). The upper row (a)(b) shows the ABA system where the original rules have been replaced by ADA update function. In the lower row (c)(d), rules have been replaced by RUF and results are averaged over 100 random rules sets.

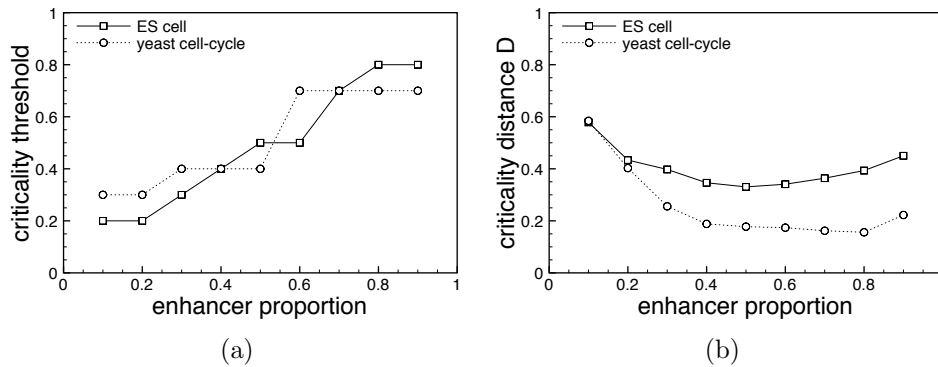


Figure D.10: Threshold and Criticality Distance with respect to Enhancer Proportion of Critical ADA Networks. On the left-hand side (a) the evolution of the threshold T . On the right-hand side (b) the evolution of the minimal criticality distance.

		ES cell	Yeast
real-life	$P_{enhancer}$	0.71	0.44
	T_c	0.17/0.83	0.25
	$D_{min}(T_c)$	0.29	0.11
man-made	$P_{enhancer}$	0.5	0.8
	T_c	0.5	0.7
	$D_{min}(T_c)$	0.33	0.16

Table D.3: Real life network properties.

i.e. the threshold for which the criticality distance is minimal, and on the minimal critical distance D itself, therefore on the criticality of the system itself. In order to answer this question, we ran simulations on both real-life GRN topologies, assigning the enhancer/repressor values of the links at random, with enhancer proportions p_e ranging from 1/10 enhancers to 9/10. For each value of p_e we generated 1000 different distributions and averaged our results over this number. Figure D.10 shows the evolution of the T values and of the minimum value of D for each discrete value of p_e for both network topologies.

Results in Figure D.10 show that the proportion of enhancers/repressors does have a major influence on how close the systems can get to the critical regime. In Figure D.10 (a) we notice that the same trend of increasing T is necessary in both system types in order to keep D to a minimum, although there are plateaus in the yeast case when $T \geq 0.6$. Table D.1 in Section D.2 shows that the real-life ES cell and Yeast cell-cycle have enhancer proportions of 0.71 and 0.44 respectively. These values agree with close to minimal, although not absolute minimal, values of D found in Section D.5.

Table D.3 shows the values of enhancer proportion $P_{enhancer}$ and critical threshold T_c that offer the minimal criticality distance $D_{min}(T_c)$ in both the original networks and the man-made ones described in this section.

Interestingly the criticality distance is smaller in the real-life case both for ES cells and yeast. This indicates that the right proportion of enhancer/repressor does not suffice to get close to the critical regime: the actual individual identity links does have an indubitable importance.

D.7 Dynamical Behaviors of Real-Life Regulatory Networks

A key notion underlying the behavior of deterministic discrete dynamical networks is that they organize their state space into a set of basins of attraction. When a discrete dynamical network perspective is used to investigate genetic regulatory networks behaviors, understanding how the

range of stable cell types can exist with identical genes becomes clearer. Different attractors or basins of attraction into which network dynamics settles from various initial states can be seen as cell types or modes of growth for unicellular organisms, while the trajectories leading to attractors can be seen as the pathways of differentiation.

To better understand real-life regulatory networks, it is not enough to qualify their regime. It is therefore useful to portray their state space. Several measures have been proposed to characterize the state space of a dynamical network by Wuensche [Wue89]. Of particular interest are the number and lengths of the attractors in the state space, together with the sizes of the basins of attraction. Directly connected to this latter characteristic is the basin entropy H . H is maximum if each state is its own basin of size one, and minimum when there is a single basin. Because it takes into account the relative basin sizes, it is quite insensitive to appearance of small and biologically irrelevant basins. Finally, according to Wuensche [Wue89]: ‘high leaf density, high branching, short transients, and small attractor cycles indicate order’. Leaves are states of the state space that do not have any predecessor, while transient times and branch lengths are the time steps (i.e. number of states) necessary from a state and a leaf respectively to reach its attractor.

In the following sections, we study the dynamical behavior of ES cells and yeast cell-cycle separately. In both cases we compare results obtained using the ADA update function and those obtained using random update functions found in classical RBN. In the case of ADA, where the update function is unique, we exhaustively enumerate the entire state space a single time. On the contrary, in the case of RUF, we sample 1000 different sets of rules unique to each gene. This is the reason why there is standard error information only in the RUF case.

D.7.1 Simulation of the Embryonic Stem Cell Regulatory Networks

Using the model of mouse embryonic stem cell regulatory networks with ADA, we constructed systems in the two available regimes: ordered and critical. Indeed, as can be seen in Section D.5, there is no value of T that clearly puts the system in the chaotic phase. Therefore, we use $T_{order} = 0.2$ and $T_c = 0.68$ for the unique ADA model. On the other hand, we built 1000 RUF models of the ES systems, with as many different sets of unique rule in each gene. In the RUF case, $p_{order} = 0.1$, $p_c = 0.13$, and $p_{chaos} = 0.5$.

Figure D.11 shows the numerical simulations results in terms of number of attractors for ADA, respectively average number of attractors for RUF, average attractor lengths, and average basin size. Error bars represent the standard error.

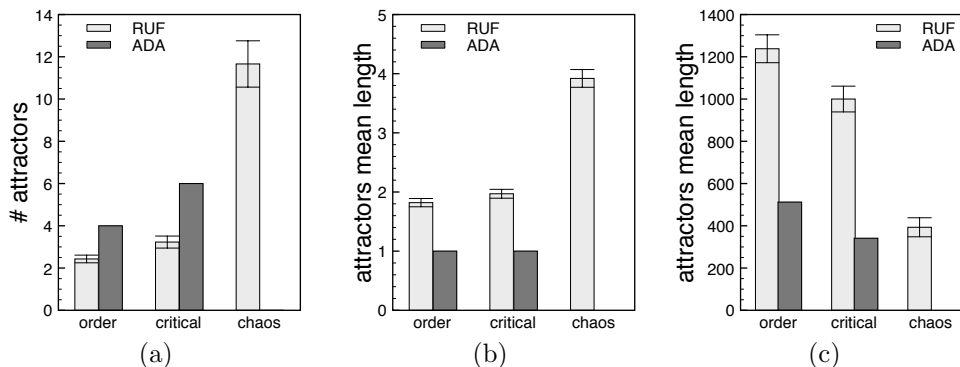


Figure D.11: Numerical simulation results for the ES model. (a) Attractors (average) number, (b) attractors average lengths, and (c) the average basin of attraction size. The statistics are computed on samples of RUF systems, hence the standard error bars, and exhaustively on ADA systems.

In the case of ES cells, we observe an increase in the number and length of the attractors, explaining the shrinkage in the basins’ sizes, as the systems are getting more chaotic. This agrees with previously obtained results on larger network models [DTG09], and it is aligned to Kauffman’s conjecture that the biggest increase in this characteristic should happen in the chaotic regime. In

the ADA case, we find more attractors of shorter length than in the case of RUF, almost all being point attractors. When studying at basin entropy H (not pictured in this work), we observe for RUF that the entropy tends to increase when moving toward the chaotic regime. This phenomenon is expected with the growth in the number of attractors (i.e. of basins of attraction), thus none of the basins seems to take over the others in size. ADA functions, on the contrary, show a drop in the value of H ($0.5 \rightarrow 0.3$), indicating that a few attractors have significantly larger basins of attraction. Between RUF and ADA models, mean transient times and mean branch lengths do not differ, showing a tendency to considerably increase in the chaotic regime. In ordered and critical regimes, the mean branch lengths and transient times are very short (smaller than 2 states), explaining the close to 1 probability of having leaf-states. These measures suggest that considering ADA functions we are focusing on a more biologically interesting and plausible subset of RUFs.

D.7.2 Simulation of the Yeast Cell-Cycle Regulatory Networks

In contrast with the mouse embryonic stem cell regulatory network, the yeast cell-cycle one with ADA can be found in all three regimes. Simulations for networks in the different regimes and with both RUF and ADA functions have been performed in the same manner as for the ES model. Figure D.12 shows the numerical simulations results in terms of number of attractors for ADA, respectively average number of attractors for RUF, average attractor lengths, and average basin size. Error bars represent the standard error.

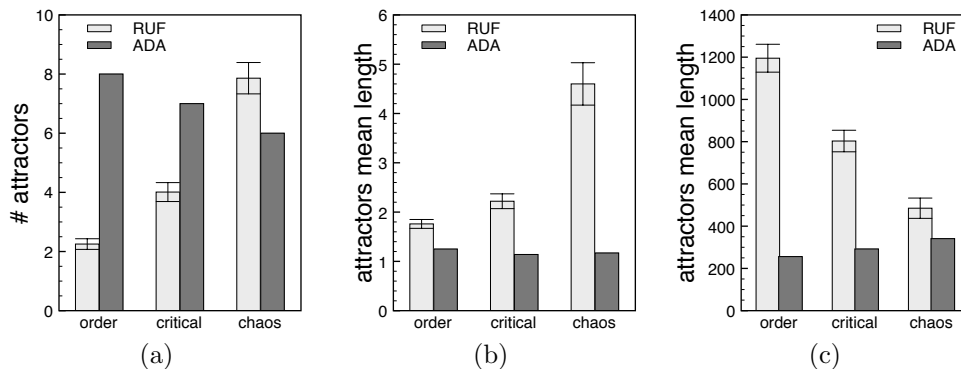


Figure D.12: Numerical simulation results for the yeast model. (a) Attractors (average) number, (b) attractors average lengths, and (c) the average basin of attraction size. The statistics are computed on samples of RUF systems, hence the standard error bars, and exhaustively on ADA systems.

In this case, while RUF behaves as expected with a growth in the number of attractors as the systems moves to chaos, surprisingly, the opposite behavior can be observed when ADA functions are employed. When considering basin entropy H , as expected RUF models tend to show higher values in critical and chaotic regimes. On the contrary, ADA systems' entropies dramatically drops from 0.6 in the ordered regimes to 0.3 in the critical one and to 0.1 when chaotic. This behavior could be expected if the decrease in the number of attractors was significant, which is not the case here. Therefore, we are witnessing the dominance in their basin sizes of a small number of attractors. These yeast cell-cycle systems thus show a biologically interesting feature, compatible of the assumption that attractors correspond to cell-cycles. Mean attractors lengths, basin sizes, transient and brach lengths, and probability of leaf-states show an identical behavior to that of ES cell regulatory networks.

D.8 Resilience to Small Perturbations

Failures in systems can occur in various ways, and the probability of some kind of error increases dramatically with the complexity of the systems. They can range from a one-time wrong output to a complete breakdown and can be system-related or due to external factors. Living organisms are robust to a great variety of genetic changes, and since RBNs are simple models of the dynamics of biological interactions, it is interesting and legitimate to ask questions about their fault tolerance aspects.

Kauffman [Kau00] defines one type of perturbation to RBNs as “gene damage”, that is the transient reversal of a single gene in the network. These temporary changes in the expression of a gene are extremely common in the normal development of an organism. The effect of a single stimulus can transiently modify the activity of a gene, resulting in a growing cascade of alternations in the expression of genes influencing each other. Although not agreed by all, some believe this to be at the origin of the cell differentiation process and guides the development.

The precise structure of attractor basins is of interest as it may reflect the stability of cell types to perturbation. A set of similar states can be specified for example that differ by one bit from a reference state (a Hamming distance of one). The distribution of the set across the basin of attraction held indicates the network’s response to a one bit perturbation to its current state of activation. The dynamics of the system might remain in the same basin or flip to a different basin.

For the mouse embryonic stem cell (left) and the yeast cell-cycle (right) regulatory networks, figure D.13 depicts the p_{HD} , the probability that two states at Hamming distance of one belong to the same basin of attraction. The two network models are studied both with RUF and ADA update functions. In the case of ES cells, both update functions show identical p_{HD} in the ordered regime, while in the critical region $p_{HD}^{ADA} > p_{HD}^{RUF}$. This same relationship holds in the critical regime of the yeast cell-cycle system, even though $p_{HD}^{ADA} < p_{HD}^{RUF}$ in the order. Therefore, in the critical region the ADA function shows higher probability, thus better resilience to single-gene perturbations.

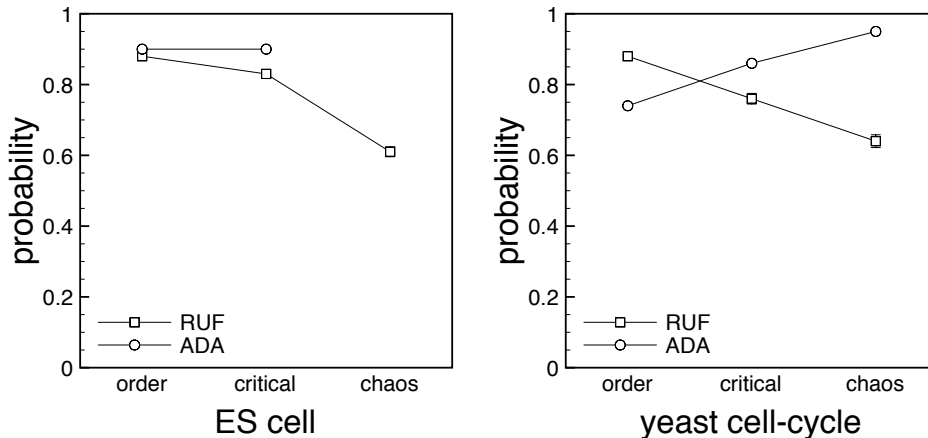


Figure D.13: Probability that two states at a Hamming distance of one are in the same basin of attraction for mouse embryonic stem cell (left) and the yeast cell-cycle (right) models. Both RUF (dark grey) and ADA (light grey) update function behaviors are shown.

D.9 Discussion, Conclusions and Future Work

Taking into account recent years’ advances in the field of cellular biology, we have proposed to identify under what conditions Kauffman’s hypothesis that living organism cells operate in a region bordering order and chaos holds. This property confers to organisms both the stability to resist transcriptional errors and external disruptions, and, at the same time, the flexibility necessary

to evolution. We studied two particular cases of genetic regulatory networks found in literature in terms of complex dynamical systems derived from the original RBN model. Therefore, we compared the behavior of these systems under the original update function and a novel additive function that we believe is closer to the actual role of living organisms.

The proposed functions, here called Activator Driven Additive (ADA), correspond to a subset of all possible Boolean functions of the original Random Boolean Network model. Moreover, using this set of update rules, the synchronous timing of the events coincides with the semi-synchronous topology driven update scheme we recently investigated. This update sequence is neither fully synchronous nor asynchronous, but rather takes into account the order in which genes affect each other.

In order to investigate the dynamical behaviors of this new model, we visualized the phase transition between order and chaos into the critical regime using Derrida plots. We also proposed a new measure, the criticality distance, that allows to numerically discriminate between different regimes by capturing the visual-only method implemented by Derrida plots.

Simulation results on two real-life genetic regulatory networks, the yeast cell-cycle and the mouse embryonic stem cell, show that there exist parameter settings in both update functions that allow the systems to operate in the critical region, and that these values are comparable in the two case studies. Both Derrida plots and criticality distances agree on the numerical values of the parameter for which the transition into the critical regime takes place. To better understand real-life regulatory networks, it is not enough to qualify their regime. The state spaces of the two real-life GRNs is portrayed using RBN-specific statistical measurements, confirming that the two systems operate at the edge of chaos. Moreover, in the critical regime, we show that ADA systems exhibit superior tolerance to transient perturbations than classical RBNs.

In order to validate ADA update functions, we used another bio-chemical regulation network operating near the critical regime (as confirmed by Derrida plot). For each node of this network, in addition to their connections, the authors defined the Boolean function that decides the state of each component at the next time-step. This new information can help us to assess the validity of the ADA update function. These results show that in this particular case, ADA is significantly closer to the real-life function than a random function. This also comforts us that, at least in some cases, the ADA function ought to be closer to the real-life update function of a regulatory network system.

A first improvement to the model could consist of the use of different threshold values for each node. Further investigations of the model should include in particular the use of weighted influences of the activator or repressor effects of a gene on another. This could be implemented by giving to each link of the network model a specific weight. The resulting nodes' ADA update functions could drive the model toward more realistic patterns of gene regulation dynamics. Finally, this new model should be validated on larger gene regulatory networks.

Authors contributions

CD conceived the design of the study, implemented the computational model and carried out the simulations. MT, FDC and PP participated in the design of the study and in the interpretation of the results. MG conceived the design of the study and coordinated the participants' contributions. All authors equally participated in writing the manuscript and approved it.

Acknowledgements

The authors thank in particular Réka Albert and her research associates for useful discussions of the network used to validate the proposed model. M. Tomassini and Ch. Darabos gratefully acknowledge financial support by the Swiss National Science Foundation under contract 200021-107419/1, and by the Rectors' Conference of the Swiss Universities. M. Giacobini and P. Provero acknowledge funding (60% grant) by the Ministero dell'Università e della Ricerca Scientifica e Tecnologica. F. Di Cunto and M. Giacobini acknowledge Compagnia di San Paolo financial support.

Part IV

Discussions, Conclusions & Future Perspectives

Chapter 10

Conclusions & Future Perspectives

Statistics: The only science that enables different experts using the same figures to draw different conclusions.

Evan Esar

10.1 Conclusions and Scientific Contributions

This five-year endeavor has led to both innovative and thorough research. It evolved in a guided organic fashion, where the main area of exploration were defined at the very early stages of the project and new idea have sprung naturally from ongoing research and advances in the fields studied as they were becoming available. This mixture of both planned and unplanned, although not uncommon in a work of this length and breadth conducted in collaboration with several international groups, kept it at the cutting edge of computational systems biology. Clearly, the biological inspiration, the computational agent based simulation, the analysis of remarkable properties of the systems owing to the networks' attributes, the quest for emergence in complex systems, and the thorough study of the robustness and fault tolerance abilities of the models can be followed throughout the entire work. Over time, the context to which these common fields of study have evolved from theoretical distributed computation with cellular automata to applied molecular biology, opening new collaborations and making results potentially useful to the biological community at large. At the end of each subsections below, we also hint at possible future developments we might undertake in the field.

Finally, I am drawing conclusions in direct relations with the theses stated at the beginning of this document.

Addressing the First Thesis

Using structured evolutionary algorithms, we have successfully evolved the underlying topology of cellular automata for solving prototypical tasks, namely the density and the synchronization tasks. Both starting from initial population of regular one-dimensional lattices networks and random graphs, we have witnessed that, with very little guidance, EAs bring about topologies that are at the crossroads of regular and random structures, and solve the tasks with high efficiency. Moreover, they share properties, in terms of network science, with both types of topologies.

Addressing the Second Thesis

The structures we evolved using EAs as support for CA computation show numerous topological similarities with biological, social and technological networks. We have shown by computer simulations that they also share their remarkable stability and robustness in the face of probabilistic

perturbations. Indeed, cellular automata that use these evolved structures keep their ability to perform the task at hand in cases where systems on regular structure fail to do so.

Addressing the Third Thesis

In conjunction with previously discovered structural properties of genetic regulatory networks, we have developed a novel semi-synchronous update timing for a Boolean model of regulatory networks. When included in the model, this new topologically driven update scheme allows it to retain its attractive simplicity, and similar or superior performance while making the model more biologically realistic. In addition, we have thoroughly studied the influence of the network topology on the system's behavior. Further advances in abstract representations of GRNs could introduce more realistic node update functions such as continuous activation, etc., in the context of plausible topologies.

Addressing the Fourth Thesis

In addition to the steady and in some cases improved performance, our new model also shows excellent flexibility to support evolution and robustness in the face of minor perturbation, that can be interpreted as mutations or transcriptional errors in biology. Our results clearly show that for similar models where only the topological structure of the underlying network differs, those of biological inspiration are more robust. This comforts us in the speculation that outstanding stability and evolvability displayed by regulatory networks is not to be solely attributed to the biochemical interactions of the organic compounds, but also to the very topological structure of these interactions.

Addressing the Fifth Thesis

Actual regulatory networks of biological organisms, although partial, have opened up new horizons in our research. Not only have we been able to put our models to the test on real-life cases, it has also provided us with invaluable insight on gene-gene interactions. This, in turn, allows us to partially overcome the main shortcoming of Kauffman's original random Boolean network model of regulatory networks, namely the randomness. By updating both the structure and the kind of interaction to mimic recent finding in genetics, we have been able to develop a new update rule, that is completely deterministic, and we believe closer than ever before to be of use to biologists as a tool of predicting interactions amongst genes. Naturally, in the view of the very limited size of the regulatory networks studied so far, much is still to be done as larger systems will become available.

10.2 Future Perspectives

As new data become more available, models for biological phenomena will greatly be improved by the introduction of this new knowledge. Below, I describe a few possible direction my research is susceptible to take:

- lacking of time, I was not able to apply fully close the circle and apply evolutionary algorithms to models of biological genetic regulatory networks. There are number of situations where EAs could be applied to models of GRN, for example:
- use EAs to reverse-engineer gene-gene interaction links from microarray data using an instance of the additive update rules proposed in this work;
- use EAs to reverse-engineer the update functions in boolean GRN models from microarray in a known sub-network model of GRN;
- use EAs to assign the threshold values in the boolean GRN model that uses an instance of the additive update function;

10.3 Final Considerations

Nowadays, in the field of systems biology, modeling complex dynamical systems is rapidly becoming unavoidable. Every aspect of science uses models to make predictions and study systems, yet, models are, by definition, imperfect, as they relay on assumptions and simplifications. This does not mean they are not useful, take weather forecast for instance. However, their use is still limited to a particular instance of a problem, under strict conditions. The fact that they are useful, sometimes crucial, justifies the investment put into improving them to better match reality.

Biological models are, in my opinion, only at their genesis. This view is backed by the tremendous efforts of scientist to work on them, with them, and improve them. Yet, we live in an interesting time, computer science and computational models are following the trend of fundamental mathematics or statistics. Indeed, computer science and models are not exclusively viewed as fields of study anymore, but more as a useful tools at the service of other scientific disciplines. Consequently, biologists are opening up to computational models, understanding the potential insight they can provide. A perverse effect is that people of either field loosely cross the line in the other one: biologists become computer modelers, and vice-versa. If that was an acceptable situation not long ago, the increasing complexity of the phenomena modeled start to push the limits of the scientists' abilities. It is no longer enough to be a computer scientist with moderate biological knowledge to build accurate models of phenomena of the complexity and granularity that is expected today by the scientific community. Inversely, a biologist and amateur programmer is also a limitation. Biologists, wet lab scientist must work hand in hand with modelers and computer specialists. The design of the models must be based on lab experiments and results, that are built in collaboration between all parties. I am personally very excited to be at the crossroads between modeling and biology.

Part V

Appendixes

Appendix A

Tools

We must use time as a tool, not
as a crutch.

John F. Kennedy

Most of the tools and simulators were developed *in house*, using non proprietary programming languages, mainly *Java* and *Python*. Statistical analysis were made using *R* scripts, or *Octave*. Figures were produced with *Plot* for *Mac OS X*.

Simulation were run on grids of *Apple Macintosh* workstations linked with *Apple's Xgrid* tool to distribute the processes over the grid(s). Results were stored in a *mySQL* database. All these systems were ran and maintained at the University of Lausanne.

Online Resources

All the libraries and simulators used during this work are available on my web page:

<http://cdarabos.googlepages.com>

Appendix B

UML Class Diagram - Jcell

This annex contains the UML class diagram of the main evolutionary CA simulator.

```

TopologyIndividual
-minDeg : int
#maxDeg : int
-mutationProb : double
-inItConnectionProb : double
-randomProb : double
-tousConnexes : boolean
#initDegree : int
#alleles : int[]
-phi : double
-clusteringCoeff : double
-avgDegree : double
-CPL : double
-degrees : int[]
-debug : boolean = false
-nodes : int[]

+TopologyIndividual(p : Params)
+TopologyIndividual(len : int, minDeg : int, maxDeg : int, mp : double, icp : double, k : int, randomP : double, tc : boolean)
+TopologyIndividual(len : int, k : int)
+TopologyIndividual(len : int)
+setInitDegree(k : int) : void
#disconnectAll() : void
+getAllele(locus : int) : Object
+getIntArrayAllele(locus : int) : int[]
+getGeneInAllele(locus : int, conn : int) : int
+setAllele(locus : int, allele : Object) : void
+connectNodes(fromNode : int, toNode : int) : boolean
#areConnected(fromNode : int, toNode : int) : boolean
-fin dFreeIndex(node : int) : int
+disconnectNodes(fromNode : int, toNode : int) : boolean
#forceDisconnectNodes(fromNode : int, toNode : int) : boolean
+getAlleleDegree(node : int) : int
+computeAvgDegree() : double
+computeSumDegree() : int
-isConnectable(node : int) : boolean
-isDisconnectable(node : int) : boolean
-existsConnectable(toIgnore : int) : boolean
+mutate(r : Random, locus : int) : void
+mutate(r : Random, locus : int, x : int) : int[]
+getPositionInAllele(nodeFrom : int, nodeTo : int) : int
+setLength(len : int) : void
#resetRing() : void
+hasRing() : boolean[]
+setRandomValues(r : Random) : void
+copyIndividual(ind : Individual) : void
+computeAll() : void
+toString() : String
+toDot() : String
+equals(obj : Object) : boolean
-isInAllele(allele : int[], gene : int) : boolean
+clone() : Object
+isValid() : boolean
+isConnexe() : boolean
-checkConn(index : int) : void
+store() : void
+computeClusteringCoeff() : double
+getNodeClusteringCoeff(index : int) : double
+computePhi() : double
#getNbShortCuts() : int
#isShortcut(start : int, end : int) : boolean
-checkNextPace(start : int, end : int) : boolean
+convertToSpecificPhi(k : int, phi : double) : boolean
+getNodesPerDegree() : int[]
+computeNodesPerDegree() : int[]
+computeCPL() : double
-areAllSpFound(sp : int[]) : boolean
-setRandomGraph() : void
+plot() : void
-debug(s : String) : void

```

```

NoisedScaleFreeDensityProblem
-nbIC : int
-faultProb : double
-lowerBound : double
-upperBound : double
-maxStepsTimesIndLen : int
-isSync : boolean
-maxSteps : int
-recordEval : boolean
-r : Random
-useRandomRuleIfTiedScore : boolean
-stopIfConvergence : boolean = true
-sumHD : int
-score : int
-scoreF : int
-sumSteps : int
-sumStepsF : int
-ics : int[]
-icsMajority : int[]
-faultedNodes : int[]
-roulette : int[]
-nbFaltedNodes : int
-fitness : double = 0
-fitnessF : double = 0
-avgHD : double = 0
-avgSteps : double = 0
-avgStepsF : double = 0
-noiseType : int
-evalCounter : int = 0
-sumFitness : double = 0
-sumFitnessF : double = 0
-sumAvgHD : double = 0
-sumAvgSteps : double = 0
-sumAvgStepsF : double = 0
-cpt : int = 0

+NoisedScaleFreeDensityProblem(r : Random, p : Params, learning : boolean, fp : double, noiseType : int)
+eval(individual : Individual) : double
-evaluateIC(ind : TopologyIndividual, icIndex : int) : void
-makeSyncStep(ind : TopologyIndividual, conf : int[], fp : double) : int[]
-makeNonSyncStep(ind : TopologyIndividual, conf : int[], fp : double) : int[]
-getNewValue(allele : int[], alleleDeg : int, conf : int[], alleleIndex : int, fp : double) : int
-inItCs() : void
-generateRandomIC(index : int) : void
-setICMajority(index : int) : double
-sumArray(tab : int[]) : int
-confToString(conf : int[]) : String
-hammingDst(a : int[], b : int[]) : int
+getResults() : String
+getResultsTag() : String
-arrayContains(tab : int[], x : int) : boolean
-setFaultedNodes(ind : TopologyIndividual) : void

```

```

ErX
+r : Random
+edgeTable : int[][]
+numEdges : int[]
+active : boolean[]

+ErX(r : Random)
-inserEdge(e : int, e : int) : void
-setup(I1 : PermutationIndividual, I2 : PermutationIndividual) : void
-getRandom() : int
-kill(n : int) : void
-next(n : int) : int
+execute(o : Object) : Object

```

```

DensityProblem
-nbIC : int
-lowerBound : double
-upperBound : double
-maxStepsTimesIndLen : int
-isSync : boolean
-maxSteps : int
-recordEval : boolean
-r : Random
-useRandomRuleIfTiedScore : boolean
-stopIfConvergence : boolean = true
-ics : int[]
-icsMajority : int[]
-evalNo : int = 0
-debug : boolean = false
-debug(s : String) : void
+DensityProblem(r : Random, nbIC : int, lowerBound : double, upperBound : double, ms : int, isSync : boolean, rndTiedRule : boolean, rec : boolean, stopIfConvergence : boolean)
+DensityProblem(r : Random, p : Params, learning : boolean)
+eval(individual : Individual) : double
-evaluateIC(ind : TopologyIndividual, icIndex : int) : int
#makeSyncStep(ind : TopologyIndividual, conf : int[]) : int[]
#makeNonSyncStep(ind : TopologyIndividual, conf : int[]) : int[]
-getNewValue(allele : int[], alleleDeg : int, conf : int[], alleleIndex : int) : int
-inItCs() : void
-generateRandomIC(index : int) : void
-setICMajority(index : int) : double
-sumArray(tab : int[]) : int
-confToString(conf : int[]) : String

```

```

IntegerIndividual
#alleles : int[]
#minAlleleValue : int
#maxAlleleValue : int
+IntegerIndividual()
+IntegerIndividual(len : int)
+IntegerIndividual(len : int, minValue : int, maxValue : int)
+getMinMaxAlleleValue(min : boolean) : int
+getAllele(locus : int) : Object
+getIntegerAllele(locus : int) : int
+setMinMaxAlleleValue(min : boolean, value : int) : void
+setAllele(locus : int, allele : Object) : void
+setIntegerAllele(locus : int, allele : int) : void
+mutate(r : Random, locus : int) : void
+setLength(len : int) : void
+setRandomValue(r : Random) : void
+copyIndividual(ind : Individual) : void
+toString() : String
+equals(obj : Object) : boolean
+clone() : Object

```

```

ScaleFreeIndividual
#avgDegree : int
#m0 : int
#m : int
#nbEdgesToAllocate : int
#offsetA : int
#moreVert : double
+ScaleFreeIndividual(p : Params, avgD : int, m0 : int, a : int)
#init() : void
+getK() : int
+main(args : String[]) : void

```

```

DynamicRulesIndividual
-rules : ArrayList[] = null
-debug : boolean = true
-params : Params
+main(args : String[]) : void
+toString() : String
+getNbRules(allele : int) : int
+DynamicRulesIndividual(p : Params)
-inItRules() : void
-resetRules() : void
-resetRules(n : int) : void
-setRandomRules() : void
+getRule(node : int, rule : int) : int
+addRule(node : int, value : int) : void
+setRule(node : int, rule : int, value : int) : void
+mutate(r : Random, l : int) : void
+mutateTopology(r : Random, locus : int) : void
-debug(s : String) : void
+clone() : Object

```

```

Params
-root : Element
+Params(xmlPath : String)
+main(args : String[]) : void
+getElement(parentEName : String, paramName : String) : Element
+intValue(paramName : String) : int
+intValue(parentEName : String, paramName : String) : int
+doubleValue(paramName : String) : double
+doubleValue(parentEName : String, paramName : String) : double
+stringValue(paramName : String) : String
+stringValue(parentEName : String, paramName : String) : String
+booleanValue(paramName : String) : boolean
+booleanValue(parentEName : String, paramName : String) : boolean

```

```

Individual
#len : int
#fitness : double
+Individual()
+Individual(len : int)
+getLength() : int
+setLength(len : int) : void
+clone() : Object
+getAllele(locus : int) : Object
+setAllele(locus : int, allele : Object) : void
+mutate(r : Random, locus : int) : void
+setRandomValue(r : Random) : void
+copyIndividual(ind : Individual) : void

```

```

Dpx
-r : Random
+Dpx(r : Random)
+execute(o : Object) : Object

```

```

ScaleFreeRewiredIndividual
-MAX_TRIES : int = 100
+ScaleFreeRewiredIndividual(p : Params, avgD : int, m0 : int, a : int, nbToRewire : int)
#rewire() : boolean
+main(args : String[]) : void

```

```

PopGrid
-dy : int
-dx : int
+PopGrid(dx : int, dy : int)
+getDimX() : int
+getDimY() : int
+getIndividual(x : int, y : int) : Individual
+getIndividual(p : Point) : Individual
+getFromPoints(pVector : Point[], iVector : Individual[]) : void
+setDimension(dx : int, dy : int) : void
+setIndividual(x : int, y : int, ind : Individual) : void
+setIndividual(p : Point, ind : Individual) : void
+toLinealX : int, y : int : int
+toLineal(p : Point) : int
+toGrid(pos : int) : Point

```

```

BinaryIndividual
#alleles : boolean[]
+BinaryIndividual()
+BinaryIndividual(len : int)
+getAllele(locus : int) : Object
+getBooleanAllele(locus : int) : boolean
+setAllele(locus : int, allele : Object) : void
+setBooleanAllele(locus : int, allele : boolean) : void
+mutate(r : Random, locus : int) : void
+setLength(len : int) : void
+setRandomValues(r : Random) : void
+copyIndividual(ind : Individual) : void
+binaryToDecimal(locus : int, len : int) : long
+toString() : String
+equals(obj : Object) : boolean
+clone() : Object

```

```

WattsSmallWorldIndividual
-avgDegree : int
-m0 : int
-m : int
-nbEdgesToAllocate : int
-offsetA : int
-moreVert : double
-initialPhi : double
-r : Random
+WattsSmallWorldIndividual(p : Params, initPhi : double)
-init() : void
+main(args : String []) : void

```

```

RecEvoI
-fitnesses : double[]
-perfs : double[]
-phis : double[]
-entropy : double[]
-genCpt : int
-inds : String == ""
+rec(genNum : int, bestFit : double, pop : Population, ent : double) : void
+recShort(genNum : int, bestFit : double, pop : Population, ent : double) : void

```

```

DynamicRulesDensityProblem
-debug : boolean = true
+DynamicRulesDensityProblem(r : Random, p : Params, learning : boolean)
#makeSyncStep(individual : TopologyIndividual, conf : int []) : int []
#makeNonSyncStep(ind : DynamicRulesIndividual, conf : int []) : int []
-getNewValue(ind : DynamicRulesIndividual, conf : int [], alleleIndex : int) : int
-debug(s : String) : void

```

```

Population
#population : Individual[]
#popSize : int
+Population(popSize : int)
+getIndividual(pos : int) : Individual
+setIndividual(pos : int, ind : Individual) : void
+setRandomPop(r : Random, ind : Individual) : void
+setRandomPop(r : Random, ind : Individual, maxFitness : double) : void
+copyPop(pop : Population) : void

```

```

Cx
-intv : int[]
-boolv : boolean[]
-r : Random
+Cx(r : Random)
+execute(o : Object) : Object

```

```

Pmx
-intv : int[]
-r : Random
+Pmx(r : Random)
+execute(o : Object) : Object

```

```

HillClimbing
-prob : Problem
-local : Operator
-steps : int
+HillClimbing(prob : Problem, local : Operator, steps : int)
+bestIndividual(iv : Individual []) : Individual
+execute(o : Object) : Object

```

```

Problem
-nEvals : int
+reset() : void
+getNEvals() : int
+evaluate(ind : Individual) : double
+evaluatePopulation(pop : Population) : void
+eval(ind : Individual) : double

```

```

TopologyMutation
-r : Random
-prob : double
+TopologyMutation(r : Random, prob : double)
+execute(o : Object) : Object

```

```

FixedNeigh
#moves : Point[]
#neigh : Point[]
#size : int
+FixedNeigh()
+FixedNeigh(size : int)
+lowSize(size : int) : void
+getNeighSize() : int
+getNeighbors(cell : Point) : Point []

```

```

NewRandomSweep
-pos : int
-sweep : PermutationIndividual
-r : Random
+NewRandomSweep(r : Random, pop : PopGrid)
+nextCell() : Point

```

```

FloatNonUniformMutation
-r : Random
-cea : CellularEA
+FloatNonUniformMutation(r : Random, cea : CellularEA)
+execute(o : Object) : Object

```

```

PermutationIndividual
+PermutationIndividual()
+PermutationIndividual(len : int)
+mutate(r : Random, locus : int) : void
+setRandomValues(r : Random) : void
+swap(i : int, j : int) : void
+inversion(i : int, j : int) : void
+relocate(i : int, j : int) : void

```

```

Ax
#pBias : double
+Ax()
+Ax(pBias : double)
+execute(o : Object) : Object

```

```

<<Interface>>
GenerationListener
+generation(cea : CellularEA) : void

```

```

Compact9
+Compact9()

```

```

FloatUniformMutation
-r : Random
-cea : CellularEA
+FloatUniformMutation(r : Random, cea : CellularEA)
+execute(o : Object) : Object

```

```

Px
-pBias : double
-r : Random
+Px(r : Random)
+Px(r : Random, pBias : double)
+execute(o : Object) : Object

```

```

Compact13
+Compact13()

```

```

TournamentSelection
-r : Random
+TournamentSelection(r : Random)
+execute(o : Object) : Object

```

```

NoisedSynchronizationProblemSimple
-nbIC : int
-faultProb : double
-lowerBound : double
-upperBound : double
-isSync : boolean
-maxSteps : int
-recordEval : boolean
-r : Random
-useRandomRuleIfTiedScore : boolean
-stopIfConvergence : boolean = true
-sumHD : int
-score : int
-scoreF : int
-sumSteps : int
-sumStepsF : int
-ics : int[]
-icsMajority : int[]
+NoisedSynchronizationProblemSimple(r : Random, fp : double, nbIC : int, lowerBound : double, upperBound : double, ms : int, isSync : boolean, rdTiedRule : boolean, rec : boolean, stopIfConvergence : boolean)
+evalIndividual(ind : Individual) : double
+evaluateCInd : TopologyIndividual, icIndex : int) : void
+makeSyncStep(ind : TopologyIndividual, conf : int [], fp : double) : int []
+makeNonSyncStep(ind : TopologyIndividual, conf : int [], fp : double) : int []
+getNewValue(allele : int [], alleleDeg : int, conf : int [], alleleIndex : int, fp : double) : int
-initICs() : void
-generateRandomIC(index : int) : void
-setICMajority(index : int) : double
-sumArray(tab : int []) : int
-confToString(conf : int []) : String
-hammingDist(a : int [], b : int []) : int

```

```

ComplexStats
+MAX_FIT_VALUE : int = 0
+MIN_FIT_VALUE : int = 1
+MAX_FIT_POS : int = 2
+MIN_FIT_POS : int = 3
+AVG_FIT : int = 4
+ENTROPY_GEN : int = 5
+VARIANCE_FIT : int = 6
+STD_DEV_FIT : int = 7
+PEARSON_FIT : int = 8
+LOG_2 : double = Math.log(2, 0)
-maxFitValue : double
-minFitValue : double
-avgFit : double
-entropyGen : double
-varianceFit : double
-stdDevFit : double
-pearsonFit : double
-maxFitPos : int
-minFitPos : int
-popAux : Population
+ComplexStats()
+log2(x : double) : double
+getStat(keyName : int) : Object
+calculate(pop : Population) : void
+toString() : String

```

```

RealIndividual
#alleles : double[]
#minAlleleValue : double
#maxAlleleValue : double
+RealIndividual()
+RealIndividual(len : int)
+RealIndividual(len : int, minVal : double, maxVal : double)
+getMinAlleleValue(min : boolean) : double
+getAllele(locus : int) : Object
+getRealAllele(locus : int) : double
+setMinAlleleValue(min : boolean, value : double) : void
+setAllele(locus : int, allele : Object) : void
+setRealAllele(locus : int, allele : double) : void
+mutate(r : Random, locus : int) : void
+nuMutate(r : Random, locus : int, generation : int, generationLimit : int) : void
+setLength(len : int) : void
+setRandomValues(r : Random) : void
+copyIndividual(ind : Individual) : void
+toString() : String
+equals(obj : Object) : boolean
+clone() : Object

```

```

DensityProblemPHI
-nbIC : int
-lowerBound : double
-upperBound : double
-maxStepsTimesIndLen : int
-isSync : boolean
-maxSteps : int
-recordEval : boolean
-r : Random
-useRandomRuleIfTiedScore : boolean
-phiWeight : double
-ics : int[]
-icsMajority : int[]
-evalNo : int = 0
+DensityProblemPHI(r : Random, p : Params, learning : boolean, w : double)
+DensityProblemPHI(r : Random, p : Params, learning : boolean)
+evalIndividual(ind : Individual) : double
+evaluateCInd : TopologyIndividual, icIndex : int) : int
+makeSyncStep(ind : TopologyIndividual, conf : int []) : int []
+makeNonSyncStep(ind : TopologyIndividual, conf : int []) : int []
+getNewValue(allele : int [], alleleDeg : int, conf : int [], alleleIndex : int) : int
-initICs() : void
-generateRandomIC(index : int) : void
-setICMajority(index : int) : double
-sumArray(tab : int []) : int
-confToString(conf : int []) : String

```

```

BiasedSynchronizationProblem
-nbIC : int
-lowerBound : double
-upperBound : double
-maxStepsTimesIndLen : int
-isSync : boolean
-maxSteps : int
-recordEval : boolean
-r : Random
-useRandomRuleIfTiedScore : boolean
-ics : int[]
-icsMajority : int[]
-evalNo : int = 0
-stopIfConvergence : boolean = true
-phiWeight : double
+BiasedSynchronizationProblem(r : Random, p : Params, learning : boolean, w : double)
+BiasedSynchronizationProblem(r : Random, p : Params, learning : boolean)
+evalIndividual(ind : Individual) : double
+evaluateCInd : TopologyIndividual, icIndex : int) : int
+makeSyncStep(ind : TopologyIndividual, conf : int []) : int []
+makeNonSyncStep(ind : TopologyIndividual, conf : int []) : int []
+getNewValue(allele : int [], alleleDeg : int, conf : int [], alleleIndex : int) : int
-initICs() : void
-generateRandomIC(index : int) : void
-setICMajority(index : int) : double
-sumArray(tab : int []) : int
-confToString(conf : int []) : String

```

PermanentlyFaultedDensityProblemSimple

```

-nbIC : int
-faultProb : double
-lowerBound : double
-upperBound : double
-isSync : boolean
-maxSteps : int
-recordEval : boolean
-r : Random
-useRandomRuleIfTiedScore : boolean
-stopIfConvergence : boolean = true
-steps : int
-sumHD : int
-score : int
-scoref : int
-sumSteps : int
-sumStepsF : int
-ics : int[]
-icsMajority : int[]
-faultStartFromStep : int
+PermanentlyFaultedDensityProblemSimple(r : Random, fp : double, nbIC : int, lowerBound : double, upperBound : double, ms : int, isSync : boolean, mdTiedRule : boolean, rec : boolean, stopIfConvergence : boolean, faultFrom : int)
+eval(individual : Individual) : double
+evaluateIC(ind : TopologyIndividual, icIndex : int) : void
+makeSyncStep(ind : TopologyIndividual, conf : int [], fp : double) : int []
+makeNonSyncStep(ind : TopologyIndividual, conf : int [], fp : double) : int []
+getNewValue(allele : int [], alleleDeg : int, conf : int [], alleleIndex : int, fp : double) : int
+initICs() : void
+generateRandomIC(index : int) : void
+setICmajority(index : int) : double
+sumArray(tab : int []) : int
+confToString(conf : int []) : String
+hammingDst(a : int [], b : int []) : int
    
```

SynchronizationProblem

```

-nbIC : int
-lowerBound : double
-upperBound : double
-maxStepsTimesIndLen : int
-isSync : boolean
-maxSteps : int
-recordEval : boolean
-r : Random
-useRandomRuleIfTiedScore : boolean
-ics : int[]
-icsMajority : int[]
-evalNo : int = 0
-stopIfConvergence : boolean = true
+SynchronizationProblem(r : Random, nbIC : int, lowerBound : double, upperBound : double, ms : int, isSync : boolean, mdTiedRule : boolean, rec : boolean, stopIfConvergence : boolean)
+SynchronizationProblem(r : Random, p : Params, learning : boolean)
+eval(individual : Individual) : double
+evaluateIC(ind : TopologyIndividual, icIndex : int) : int
+makeSyncStep(ind : TopologyIndividual, conf : int []) : int []
+makeNonSyncStep(ind : TopologyIndividual, conf : int []) : int []
+getNewValue(allele : int [], alleleDeg : int, conf : int [], alleleIndex : int) : int
+initICs() : void
+generateRandomIC(index : int) : void
+setICmajority(index : int) : double
+sumArray(tab : int []) : int
+confToString(conf : int []) : String
    
```

SelectionCEA
-maxFitness : double
+SelectionCEA(r : Random)
+getNumBest() : int
+experiment(generationLimit : int) : void

<<Interface>> Statistic
+calculate(pop : Population) : void
+getStat(keyName : int) : Object

UniformChoice
-r : Random
+UniformChoice(r : Random, pop : PopGrid)
+nextCell() : Point

Ox
-boolv : boolean[]
-r : Random
+Ox(r : Random)
+execute(o : Object) : Object

Spx
-r : Random
+Spx(r : Random)
+execute(o : Object) : Object

LineSweep
-pos : int
+LineSweep(pop : PopGrid)
+nextCell() : Point

DefaultMutation
-r : Random
+DefaultMutation(r : Random)
+execute(o : Object) : Object

LinearRankSelection
-r : Random
+LinearRankSelection(r : Random)
+execute(o : Object) : Object

CellUpdate
#pop : PopGrid
+CellUpdate(pop : PopGrid)
+nextCell() : Point

BestSelection
+execute(o : Object) : Object

SynchronousCEA
+SynchronousCEA(r : Random)
+experiment(generationLimit : int) : void

RouletteWheelSelection
-r : Random
+RouletteWheelSelection(r : Random)
+execute(o : Object) : Object

FixedRandomSweep
-pos : int
-sweep : PermutationIndividual
+FixedRandomSweep(pop : PopGrid)
+nextCell() : Point

<<Interface>> Neighborhood
+getNeighborCell : Point : Point []
+getNeighSize() : int

AsynchronousCEA
+AsynchronousCEA(r : Random)
+experiment(generationLimit : int) : void

<<Interface>> Operator
+execute(o : Object) : Object

ReplaceIfBetter
+execute(o : Object) : Object

ReplaceAlways
+execute(o : Object) : Object

CenterSelection
+execute(o : Object) : Object

Linear5
+Linear5(x : int, y : int)


```

RandomlyFaultedDensityProblem
--nbIC : int
--lowerBound : double
--upperBound : double
--maxStepsTimesIndLen : int
--isSync : boolean
--maxSteps : int
--recordEval : boolean
--r : Random
--useRandomRuleIfTiedScore : boolean
--stopIfConvergence : boolean = true
--synchPb : boolean
--ics : int[]
--icsMajority : int[]
--faultProb : double
--sumAvgHammingDst : double
--sumSteps : int
--sumStepsF : int
--sumScore : double
--sumScoreF : double
--nbEvals : int
--rightlyConverged : double[]
--wronglyConverged : double[]
--notConverged : double[]
--nbIndsPerConvergenceCategory : int[]
--valuesForStdDev : double[]
--sums : double[]
--debug : boolean = false
+RandomlyFaultedDensityProblem(r : Random, p : Params, learning : boolean, fp : double)
--debug(s : String) : void
+eval(individual : Individual) : double
+printStats() : void
--evaluateIC(ind : TopologyIndividual, icIndex : int) : void
#makeSyncStep(ind : TopologyIndividual, conf : int [], fp : double) : int []
#makeNonSyncStep(ind : TopologyIndividual, conf : int []) : int []
--getNewValue(allele : int [], alleleDeg : int, conf : int [], alleleIndex : int) : int
--initICs() : void
--generateRandomIC(index : int) : void
--setICmajority(index : int) : double
--sumArray(tab : int []) : int
--confToString(conf : int []) : String
--hammingDst(a : int [], b : int []) : int

```

```

EqRes
--eq1 : double
--eq2 : double
--a : int
--c : int
--kStart : int
--kEnd : int
--diff : double
+EqRes(s1 : double, s2 : double, a : int, c : int, ks : int, ke : int)
+compareTo(e : Object) : int
+isBetter(e : EqRes) : boolean
+isBest() : boolean
+toString() : String

```

```

EqRes
--eq1 : double
--eq2 : double
--a : int
--c : int
--kStart : int
--kEnd : int
--diff : double
+EqRes(s1 : double, s2 : double, a : int, c : int, ks : int, ke : int)
+compareTo(e : Object) : int
+isBetter(e : EqRes) : boolean
+isBest() : boolean
+toString() : String

```

```

ConfigurationModelIndividual
#gamma : double
#avgDegree : int
+a : int
+c : int
+kStart : int
+kEnd : int
+paramsSet : boolean = false
+init : boolean = false
--offset : int = 0
--nodesStomps : ArrayList = null
--MAX_TRIES : int = 100
+ConfigurationModelIndividual(p : Params, avgD : int, gamma : double)
#init() : boolean
--wire() : boolean
#rewire(start1 : int, end1 : int) : boolean
#sumArray(x : int []) : int
#getParams() : EqRes
+main(args : String []) : void

```

```

ConfigurationGraphIndividual
#gamma : double
#avgDegree : int
+a : int
+c : int
+kStart : int
+kEnd : int
+paramsSet : boolean = false
--MAX_TRIES : int = 100
+ConfigurationGraphIndividual(p : Params, avgD : int, gamma : double)
#init() : void
#rewire(start1 : int, end1 : int) : boolean
#IntArrayShuffle(x : int []) : int []
#getParams() : EqRes
+main(args : String []) : void

```

```

SimulatedAnnealing
--r : Random
--prob : Problem
--suc : Operator
--steps : int
--tmax : double
--tmin : double
--coolingRate : double
+SimulatedAnnealing(r : Random, prob : Problem, suc : Operator, steps : int, tmax : double, tmin : double, coolingRate : double)
+execute(o : Object) : Object

```

```

PermanentlyFaultedSynchronizationProblemSimple
--nbIC : int
--faultProb : double
--lowerBound : double
--upperBound : double
--isSync : boolean
--maxSteps : int
--recordEval : boolean
--r : Random
--useRandomRuleIfTiedScore : boolean
--stopIfConvergence : boolean = true
--sumHD : int
--score : int
--scoreF : int
--sumSteps : int
--sumStepsF : int
--ics : int[]
--icsMajority : int[]
--steps : int
--faultStartsFromStep : int
+PermanentlyFaultedSynchronizationProblemSimple(r : Random, fp : double, nbIC : int, lowerBound : double, upperBound : double, ms : int, isSync : boolean, rndTiedRule : boolean, rec : boolean, stopIfConvergence : boolean, faultFrom : int)
+eval(individual : Individual) : double
--evaluateIC(ind : TopologyIndividual, icIndex : int) : void
--makeSyncStep(ind : TopologyIndividual, conf : int [], fp : double) : int []
--makeNonSyncStep(ind : TopologyIndividual, conf : int [], fp : double) : int []
--getNewValue(allele : int [], alleleDeg : int, conf : int [], alleleIndex : int, fp : double) : int
--initICs() : void
--generateRandomIC(index : int) : void
--setICmajority(index : int) : double
--sumArray(tab : int []) : int
--confToString(conf : int []) : String
--hammingDst(a : int [], b : int []) : int

```

```

NoisedSynchronizationProblem
--nbIC : int
--faultProb : double
--lowerBound : double
--upperBound : double
--maxStepsTimesIndLen : int
--isSync : boolean
--maxSteps : int
--recordEval : boolean
--r : Random
--useRandomRuleIfTiedScore : boolean
--stopIfConvergence : boolean = true
--sumHD : int
--score : int
--scoreF : int
--sumSteps : int
--sumStepsF : int
--ics : int[]
--icsMajority : int[]
+NoisedSynchronizationProblem(r : Random, fp : double, nbIC : int, lowerBound : double, upperBound : double, ms : int, isSync : boolean, rndTiedRule : boolean, rec : boolean, stopIfConvergence : boolean)
+eval(individual : Individual) : double
--evaluateIC(ind : TopologyIndividual, icIndex : int) : void
--makeSyncStep(ind : TopologyIndividual, conf : int [], fp : double) : int []
--makeNonSyncStep(ind : TopologyIndividual, conf : int [], fp : double) : int []
--getNewValue(allele : int [], alleleDeg : int, conf : int [], alleleIndex : int, fp : double) : int
--initICs() : void
--generateRandomIC(index : int) : void
--setICmajority(index : int) : double
--sumArray(tab : int []) : int
--confToString(conf : int []) : String
--hammingDst(a : int [], b : int []) : int

```

```

CellularEA
+PARAM POPULATION : int = 0
+PARAM STATISTIC : int = 1
+PARAM NEIGHBOURHOOD : int = 2
+PARAM CELL_UPDATE : int = 3
+PARAM LISTENER : int = 4
+PARAM PROBLEM : int = 5
+PARAM MUTATION_PROB : int = 6
+PARAM CROSSOVER_PROB : int = 7
+PARAM LOCAL_SEARCH_PROB : int = 8
+PARAM TARGET_FITNESS : int = 9
+PARAM GENERATION_NUMBER : int = 10
+PARAM SELECTED_CELL : int = 11
+PARAM GENERATION_LIMIT : int = 12
+PARAM NEWS : int = 21
#population : PopGrid
#statistic : Statistic
#neighbourhood : Neighbourhood
#cellUpdate : CellUpdate
#listener : GenerationListener
#problem : Problem
#selectedCell : Point
#mutationProb : double
#crossOverProb : double
#localSearchProb : double
#targetFitness : double
#generationNumber : int
#generationLimit : int
#operators : Map
#r : Random
#neighType : int = 0
+CellularEA(r : Random)
+getParam(keyValue : int) : Object
+getParam(keyName : String) : Object
+setParam(keyValue : int, param : Object) : void
+setParam(keyName : String, param : Object) : void
+experiment(generationLimit : int) : void

```

```

NoisedDensityProblem
-nbIC : int
-faultProb : double
-lowerBound : double
-upperBound : double
-maxStepsTimesIndLen : int
-isSync : boolean
-maxSteps : int
-recordEval : boolean
-r : Random
-useRandomRuleIfTiedScore : boolean
-stopIfConvergence : boolean = true
-sumHD : int
-score : int
-scoreF : int
-sumSteps : int
-sumStepsF : int
-ics : int[]
-icsMajority : int[]
-fitness : double = 0
-fitnessF : double = 0
-avgHD : double = 0
-avgSteps : double = 0
-avgStepsF : double = 0
-evalCounter : int = 0
+NoisedDensityProblem(r : Random, fp : double, nbIC : int, lowerBound : double, upperBound : double, ms : int, isSync : boolean, rndTiedRule : boolean, rec : boolean, stopIfConvergence : boolean)
+NoisedDensityProblem(r : Random, p : Params, learning : boolean, fp : double)
+eval(individual : Individual) : double
+evalAtC(ind : TopologyIndividual, icIndex : int) : void
+makeSyncStep(ind : TopologyIndividual, conf : int [], fp : double) : int []
+makeNonSyncStep(ind : TopologyIndividual, conf : int [], fp : double) : int []
+getNewValue(allele : int [], alleleDeg : int, conf : int [], alleleIndex : int, fp : double) : int
-inITCs() : void
+generateRandomIC(index : int) : void
+setICmajority(index : int) : double
-sumArray(tab : int []) : int
+confToString(conf : int []) : String
-hammingDst(a : int [], b : int []) : int
+getResults() : String

```

```

NoisedDensityProblemSimple
-nbIC : int
-faultProb : double
-lowerBound : double
-upperBound : double
-isSync : boolean
-maxSteps : int
-recordEval : boolean
-r : Random
-useRandomRuleIfTiedScore : boolean
-stopIfConvergence : boolean = true
-sumHD : int
-score : int
-scoreF : int
-sumSteps : int
-sumStepsF : int
-ics : int[]
-icsMajority : int[]
+NoisedDensityProblemSimple(r : Random, fp : double, nbIC : int, lowerBound : double, upperBound : double, ms : int, isSync : boolean, rndTiedRule : boolean, rec : boolean, stopIfConvergence : boolean)
+eval(individual : Individual) : double
+evalAtC(ind : TopologyIndividual, icIndex : int) : void
+makeSyncStep(ind : TopologyIndividual, conf : int [], fp : double) : int []
+makeNonSyncStep(ind : TopologyIndividual, conf : int [], fp : double) : int []
+getNewValue(allele : int [], alleleDeg : int, conf : int [], alleleIndex : int, fp : double) : int
-inITCs() : void
+generateRandomIC(index : int) : void
+setICmajority(index : int) : double
-sumArray(tab : int []) : int
+confToString(conf : int []) : String
-hammingDst(a : int [], b : int []) : int

```

```

SimpleStats
+MAX_FIT_VALUE : int = 0
+MIN_FIT_VALUE : int = 1
+MAX_FIT_POS : int = 2
+MIN_FIT_POS : int = 3
+AVG_FIT : int = 4
-minFitValue : double
-maxFitValue : double
-avgFit : double
-maxFitPos : int
-minFitPos : int
+getStatKeyName : int : Object
+calculate(pop : Population) : void
+toString() : String

```

```

RecordInfo
-record : String[] = new String[5]
+rec() : void
+rec(s : String, i : int) : void
+rec(s : String) : void
+print() : void
+print(i : int) : void
+reset() : void

```

Appendix C

UML Class Diagram - RBN

This annex contains the UML class diagram of the main RBN simulator.

DegreeDistribution
+nodesPerDegree : int[] -degree : double -exponent : double -expOriginal : double +type : DistributionType -nbStomps : int = -1 -minDeg : int -maxDeg : int -netSize : int -stringType : String -poissonDistributions : Map<String, int[]> = new TreeMap<String, int[]>() -powerLawDistributions : Map<String, int[]> = new TreeMap<String, int[]>() -queries : MySQLdbManager -db_host : String = "isi13.unil.ch:8889" -db_name : String = "cdarabos" -db_user : String = "cdarabos" -db_pwd : String = "yryr00" -noDB : boolean
+DegreeDistribution(t : String, g : double, net : Network, d : double) +DegreeDistribution(t : String, netSize : int, g : double, d : double) +DegreeDistribution(t : String, gamma : double, avgDegree : double, netSize : int, minDeg : int, maxDeg : int) +DegreeDistribution(t : String, n : Network, d : double) +makeNewDistribution() : void +toString() : String +getType() : String +meanDegree() : double +maxDegree() : int -getDistributionFromDB() : int [] -makeClassicalPLDistribution() : int [] -makePowerLawDistribution() : int [] -sumBelowCurve(a : double, b : double) : PowerLawSolution -makePoissonDistribution() : int [] -makeNormalDistribution() : int [] +main(args : String []) : void

<<Enum>> DistributionType
<<Constant>> -CLASSICAL_POWERLAW = 0 <<Constant>> -POWERLAW = 1 <<Constant>> -NORMAL = 2 <<Constant>> -FIXED = 3 <<Constant>> -POISSON = 4 <<Constant>> -NONE = 100 -type : int -value : double -DistributionType(type : int) +getType() : int +setValue(value : double) : void +getValue() : double +asString(i : int) : String

Utils
-rnd : Random = null -Utils() +longToBooleanArray(l : long) : Boolean [] +booleanArrayToLong(values : boolean []) : long +bitSetToLong(bs : BitSet) : long +rnd() : Random +round(d : double, z : int) : double +average(list : List) : double +standardDeviation(list : List) : double +median(list : List) : double +countTrue(list : List<Boolean>) : int +factorial(n : int) : int +uniqueID() : String +main(args : String []) : void +shuffle(tab : T []) : void +shuffle(tab : int []) : void +ToFile(s : String, path : String) : void +serialize(o : Object) : byte [] +unserialize(byteObject : byte []) : Object

PowerLawSolution
+a : double +b : double +nodes : int +stomps : int +deltaNodes : int +deltaStomps : int +above : boolean +PowerLawSolution(a : double, b : double, s : int, n : int, ds : int, dn : int, abv : boolean) +toString() : String +compareTo(p : PowerLawSolution) : int

FitnessLandscapeNetwork
+FitnessLandscapeNetwork(size : int, net : FLNode []) +sortByFitness() : FitnessLandscapeVertex [] +sortByDegree() : FitnessLandscapeVertex [] -sort() : FitnessLandscapeVertex [] +currentState() : Object +currentStateToString() : String +removeAll(c : Collection<?>) : boolean +setAllToRandomValues() : void +main(args : String []) : void +clone(cn : String ...) : FitnessLandscapeNetwork

Edge
-from : Vertex -to : Vertex +Edge(from : Vertex, to : Vertex) +equals(o : Object) : boolean

SqlQueries
-DB_NAME : String = "temp" +main(args : String []) : void

OutPut
+hammingDistance(out : OutPut) : int

RunInfo
-runId : String +RunInfo() +add(t : String, n : String, v : String) : void +add(t : String, map : Map) : void +add(t : String, o : Object) : void +save(tableName : String) : void

FLNode
+id : int +fitness : double +basin : int +neigh : List<Integer> +FLNode(id : int)

ActivatedWaterfallUpdate
+ActivatedWaterfallUpdate(n : Network) #makeUpdate(v : Vertex) : List<Vertex>

StandardWaterfallUpdate
+StandardWaterfallUpdate(n : Network) #makeUpdate(v : Vertex) : List<Vertex>

RandomlyInitializedBinaryLookupTable
-rnd : Random -probability : double +RandomlyInitializedBinaryLookupTable(p : double) +RandomlyInitializedBinaryLookupTable() +getRule(key : BitSet) : Boolean +newInstance() : RandomlyInitializedBinaryLookupTable

BoxPlotParse
+main(args : String []) : void ~boxStats(l : ArrayList<Double>) : String ~getTag(ary : String []) : String

SynchronousUpdate
~verticesToBeUpdated : List<Vertex> +SynchronousUpdate(n : Network) +step(pos : int ...) : List<Vertex>

BucketParse
+main(args : String []) : void ~getBucket(x : int, bs : int) : String ~getTag(ary : String []) : String

NetworkFactory
+build(type : String, size : int, v : Vertex) : Network

VertexFactory
+build(type : String, rs : RuleSet) : Vertex

UpdateSchemeFactory
+build(type : String, net : Network) : UpdateScheme

Main
-MIN_DEGREE : int -DEFAULT_MIN_DEGREE : int = 1 -MAX_DEGREE : int -DESIRED_DEGREE : double -REPETITIONS : int -DEFAULT_REPETITIONS : int = 20 -NB_ICS : int -DEFAULT_NB_ICS : int = 500 -TOPOLOGIES : int -DEFAULT_TOPOLOGIES : int = 1 -NETWORK_SIZE : int -GAMMA_OUT : double -GAMMA_IN : double -ATTRACTOR_MAX_SIZE : int -DEFAULT_ATTRACTOR_MAX_SIZE : int = -1 -ATTRACTOR_MAX_NB : int -DEFAULT_ATTRACTOR_MAX_NB : int = -1 -ATTRACTOR_INIT_STEPS : int -ATTRACTOR_STEPS : int -DEFAULT_ATTRACTOR_STEPS : int = 1000 -NB_AVALANCHES : int -DEFAULT_NB_AVALANCHES : int = 0 -UPDATE : String -DEFAULT_UPDATE : String = "Active Waterfall" -NET_TYPE : String -DEFAULT_NET_TYPE : String = "Boolean" -REMARK : String -IN_DISTR : String -OUT_DISTR : String -DEFAULT_IN_DISTR : String = "normal" -DEFAULT_OUT_DISTR : String = "aldana" -db_host : String = "isi13.unil.ch:8889" -db_name : String = "rbn" -db_user : String = "cdarabos" -db_pwd : String = "yryr00" +Main(args : String []) +main(args : String []) : void

ScaleFreeNetworkGenerator
-MAX_RUN : int = 5 -netSize : int -minDegree : int -maxDegree : int -gammaOut : double -rnd : Random -paramsSet : boolean = false -outDegrees : int[] -ScaleFreeNetworkGenerator(net : Network, g : double) -connectNormal(net : Network) : void -connectRnd(net : Network) : void +fill(net : Network, g : double, type : String) : void +fill(net : Network, g : double, type : String, avgDegree : int) : void +fill(net : Network, gOut : double, gIn : double, type : String, avgDegree : int) : void -connect(net : Network, inDegrees : int []) : void -generateMcmInDistribution(avgD : int, gammaIn : double) : int [] -connect(net : Network, type : String) : void +generateOutDistribution() : void +generateMcmOutDistribution(avgD : int) : void #getPowerLaw_A_Param(avgDegree : int, gamma : double) : int #init(avgDegree : int, a : int, gamma : double) : int [] +main(args : String []) : void

UpdateScheme
#net : Network +UpdateScheme(n : Network) +setNetwork(n : Network) : void +reset(index : int ...) : void +step(pos : int ...) : List<Vertex> +getRecentVerticesEvo() : int +resetRecentVerticesEvo() : void +setVerticesToBeUpdated(lv : List<Vertex>) : void +getVerticesToBeUpdated() : List<Vertex>

WaterfallUpdate
~rnd : Random ~recentVerticesEvo : int = 0 ~verticesToBeUpdated : List<Vertex> #WaterfallUpdate(n : Network) +reset(pos : int ...) : void #makeUpdate(v : Vertex) : List<Vertex> +step(pos : int ...) : List<Vertex> +resetRecentVerticesEvo() : void

AttractorsList
-attractorMaxLength : int -maxAttractors : int -attractorPerLength : Map<Integer, Integer> -nbRealizationsWithKattractors : Map<Integer, Integer> -nbAttractors : int -nbAttractorsBelowMaxLength : int -attractorsAverageLength : double -attractorsAverageLengthBelowMaxLength : double +AttractorsList(maxAtt : int, attMaxLength : int) +containsState(state : Object) : Attractor +containsAttractor(needle : Attractor) : boolean +getAttractor(needle : Attractor) : Attractor +recordStats() : void +nbRealizationsWithKattractors() : Map<Integer, Integer> +attractorPerLength() : Map<Integer, Integer>

BooleanNetwork
~rnd : Random +BooleanNetwork(nbVertices : int, v : Vertex) +setAllToTrue() : void +setAllToFalse() : void +setAllToRandomValues() : void +toBitSet() : BitSet +getVertex(index : int) : BooleanVertex +currentState() : BitSet +setCurrentState(bs : BitSet) : void +currentStateToString() : String +generateAllCombinations() : void +nextState(bs : BitSet, current : int) : void +clone() : BooleanNetwork +readInTopology(net : BooleanNetwork) : void

Attractor
-maxLength : int -usage : int -cyclic : boolean -id : String +Attractor(l : int) +Attractor() +add(state : S) : boolean +length() : int +incrementUsage() : void +isCyclic() : boolean +isCyclicBinary() : int +removeRange(a : int, b : int) : void

BooleanVertex
+BooleanVertex(synchUpdate : boolean, r : RuleSet<BitSet, Boolean>, ud : boolean ...) +BooleanVertex(r : RuleSet<BitSet, Boolean>, ud : boolean ...) +BooleanVertex(r : RuleSet<BitSet, Boolean>) +setRandomValue(rnd : Random) : void +newInstance() : BooleanVertex +generateKey() : BitSet

RuleSet
-nbRuleSets : int = 0 -id : int +RuleSet() +getRule(key : K) : T +hasRule(key : K) : boolean +setRule(key : K, value : T) : void +newInstance() : RuleSet<K, T>

FitnessLandscapeVertex
+compareByFitness : boolean -basinSize : int +FitnessLandscapeVertex() +newInstance() : Vertex +generateKey() : BitSet +compareTo(o : Object) : int +toString() : String

MisbehavingBooleanNetwork
-misbehavingVertices : List<BooleanVertex> +MisbehavingBooleanNetwork(nbVertices : int, v : Vertex) +flipOne() : int +countMisbehaved() : int +compareBehavior(net : Network) : void

RunInfoItem
~type : String ~name : String ~value : String +RunInfoItem(t : String, n : String, v : String) +toSingleQuoteString() : String

NetworkProperties
-dstMatrix : int[][] +NetworkProperties(net : Network) +cpl() : double +diameter() : int +radius() : int

Vertex
<pre> -vertexCount : int = 0 #synchronousUpdate : boolean = true -updated : boolean -outgoingEdges : List<Vertex> -incomingEdges : List<Vertex> -maxOutgoingEdges : int = -1 -minOutgoingEdges : int = -1 -id : int -distances : HashMap<Vertex, Integer> #value : VALUE_TYPE #tmpValue : VALUE_TYPE #rules : RuleSet<RULESET_KEY, VALUE_TYPE> -undirected : boolean = false -globalMinDst : int #Vertex(synchUpdate : boolean, r : RuleSet<RULESET_KEY, VALUE_TYPE>, ud : boolean ...) +newInstance() : Vertex +setUpdateSynchronicity(b : boolean) : void +compareTo(o : Object) : int +hasMaxOutgoingEdges() : boolean +hasMinOutgoingEdges() : boolean +outDegree() : int +inDegree() : int +addIncominEdge(v : Vertex) : void -removelIncomingEdge(v : Vertex) : void +addOutgoingEdge(n : Vertex) : boolean #getOutgoingEdge(pos : int) : Vertex #removeOutgoingEdge(pos : int) : Vertex +removeOutgoingEdge(v : Vertex) : boolean +hasOutgoingEdgeTo(o : Vertex) : boolean +indexOfOutgoingEdgeTo(o : Vertex) : int +hasIncomingEdgeTo(o : Vertex) : boolean +clearAllEdges() : void +generateKey() : RULESET_KEY +update() : Object +finalizeSynchronousUpdate() : void +toString() : String +clearRules() : void +clusteringCoefficient() : double +resetDistances() : int +setDistanceTo(v : Vertex, dst : int) : int +getDistanceTo(v : Vertex) : int +isSetDistanceTo(v : Vertex) : boolean +modified() : void +eccentricity() : int +getNeighborsAverageOutDegree() : double #undirectedOnly() : void +addEdge(v : Vertex) : void +removeEdge(v : Vertex) : boolean +degree() : int +resetVertexId() : void </pre>

Network
<pre> +IN : int = 0 +OUT : int = 1 #undirected : boolean = false +modified : boolean -inDegreeDistribution : DegreeDistribution = null -outDegreeDistribution : DegreeDistribution = null +Network(nbVertices : int, vertex : V) +setDegreeDistributions(ddin : DegreeDistribution, ddout : DegreeDistribution) : void +shuffleVertices() : void +disconnectAll() : void +connectAll() : void +connect() : void +averageOutDegree() : double +averageInDegree() : double +maxOutDegree() : int +maxInDegree() : int +hammingDistance(net : Network) : double +printStats(show : boolean) : String +setDistances() : void +toString() : String +size() : int +setAllVerticesTo(value : T) : void +clearAllRules() : void +getVertex(index : int) : V +getVertexWithId(id : int) : V +getMinOutDegree() : int +getMaxOutDegree() : int +setUpdateSynchronicity(b : boolean) : void +setAllToRandomValues() : void +currentState() : Object +currentStateToString() : String +clusteringCoefficient() : double +radius() : int +diameter() : int +cpl() : double +assortativity() : double +getEdges() : List<Edge> +compare(net : Network) : void #undirectedOnly() : void +averageDegree() : double +toGraphML(fout : File) : void </pre>

Appendix D

UML Class Diagram - eaRBN

This annex contains the UML class diagram of the secondary RBN simulator dependent on RBN.

```

StringLengthComparator
+compare(o1 : String, o2 : String) : int

```

```

FileConverter
+main(args : String []) : void

```

```

RunMakeNetwork
-PATH : String = "/Users/cdarabos/Desktop/landscapes/"
-ACU_SIZE : int = 6
+main(args : String []) : void

```

```

RealLifeUpdateFunction
-fcn : String
+RealLifeUpdateFunction()
+RealLifeUpdateFunction(f : String)
+copy() : UpdateFunction
+getNewValue(target : Vertex) : boolean
+getThreshold() : double
+main(args : String []) : void
+toString() : String
+type() : int

```

```

SynchronousUpdateTiming
+SynchronousUpdateTiming()
+update(net : Network) : void

```

```

PromoterMajorityUpdateFunction
-threshold : double
+PromoterMajorityUpdateFunction()
+PromoterMajorityUpdateFunction(t : double)
+getNewValue(target : Vertex) : boolean
+copy() : UpdateFunction
+toString() : String

```

```

GlobalActivatedCascadeUpdateTiming
~updatedVertices : ArrayList<Vertex> = null
+GlobalActivatedCascadeUpdateTiming()
+update(net : Network) : void

```

```

<<Interface>>
UpdateFunction
+RANDOM : int = 0
+MAJORITY : int = 1
+REALLIFE : int = 2
+getNewValue(target : Vertex) : boolean
+copy() : UpdateFunction
+type() : int
+getThreshold() : double
+toString() : String
+get(i : long) : Boolean

```

```

RunCompareRules
+main(args : String []) : void
~functionToBitSet(net : Network, bss : BitSet ...) : BitSet

```

```

Edge
-startVertexID : int
-endVertexID : int
-promoter : boolean
+Edge(start : int, end : int, p : boolean)
+Edge(start : Vertex, end : Vertex)
+compareTo(e : Edge) : int
+isLoop() : boolean
+getStartVertexID() : int
+isPromoter() : boolean
+clone() : Edge

```

```

StateCoordinates
+x : int
+y : int
+i : int
+color : Color
+StateCoordinates(x : int, y : int, color : Color)
+StateCoordinates(x : int, y : int, color : Color, i : int)

```

```

Chromosome
-netSize : int
-Chromosome(size : int)
+Chromosome(vertices : Vertex [])
+clone() : Chromosome

```

```

RunDefineLandscape
-PAJEK_FILE : String = "/Users/cdarabos/projects/earBN/netdata/stemCell.pajek"
-REMARK : String = "stemCell"
#db_host : String = "tiburtna.unil.ch"
#db_name : String = "earbn"
#db_user : String = "cdarabos"
#db_pwd : String = "grid-db"
#db_test : String = "earbn_test"
+main(args : String []) : void
+mainOpt(args : String []) : void
+getNetwork(size : int, outDD : DegreeDistribution, inDD : DegreeDistribution, name : String) : Network_old
+getNetwork(path : String) : Network_old
+exhaustive(net : Network, timing : UpdateTiming) : Landscape
+exhaustive(net : Network, timing : UpdateTiming, stats : boolean) : Landscape
+exhaustive(net : Network, timing : UpdateTiming, nbChunks : int, chunk : int) : Landscape
+sample(net : Network, timing : UpdateTiming, sampleSize : int, initSteps : int, attMaxLength : int) : Landscape

```

```

ActivatedCascadeUpdateTiming
-nextStep : Set<Vertex> = null
+ActivatedCascadeUpdateTiming()
+update(net : Network) : void

```

```

Basin
-attractor : ArrayList<State>
-gardensOfEden : ArrayList<State>
-id : int
-mean_transient : double
-sd_transient : double
-goe_prob : double
-mean_branch_len : double
-sd_branch_len : double
-med_transient : int
-med_branch_len : int
+Basin()
+Basin(b : Basin)
#setID(id : int) : void
+identifyAttractor(tmpList : ArrayList<State>) : void
+identifyGardensOfEden() : void
+nbGardensOfEden() : int
+getAttractorCycleLength() : int
+setStatesTransients() : void
+setStateTransient(current : State) : int
+clone() : Object
+makeClone(id : int) : Basin
+getName() : String
+toString() : String
+toString(delimiter : String) : String
+toDot() : String
+toDotSubNet() : String
+toGDLSubnet() : String
+computeStats() : void
+store(db : JdbcDbManager, realID : long) : long
+toGraphMLSubgraph() : String
+plot(fw : int, fh : int) : void
+plot(fw : int, fh : int, title : String) : void

```

```

Vertex
-nbVertices : int = 0
-name : String
-id : int
-value : boolean
-tmpValue : boolean
-updated : boolean
-initialValue : Boolean = null
-updateFunction : UpdateFunction
-outgoingEdgesTo : Set<Vertex>
-incomingEdgesFrom : TreeMap<Vertex, Boolean>
+Vertex(id : int, name : String)
+Vertex(id : int)
+randomValue() : void
+connect(v : Vertex, promoter : boolean) : boolean
+disconnect(v : Vertex) : void
#disconnectIncomingEdgeFrom(v : Vertex) : void
+disconnectAll() : void
#addIncomingEdgeFrom(v : Vertex, promoter : boolean) : boolean
+hasOutgoingEdgeTo(v : Vertex) : boolean
+hasIncomingEdgeFrom(v : Vertex) : boolean
+compareTo(v : Vertex) : int
+getInDegree() : int
+getOutDegree() : int
+isEnhancedBy(v : Vertex) : boolean
+isEnhancerOf(v : Vertex) : boolean
+setEnhancerOf(target : Vertex, enhancer : boolean) : void
+setEnhancedBy(source : Vertex, enhancer : boolean) : void
+isActive() : boolean
+updateImmediately() : void
+updateImmediately(uf : UpdateFunction) : void
+updateSynchronous() : void
+updateSynchronous(uf : UpdateFunction) : void
+finalizeUpdate() : void
+getFormattedName() : String
+toString() : String
+setInitialValue(initialValue : boolean) : void
+setToInitialValue() : void

```

```

RunDefineLandscapeByChunks
-db_host : String = "tiburtna.unil.ch"
-db_name : String = "earbn"
-db_user : String = "cdarabos"
-db_pwd : String = "grid-db"
+main(args : String []) : void

```

```

RandomUpdateFunction
-threshold : double
+RandomUpdateFunction(t : double)
+RandomUpdateFunction()
+toString() : String
+getNewValue(target : Vertex) : boolean
+getNewValue(ruleID : int) : boolean
+copy() : UpdateFunction
+typeAsString() : String

```

```

RunFromDB
#net : Network
#land : Landscape
#timing : UpdateTiming
-db_host : String = "isi24.unil.ch:8889"
-db_name : String = "earBN"
-db_user : String = "cdarabos"
-db_pwd : String = "yryr00"
+main(args : String []) : void
+RunFromDB(args : String [])

```

```

RunDerrida
+RunDerrida(args : String [])
+main(args : String []) : void

```


Landscape
<pre> -basins : ArrayList<Basin> -id : long -stats : boolean -meanBasinSize : double -stdDevBasinSize : double -meanAttractorCycleLength : double -stdDevAttractorCycleLength : double -meanTransientSteps : double -stdDevTransientSteps : double -meanTransientBranchLength : double -stdDevTransientBranchLength : double -gardenOfEdenProbability : double -basinEntropy : double -meanInDegree : double -stdDevInDegree : double -hd1probability : double -medianBasinSize : int -medianAttractorCycleLength : int -medianTransientSteps : int -medianTransientBranchLength : int -medianInDegree : int </pre>
<pre> +Landscape(size : int) +Landscape() +add(o : State) : boolean +netSize() : int +add(from : State, to : State) : boolean +resetBasins() : void +identifyBasins() : void +identifyBasins(stats : boolean) : void +getBasinContaining(s : State) : Basin +toString() : String +computeStats() : void +toString(delimiter : String) : String +toDot() : String +toDotSubNet() : String +toGDL() : String +toGDLsubnet() : String +getRandomState() : State +getRandomBasin() : Basin +store(db : JdbcDbManager, netID : long, realID : long, nbICs : int, update : int, timing : int, storeObject : boolean) : long +getFromDB(db : JdbcDbManager, id : long) : Landscape +toGraphML() : String +toGraphMLsubgraph() : String +derridaToString() : String +derrida() : double [] +branchingDistribution() : double [] +plotInSparateFrames(fw : int, fh : int) : void +plotInSparateFrames(fw : int, fh : int, title : String) : void +plot(title : String) : void </pre>

LandscapeFrame
<pre> -NODE_RADIUS : int = 10 -DELTA_RADIUS : int = 30 -FRAME_WIDTH : int = 1600 -FRAME_HEIGHT : int = 1100 -nbColumns : int -nbRows : int -coords : ArrayList<StateCoordinates> = new ArrayList<StateCoordinates>() -arrows : ArrayList<int[]> = new ArrayList<int[]>() </pre>
<pre> +LandscapeFrame(basins : ArrayList<Basin>, title : String) -translateX(x : int, i : int) : int -rotateY(y : int, i : int) : int +paint(g : Graphics) : void </pre>

Factory
<pre> +makeUpdateTiming(type : int) : UpdateTiming +makeUpdateFunction(type : double ...) : UpdateFunction </pre>

AttractorFrame
<pre> -NODE_RADIUS : int = 10 -DELTA_RADIUS : int = 30 -NODE_COLOR : Color = Color.YELLOW -ATT_COLOR : Color = Color.RED -GOE_COLOR : Color = Color.GREEN -coords : ArrayList<StateCoordinates> = new ArrayList<StateCoordinates>() -arrows : ArrayList<int[]> = new ArrayList<int[]>() </pre>
<pre> +AttractorFrame(b : Basin, title : String, fw : int, fh : int, visible : boolean) -findNodes(b : Basin, depth : int, from : State, fromX : int, fromY : int, deltaAngle : double, arc : double) : void +paint(g : Graphics) : void +getStateCoordinates() : ArrayList<StateCoordinates> </pre>

UpdateCommon
<pre> #TYPE : int #TYPE_STRING : String </pre>
<pre> +UpdateCommon() +UpdateCommon(t : int, s : String) +type() : int +toString() : String #neighborsValuesToBitSet(target : Vertex) : BitSet +get(longValue : long) : Boolean #get(v : Vertex) : Boolean #put(v : Vertex, b : boolean) : Boolean #neighborsValuesToLong(target : Vertex) : long </pre>

```

Network
-nbNetworks : int = 0
-id : int
-name : String
-remark : String
-inDDtype : DistributionType
-outDDtype : DistributionType
-verticesToBeUpdated : ArrayList<Vertex>

+Network(filePath : String, netSize : int, name : String, fileType : int)
+parseBooleanFctFile(source : BufferedReader) : void
+parsePajek(source : BufferedReader) : void
+Network(size : int, name : String)
+Network(size : int)
+Network(c : Chromosome)
+Network(size : int, outDD : DegreeDistribution, inDD : DegreeDistribution, name : String)
+getInDDtype() : DistributionType
+getOutDDtype() : DistributionType
+connectAtRandom(outDD : DegreeDistribution, inDD : DegreeDistribution) : void
+disconnectAll() : void
+getState() : State
+getState(state : State) : State
+setState(value : long) : State
+setState(value : long, state : State) : State
+setState(value : long, ul : long, state : State) : State
+setState(state : State) : void
+setRandomVerticesValues() : void
+setUpdateFunction(uf : UpdateFunction) : void
+computeEnhancersPorportion() : double
+toString() : String
+toString(separator : String) : String
+getDegreeDistribution(dir : DD_DIRECTION) : int []
+toDot() : String
+toDot(land : Landscape) : String
+toDotSubNet() : String
+averageDegree() : double
+main(args : String []) : void
+toTGF() : String
+setEnhancers(enhancerProb : double) : void
+store(db : JdbcDbManager, rem : String) : long
+alreadyStored(db : JdbcDbManager) : long
+getFromDB(db : JdbcDbManager, netid : long) : Network
+storeRealization(db : JdbcDbManager, threshold : double, enhancerProportion : double) : long
+searchName(name : String) : Vertex
+getVertexByName(name : String) : Vertex
+clearVerticesToBeUpdated() : void
+setVerticesToBeUpdated(vtbu : Collection<Vertex>) : void
+setVerticesToBeUpdated(value : long) : void
+getVerticesToBeUpdateAsLong() : long
+toGraphML() : String
+toGraphMLsubgraph() : String
+isConnexe() : boolean

```

```

<<Enum>>
DD_DIRECTION
<<Constant>> -IN
<<Constant>> -OUT

```

```

State
#nextState : State
#finalized : boolean
#id : long
#updateList : long
#maxSize : int
#transientLength : int
#inDegree : int

+State(size : int)
+isTransientSet() : boolean
+State(size : int, value : long)
+size() : int
-State(original : State)
+copyFrom(original : State) : void
+set(bitIndex : int) : void
+set(bitIndex : int, value : boolean) : void
+setBit(bitIndex : int) : boolean
+setBit(bitIndex : int, value : boolean) : boolean
+clone() : Object
+makeClone() : State
+setLong(value : long) : void
+isGardenOfEden() : boolean
+increaseInDegree() : void
+resetInDegree() : void
#setFinalized() : void
#setId(id : long) : void
+nextState() : State
+makeID(id : long) : void
+getID() : long
+toLong() : long
+hasNext() : boolean
+hammingDistance(s : State) : int
+toString() : String
+equals(obj : Object) : boolean

```

```

Utils
+newUniqueID(db : JdbcDbManager) : long
+longToBitSet(value : long) : BitSet
+longToBitSet(value : long, bs : BitSet) : BitSet
+bitSetToLong(bs : BitSet) : long

```

```

Network_old
-nbNetworks : int = 0
-id : int
-name : String
-inDDtype : DistributionType
-outDDtype : DistributionType
-vertices : Vertex[]

+Network_old(source : LineNumberReader, name : String)
+Network_old(size : int, name : String)
+Network_old(size : int)
+Network_old(c : Chromosome)
+Network_old(size : int, outDD : DegreeDistribution, inDD : DegreeDistribution, name : String)
+getInDDtype() : DistributionType
+getOutDDtype() : DistributionType
+size() : int
+connectAtRandom(outDD : DegreeDistribution, inDD : DegreeDistribution, promoterPorportion : double) : void
+disconnectAll() : void
+getState() : State
+getState(state : State) : State
+setState(value : long) : State
+setState(value : long, state : State) : State
+setState(state : State) : void
+setRandomVerticesValues() : void
+setUpdateFunction(uf : UpdateFunction) : void
+toString() : String
+toString(separator : String) : String
+getDegreeDistribution(dir : DD_DIRECTION) : int []
+toDot() : String
+toDot(land : Landscape) : String
+toDotSubNet() : String
+averageDegree() : double
+main(args : String []) : void
+store(db : JdbcDbManager, rem : String) : long []
+alreadyStored(db : JdbcDbManager) : long
+getFromDB(db : JdbcDbManager, ids : long ...) : Network_old

```

```

<<Enum>>
DD_DIRECTION
<<Constant>> -IN
<<Constant>> -OUT

```

```

<<Interface>>
UpdateTiming
+SYNCHRONOUS : int = 0
+ACTIVATED_CASCADE : int = 1
+GLOBAL_ACTIVATED_CASCADE : int = 2

+update(net : Network) : void
+type() : int
+toString() : String

```

```

NetworkToDB
+PAJEK : int = 0
+BOOLEAN_FCT : int = 1
-FILES : String[] = {
    "/Volumes/Data/cdarabos/Desktop/aba_simple.txt",
    //"/Volumes/Data/cdarabos/Desktop/LGL_simple.txt",
    "/Volumes/Data/cdarabos/Desktop/immune.txt"
}
-REMARK : String[] = {
    "aba_simplified",
    //"/LGL_simplified",
    "immune"
}
-db_host : String = "tiburtina.unil.ch"
-db_name : String = "earbn"
-db_user : String = "cdarabos"
-db_pwd : String = "grid-db"
-db_test : String = "earbn_test"
+main(args : String []) : void

```


List of Publications

International Journal Publications

*Additive functions in Boolean networks: where synchronicity meets topology-driven update
A case study on three real-life regulatory networks*

Christian Darabos, Marco Tomassini, Ferdinando Di Cunto, Paolo Provero, and Mario Giacobini
BMC Systems Biology, under review, submitted January 29, 2010

Dynamics of Unperturbed and Noisy Generalized Boolean Networks

Christian Darabos, Mario Giacobini and Marco Tomassini
Journal of Theoretical Biology, 2009, Volume 260, pp. 531-544

Performance and Robustness of Cellular Automata Computation on Irregular Networks

Marco Tomassini, Mario Giacobini and Christian Darabos
Advances in Complex Systems, 2007, Volume 10, Number 1, pp. 85-110

Robustness of Evolved Small-World Automata Networks

Marco Tomassini, Mario Giacobini and Christian Darabos
Intelligenza Artificiale, 2006, Year 3, Number 3

Evolution and Dynamics of Small-World Cellular Automata

Marco Tomassini, Mario Giacobini and Christian Darabos
Complex Systems, 2005, Volume 15, Number 4, pp. 261-284

Peer Reviewed Book Chapters

Generalized Boolean Networks: How Spatial and Temporal Choices Influence Their Dynamics

Ch. Darabos, M. Giacobini and M. Tomassini

in **Computational Methodologies in Gene Regulatory Networks**, A book edited by Sanjoy Das, Doina Caragea, W. H. Hsu, Stephen M. Welch
Kansas State University, USA. To appear in 2010

Peer Reviewed Conferences & Workshops Publications

Toward Robust Network Based Complex Systems: From Evolutionary Cellular Automata to Biological Models Networks

Christian Darabos

Voted best paper in the "PhD Thesis summary" category

EvoPhD2010 - 5th European Graduate Student Workshop on Evolutionary Computation part of *EvoStar2010* - European events on Evolutionary Computation, April 2010.

Editors: C. Di Chio, M. Giacobini, J. van Hemerts, ISSN: 1974-4145

Are cells really operating at the Edge of Chaos ? A case study of two real-life regulatory networks

Ch. Darabos, M. Giacobini, M. Tomassini, P. Provero, F. DiCunto

ECAL 2009 - 10th European Conference on Advances in Artificial Life, to appear in Springer Editions, 2009

Transient Perturbations on Scale-Free Boolean Networks with Topology Driven Dynamics

Ch. Darabos, M. Giacobini, M. Tomassini

ECAL 2009 - 10th European Conference on Advances in Artificial Life, to appear in Springer Editions, 2009

Dynamics of Interconnected Boolean Networks with Scale-Free Topology

C. Damiani, M. Villani, Ch. Darabos, M. Tomassini

Wivace2008 - Italian Workshop on Artificial Life and Evolutionary Computation, CD-ROM Proceedings, 2008, ISBN: 978-88-903581-0-4

A Study of NK Landscapes' Basins and Local Optima Networks

Sebastien Verel, Gabriela Ochoa, Marco Tomassini, and Christian Darabos

GECCO'08 - Genetic and Evolutionary Computation Conference, 2008, Proceedings GECCO '08, ACM Press, 2008, pp. 555-562.

Semi-Synchronous Activation in Scale-Free Boolean Networks

Christian Darabos, Mario Giacobini and Marco Tomassini

ECAL 2007 - 9th European Conference on Advances in Artificial Life, Almeida e Costa F. et al. Eds, Springer Verlag, LNAI4648, 2007, pp. 976-985

Scale-Free Automata Networks Are Not Robust in a Collective Computational Task

Christian Darabos, Mario Giacobini and Marco Tomassini

ACRI 2006 - 5th International Conference on Cellular Automata for Research and Industry, El Yacoubi S. et al. Eds, Springer Verlag, LNCS4173, 2006, pp. 512-521

A New Fault-Tolerance Measure for Evolved Small-World Automata Networks

Christian Darabos, Mario Giacobini and Marco Tomassini

GSICE 06 - Second AI*IA Workshop on Evolutionary Computation, Siena, 2006

Evolution of Small-World Networks as a support for Cellular Automata Computation

Christian Darabos

EvoPhD2006 Workshop at EuroGP2006 & EvoCOP2006, incorporating EvoWorkshops2006, EvoPhD Workshop proceedings, 2006, pp. 15-30

Robustness of Evolved Small-World Automata Networks

Christian Darabos, Mario Giacobini and Marco Tomassini

GSICE 05 - First AI*IA Workshop on Evolutionary Computation, ISBN 88-900910-0-2, Milano (Italy), 2005

Evolution of Small-World Networks of Automata for Computation

Voted second best article of the conference.

Marco Tomassini, Mario Giacobini and Christian Darabos

Parallel Problem Solving from Nature, PPSN VIII, Eighth International Conference, Birmingham, United Kingdom, September 2004, Lecture Notes in Computer Science, LNCS 3242, Xin Yao et al. Eds., Springer Verlag, 672-681.

Peer Reviewed Abstracts Presented at International Conferences

Generalized Boolean Networks with Topology Driven Dynamics

Given as an oral presentation

Ch. Darabos, F. DiCunto, M. Tomassini, P. Provero, M. Giacobini

NetSci2009 - International Workshop on Network Science, Venice, Italy, 2009

How Spatial and Temporal Choices Influence the Dynamics of Generalized Boolean Networks

Ch. Darabos, M. Giacobini and M. Tomassini

ICSB 2008 - 9th International Conference on Systems Biology, August 2008, Gothenburg, Sweden

The Influence of Spatial and Temporal Choices on the Dynamics of Generalized Boolean Networks

Ch. Darabos, M. Giacobini and M. Tomassini

ISMB 2008 - 16th Annual International Conference Intelligent Systems for Molecular Biology, July 2008, Toronto, Canada

Bibliography

- [AB02] R. Albert and A.-L. Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, 2002.
- [ABCA⁺08] E. R. Álvarez-Buylla, Á. Chaos, M. Aldana, M. Benítez, Y. Cortes-Poza, C. Espinosa-Soto, D. A. Hartasánchez, R. B. Lotto, D. Malkin, G. J. Escalera-Santos, and P. Padilla-Longoria. Floral morphogenesis: Stochastic explorations of a gene network epigenetic landscape. *PLoS ONE*, 3(11):e3626, 11 2008.
- [ABKR07] M. Aldana, E. Balleza, S. A. Kauffman, and O. Resendiz. Robustness and evolvability in genetic regulatory networks. *Journal of Theoretical Biology*, 245:433–448, 2007.
- [AC03a] M. Aldana and P. Cluzel. A natural class of robust networks. *Proc. Nat. Acad. Sci. USA*, 100(15):8710–8714, 2003.
- [AC03b] M. Aldana and P. Cluzel. A natural class of robust networks. *Proc. Natl. Acad. Sci. USA*, 100(15):8710–8714, 2003.
- [ACK03] M. Aldana, S. Coppersmith, and L. P. Kadanoff. Boolean dynamics with random couplings. In E Kaplan, J. E. Marsden, and K. R. Sreenivasan, editors, *Perspectives and Problems in Nonlinear Science*, Springer Applied Mathematical Sciences Series, pages 23–89. Springer, Berlin, 2003.
- [ADGM⁺04] L. A. N. Amaral, A. Díaz-Guilera, A. Moreira, A. L. Goldberger, and L. A. Lipsitz. Emergence of complex dynamics in a simple model of signaling networks. *Proc. Nat. Acad. Sci. USA*, 101(44):15551–15555, 2004.
- [AJB00] R. Albert, H. Jeong, and L. Barabási. Error and attack tolerance of complex networks. *Nature*, 406:378–382, 2000.
- [Alb04] R. Albert. Boolean modeling of genetic regulatory networks. In E. Ben-Naim, H. Frauenfelder, and Z. Toroczkai, editors, *Complex Networks*, volume 650 of *Lecture Notes in Physics*, pages 459–479. Springer, Berlin, 2004.
- [Alb05] R. Albert. Scale-free networks in cell biology. *J. of Cell Science*, 118:4947–4957, 2005.
- [Ald03] M. Aldana. Boolean dynamics of networks with scale-free topology. *Physica D*, 185:45–66, 2003.
- [AO03] R. Albert and H. G. Othmer. The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster. *Journal of Theoretical Biology*, 223:1–18, 2003.
- [ARM98] A. Arkin, J. Ross, and H. H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage $\{\lambda\}$ -infected escherichia coli cells. *Genetics*, 149(4):1633–1648, 8 1998.
- [Art99] W. Brian Arthur. Complexity and the Economy. *Science*, 284(5411):107–109, 1999.

- [ASBS00] L. A. N. Amaral, A. Scala, M. Barthélemy, and H. E. Stanley. Classes of small-world networks. *Proc. Natl. Acad. Sci. USA*, 97(21):11149–11152, 2000.
- [AT02] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–462, 2002.
- [Axe97] R. Axelrod. *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*. Princeton Studies in Complexity. Princeton University Press, Princeton, NJ, 1997.
- [BABC⁺08] E. Balleza, E. R. Alvarez-Buylla, A. Chaos, S. A. Kauffman, I. Shmulevich, and M. Aldana. Critical dynamics in genetic regulatory networks: Examples from four kingdoms. *PLoS ONE*, 3(6):e2456, 06 2008.
- [Bak87] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 14–21, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc.
- [Bak96] P. Bak. *How Nature Works: The Science of Self-Organized Criticality*. Copernicus, New York, NY, USA, 1996.
- [BB01] J. M. Bower and H. Bolouri, editors. *Computational Modeling of Genetic and Biochemical Networks*. MIT Press, 2001.
- [BC78] E. A. Bender and E. R. Canfield. The asymptotic number of labelled graphs with given degree sequences. *J. of Combinatorial Theory A*, 24:296–307, 1978.
- [BKCC03] W. J. Blake, M. Kaern, C. R. Cantor, and J. J. Collins. Noise in eukaryotic gene expression. *Nature*, 422(6932):633–637, 04 2003.
- [BS01] S. Bilke and F. Sjunnesson. Stability of the kauffman model. *Phys. Rev. E*, 65(1):016129, Dec 2001.
- [BT95] T. Blickle and L. Thiele. A comparison of selection schemes used in genetic algorithms. Technical Report 11, Swiss Federal Institute of Technology, Gloriastrasse 35, 8092 Zurich, Switzerland, 1995.
- [BTW88] P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality. *Physical Review A*, 38(1):364–374, July 1988.
- [CGMA07] C. Christensen, A. Gupta, C. D. Maranas, and R. Albert. Inference and graph-theoretical analysis of *Bacillus Subtilis* gene regulatory networks. *Physica A*, 373:796–810, 2007.
- [Chu68] C. West Churchman. *The Systems Approach*. Dell Publishing, 1968.
- [Cil98] P. Cilliers. *Complexity and Postmodernism: Understanding Complex Systems*. Routledge, London, UK, 1998.
- [CMD03] J. P. Crutchfield, M. Mitchell, and R. Das. Evolutionary design of collective computation in cellular automata. In J. P. Crutchfield and P. Schuster, editors, *Evolutionary Dynamics: Exploring the Interplay of Selection, Accident, Neutrality, and Function*, pages 361–411. Oxford University Press, Oxford, 2003.
- [Coe99] C. A. Coello. An updated survey of evolutionary multiobjective optimization techniques: State of the art and future trends. In *Proceedings of the Congress on Evolutionary Computation*, pages 3–13. IEEE Press, 1999.
- [CST96] M. S. Capcarrère, M. Sipper, and M. Tomassini. Two-state, $r = 1$ cellular automaton that classifies density. *Physical Review Letters*, 77(24):4969–4971, 1996.

- [CXY⁺08] X. Chen, H. Xu, P. Yuan, F. Fang, M. Huss, V. B. Vega, E. Wong, Y. L. Orlov, W. Zhang, J. Jiang, Y.-H. Loh, H. C. Yeo, Z. X. Yeo, V. Narang, K. R. Govindarajan, B. Leong, A. Shahab, Y. Ruan, G. Bourque, W.-K. Sung, N. D. Clarke, C.-L. Wei, and H.-H. Ng. Integration of external signaling pathways with the core transcriptional network in embryonic stem cells. *Cell*, 133(6):1106–1117, 06 2008.
- [DBC05] H. J. Dupuy, D. Bertin, N. Cusick, and M. Vidal. Effect of sampling on topology predictions of protein-protein interaction networks. *Nature Biotech.*, 23:839–844, 2005.
- [DCMH95] R. Das, J. P. Crutchfield, M. Mitchell, and J. E. Hanson. Evolving globally synchronized cellular automata. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 336–343, San Francisco, CA, 1995. Morgan Kaufmann.
- [De02] E. H. Davidson and et al. A genomic regulatory network for development. *Science*, 295:1669–1678, March 1 2002.
- [DGT06] C. Darabos, M. Giacobini, and M. Tomassini. Scale-free automata networks are not robust in a collective computational task. In S. El Yacoubi et al., editor, *Cellular Automata, ACRI 2006*, volume 4173 of *Lecture Notes in Computer Science*, pages 512–521. Springer Verlag, Berlin, 2006.
- [DGT07] C. Darabos, M. Giacobini, and M. Tomassini. Semi-synchronous activation in scale-free boolean networks. In *Advances in Artificial Life*, volume 4648 of *Lecture Notes in Artificial Intelligence*, pages 976–985. Springer-Verlag, Heidelberg, 2007.
- [DGT⁺09] C. Darabos, M. Giacobini, M. Tomassini, P. Provero, and Ferdinando Di Cunto. Are cells really operating at the edge of chaos? a case study of two real-life regulatory networks. In G. Kampis et al., editor, *Advances in Artificial Life, ECAL2009*, Lecture Notes in Computer Science, pages 89 – 102, Berlin, 2009. Springer.
- [DJS92] K. A. De Jong and W. M. Spears. A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 5(1):1–26, 1992.
- [DM03] S. N. Dorogovtsev and J. F. F. Mendes. *Evolution of Networks: From Biological Nets to the Internet and WWW*. Oxford University Press, Oxford, UK, 2003.
- [DMC94] R. Das, M. Mitchell, and J. P. Crutchfield. A genetic algorithm discovers particle-based computation in cellular automata. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature- PPSN III*, volume 866 of *Lecture Notes in Computer Science*, pages 344–353, Heidelberg, 1994. Springer-Verlag.
- [DP86] B. Derrida and Y. Pomeau. Random networks of automata: a simple annealed approximation. *Europhysics Letters*, 1(2):45–49, 1986.
- [Dro08] B. Drossel. Random boolean networks. *Reviews of Nonlinear Dynamics and Complexity*, 1, 2008.
- [dSP65] D. J. de Solla Price. Networks of Scientific Papers. *Science*, 149(3683):510–515, 1965.
- [DTG09] C. Darabos, M. Tomassini, and M. Giacobini. Dynamics of unperturbed and noisy generalized boolean networks. *Journal of Theoretical Biology*, 260(4):531 – 544, 2009.
- [EG06] R. Edwards and L. Glass. A calculus for relating the dynamics and structure of complex biological networks. In R. S. Berry and J. Jortner, editors, *Advances in Chemical Physics*, volume 132 of *Advances in Chemical Physics*, pages 151–178. J. Wiley and Sons, New York, 2006.

- [EL00] M. B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, 01 2000.
- [ELSS02] M. B. Elowitz, A. J. Levine, E. D. Siggia, and P. S. Swain. Stochastic gene expression in a single cell. *Science*, 297(5584):1183–1186, 8 2002.
- [ER59a] P. Erdős and A. Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.
- [ER59b] P. Erdős and A. Rényi. On random graphs I. *Publicationes Mathematicae*, 5:290–297, 1959.
- [Fuk97] H. Fukš. Solution of the density classification problem with two cellular automata rules. *Physical Review E*, 55(3):2081–2084, 1997.
- [Gar70] M. Gardner. The fantastic combinations of John Conway’s new solitaire game “life”. *Scientific American*, 223(4):120–123, October 1970.
- [Gar71] M. Gardner. On cellular automata, self-reproduction, the Garden of Eden and the game “life”. *Scientific American*, 224(2):112–117, February 1971.
- [Gar95] M. Garzon. *Models of Massive Parallelism: Analysis of Cellular Automata and Neural Networks*. Springer-Verlag, Berlin, 1995.
- [GATT04] M. Giacobini, E. Alba, A. Tettamanzi, and M. Tomassini. Modeling selection intensity for toroidal cellular evolutionary algorithms. In *Proceedings of the genetic and evolutionary computation conference GECCO’04*, pages 1138–1149. Springer Verlag, Berlin, 2004.
- [GCC00] T. S. Gardner, C. R. Cantor, and J. J. Collins. Construction of a genetic toggle switch in *escherichia coli*. *Nature*, 403(6767):339–342, 01 2000.
- [GD91] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms, 1991.
- [Ger04] C. Gershenson. Updating schemes in random Boolean networks: Do they really matter? In J. Pollack, editor, *Artificial Life IX Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems*, pages 238–243. MIT Press, 2004.
- [GK99] N. Goldenfeld and L. P. Kadanoff. Simple Lessons from Complexity. *Science*, 284(5411):87–89, 1999.
- [Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.
- [GS91] M. Gorges-Schleuter. Explicit parallelism of genetic algorithms through population structures. In *PPSN I: Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, pages 150–159, London, UK, 1991. Springer-Verlag.
- [GTRP06] M. Giacobini, M. Tomassini, P. De Los Rios, and E. Pestelacci. Dynamics of scale-free semi-synchronous boolean networks. In L. M. Rocha et al., editor, *Artificial Life X*, pages 1–7, Cambridge, Massachusetts, 2006. The MIT Press.
- [Has95] M. H. Hassoun. *Fundamentals of artificial neural networks*. MIT Press, Cambridge, MA, 1995.
- [HB97] I. Harvey and T. Bossomaier. Time out of joint: attractors in asynchronous random boolean networks. In P. Husbands and I. Harvey, editors, *Proceedings of the Fourth European Conference on Artificial Life*, pages 67–75, Cambridge, MA, 1997. The MIT Press.

- [HBNP07] F. Hormozdiari, P. Berenbrink, and S. C. Sahinalp N. Przuli. Not all scale-free networks are born equal: the role of the seed graph in PPI network evolution. *PLoS Computational Biology*, 3:1373–1384, 2007.
- [HMICnt] J. Hasty, D. McMillen, F. Isaacs, and J. J. Collins. Computational studies of gene regulatory networks: in numero molecular biology. *Nat Rev Genet*, 2(4):268–279, 04 2001/04//print.
- [Hol75] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, Michigan, 1975.
- [IKY07] K. Iguchi, S. Kinoshita, and H. S. Yamada. Boolean dynamics of Kauffman models with a scale-free network. *J. Theor. Biol.*, 247:138–151, 2007.
- [Kau69] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol.*, 22:437–467, 1969.
- [Kau93] S. A. Kauffman. *The Origins of Order*. Oxford University Press, New York, 1993.
- [Kau00] S. A. Kauffman. *Investigations*. Oxford University Press, New York, 2000.
- [Kau03] S. A. Kauffman. Understanding genetic regulatory networks. *International Journal of Astrobiology*, 2(02):131–139, 2003.
- [Kitnt] H. Kitano. Biological robustness. *Nat Rev Genet*, 5(11):826–837, 11 2004/11//print.
- [LAA06] S. Li, S. M Assmann, and R. Albert. Predicting essential components of signal transduction networks: A dynamic model of guard cell abscisic acid signaling. *PLoS Biol*, 4(10):e312, 09 2006.
- [Lan90] C. G. Langton. Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D*, 42:12–37, 1990.
- [LB95] M. Land and R. K. Belew. No perfect two-state cellular automata for density classification exists. *Physical Review Letters*, 74(25):5148–5150, 1995.
- [Lec08] R. D. Leclerc. Survival of the sparsest: robust gene networks are parsimonious. *Mol Syst Biol*, 4, 08 2008.
- [Lei92] F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann, San Mateo, CA, 1992.
- [LLL⁺04] F. Li, T. Long, Y. Lu, Q. Ouyang, and C. Tang. The yeast cell-cycle network is robustly designed. *Proceedings of the National Academy of Sciences of the United States of America*, 101(14):4781–4786, 2004.
- [LT03] M. Levine and R. Tjian. Transcription regulation and animal diversity. *Nature*, 424(6945):147–151, 07 2003.
- [MCH94] M. Mitchell, J. P. Crutchfield, and P. T. Hraber. Evolving cellular automata to perform computations: mechanisms and impediments. *Physica D*, 75:361–391, 1994.
- [MFH91] M. Mitchell, S. Forrest, and J. H. Holland. The royal road for genetic algorithms: Fitness landscapes and ga performance. In *Proceedings of the First European Conference on Artificial Life*, pages 245–254. MIT Press, 1991.
- [MHC93] M. Mitchell, P. T. Hraber, and J. P. Crutchfield. Revisiting the edge of chaos: evolving cellular automata to perform computations. *Complex Systems*, 7:89–130, 1993.
- [Mil67] S. Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.

- [MMT05] C. Marr and M.-T.Hütt. Topology regulates pattern formation capacity of binary cellular automata on graphs. *Physica A*, 354:641–662, 2005.
- [Mor53] J. L. Moreno. *Who Shall Survive*. Beacon, NY, 1953.
- [MOW84] O. Martin, A. M. Odlyzko, and S. Wolfram. Algebraic properties of cellular automata. *Communications in Mathematical Physics*, 93:219–258, 1984.
- [MSA03] S. Maslov, K. Sneppen, and U. Alon. Correlation profiles and motifs in complex networks. In S. Bornholt and H. G. Schuster, editors, *Handbook of Graphs and Networks*, pages 168–198, Weinheim, GE, 2003. Wiley-VCH.
- [MT03] B. Mesot and C. Teuscher. Critical values in asynchronous random boolean networks. In W. Banzhaf, editor, *Advances in Artificial Life, ECAL2003*, volume 2801 of *Lecture Notes in Artificial Intelligence*, pages 367–376, Berlin, 2003. Springer.
- [New03] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.
- [Now06] M. A. Nowak. *Evolutionary Dynamics: Exploring the Equations of Life*. Harvard edition World, 2006.
- [OD04] P. Olivieri and Eric H. Davidson. Genes regulatory network controlling embryonic specification in the sea urchin. *Current Opinion in Genetics and Development*, 14:351–360, 2004.
- [OS02] C. Oosawa and M. A. Savageau. Effects of alternative connectivity on behavior of randomly constructed Boolean networks. *Physica D*, 170:143–161, 2002.
- [Pac88] N. H. Packard. Adaptation toward the edge of chaos. In J. A. S. Kelso, A. J. Mandell, and M. F. Shlesinger, editors, *Dynamic Patterns in Complex Systems*, pages 293–301. World Scientific, Singapore, 1988.
- [Pea06] H. Pearson. Genetics: What is a gene? *Nature*, 441(7092):398–401, 05 2006.
- [Pen07] E. Pennisi. GENOMICS: DNA Study Forces Rethink of What It Means to Be a Gene. *Science*, 316(5831):1556–1557, 2007.
- [Pri76] D. De Solla Price. A general theory of bibliometric and other cumulative advantage processes. *Journal of the American Society for Information Science*, 27(5):292–306, 1976.
- [Pri97] I. Prigogine. *The end of certainty*. The Free Press, 1997.
- [RB03] E. Ravasz and A. L. Barabasi. Hierarchical organization in complex networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 67(2), 2003.
- [Rie92] B. Riemann. Ueber die anzahl der primzahlen unter einer gegebenen groesse. *Monatsberichte der Berliner Akademie*, 1892.
- [RK07] A. S. Ribeiro and S. A. Kauffman. Noisy attractors and ergodic sets in models of gene regulatory networks. *J. Theor. Biol.*, 247:743–755, 2007.
- [RO05] J. M. Raser and E. K. O’Shea. Noise in gene expression: Origins, consequences, and control. *Science*, 309(5743):2010–2013, 9 2005.
- [RZ06] M. R. Roussel and R. Zhu. Validation of an algorithm for delay stochastic simulation of transcription and translation in prokaryotic gene expression. *Physical Biology*, 3(4):274–284, 2006.

- [SBB00] P. Smolen, D. Baxter, and J. Byrne. Modeling transcriptional control in gene networks—methods, recent results, and future directions. *Bulletin of Mathematical Biology*, 62(2):247–292, March 2000.
- [SDMnt] J. P. Sethna, K. A. Dahmen, and C. R. Myers. Crackling noise. *Nature*, 410(6825):242–250, 03 2001/03/08/print.
- [Sip97] M. Sipper. *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*, volume 1194 of *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, New York, 1997.
- [Sor03] D. Sornette. *Critical Phenomena in Natural Sciences*. Springer, Berlin, 2003.
- [SR97] M. Sipper and E. Ruppin. Co-evolving architectures for cellular machines. *Physica D*, 99:428–441, 1997.
- [SRN07] G. Stoll, J. Rougemont, and F. Naef. Representing perturbed dynamics in biological network models. *Phys. Rev. Lett. E*, 76, 2007.
- [STB96] M. Sipper, M. Tomassini, and O. Beuret. Studying probabilistic faults in evolved non-uniform cellular automata. *International Journal of Modern Physics C*, 7(6):923–939, 1996.
- [Str01a] S. H. Strogatz. Exploring complex networks. *Nature*, 410(6825):268–276, 03 2001.
- [Str01b] S. H. Strogatz. *Nonlinear Dynamics And Chaos: With Applications To Physics, Biology, Chemistry, And Engineering (Studies in nonlinearity)*. Studies in nonlinearity. Westview Press, 1 edition, January 2001.
- [SVA04] R. Serra, M. Villani, and L. Agostini. On the dynamics of random boolean networks with scale-free outgoing connections. *Physica A: Statistical Mechanics and its Applications*, 339(3-4):665–673, 8 2004.
- [SVGK07] R. Serra, M. Villani, A. Graudenzi, and S.A. Kauffman. Why a simple model of genetic regulatory networks describes the distribution of avalanches in gene expression data. *Journal of Theoretical Biology*, 246:449–460, 2007.
- [SVS04] R. Serra, M. Villani, and A. Semeria. Genetic network models and statistical properties of gene expression data in knock-out experiments. *Journal of Theoretical Biology*, 227:149–157, 2004.
- [Sys89] G. Syswerda. Uniform crossover in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 2–9, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.
- [TGD04] M. Tomassini, M. Giacobini, and C. Darabos. Evolution of small-world networks of automata for computation. In X. Yao et al., editor, *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 672–681. Springer Verlag, Berlin, 2004.
- [TGD05] M. Tomassini, M. Giacobini, and C. Darabos. Evolution and dynamics of small-world cellular automata. *Complex Systems*, 15:261–284, 2005.
- [Tom05] M. Tomassini. *Spacially Structured Evolutionary Algorithms*. Springer, 2005.
- [TT01] A. Tettamanzi and M. Tomassini. *Soft Computing: Integrating Evolutionary, Neural and Fuzzy Systems*. Springer, Berlin, Heidelberg, New York, 2001.
- [TV01] M. Tomassini and M. Venzi. Evolving robust asynchronous CA for the density task. *Complex Systems*, 13(3):185–204, 2001.

- [TYD05] R. Tanaka, T. Yi, and J. Doyle. Some protein interaction data do not exhibit power-law statistics. *FEBS Letters*, 579:5140–5144, 2005.
- [Uni01] Defense Acquisition University. *Systems Engineering Fundamentals*. Defense Acquisition University Press, 2001.
- [VDS⁺04] A. Vázquez, R. Dobrin, D. Sergi, J.-P. Eckmann, Z. N. Oltvai, and A.-L. Barabási. The topological relationships between the large-scale attributes and local interactions patterns of complex networks. *Proc. Natl. Acad. Sci USA*, 101(52):17940–17945, 2004.
- [vH78] F. von Hayek. *The Results of Human Action but Not of Human Design*, pages 96–105. New Studies in Philosophy, Politics, Economics. University of Chicago Press, 1978.
- [vN66] J. von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Illinois, 1966. Edited and completed by A. W. Burks.
- [Voh01] J. Vohradsky. Neural Model of the Genetic Network. *Journal of Biological Chemistry*, 276(39):36168–36173, 2001.
- [VSKB92] H.M. Voigt, I. Santibáñez-Koref, and J. Born. Hierarchically structured distributed genetic algorithms. In Männer R. and Manderick B., editors, *Proceedings of the International Conference Parallel Problem Solving from Nature 2*, 1992.
- [Wat99] D. J. Watts. *Small Worlds: The Dynamics of Networks Between Order and Randomness*. Princeton University Press, Princeton, NJ, 1999.
- [WBI99] G. Weng, U. S. Bhalla, and R. Iyengar. Complexity in Biological Signaling Systems. *Science*, 284(5411):92–96, 1999.
- [Wea48] W. Weaver. Science and complexity. *American Scientist*, 36(536), 1948.
- [WI99] G. M. Whitesides and R. F. Ismagilov. Complexity in Chemistry. *Science*, 284(5411):89–92, 1999.
- [Wol94] S. Wolfram. *Cellular Automata and Complexity*. Addison-Wesley, Reading, MA, 1994.
- [WS98] D. J. Watts and S. H. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393:440–442, 1998.
- [Wue89] A. Wuensche. Genomic regulation modeled as a network with basins of attraction. In P. Altman et al., editor, *Pacific Symposium on Biocomputing*, pages 89 – 102. World Scientific, 1989.

