Artificial Evolution on Network Structures
How Time and Space Influence Dynamics

Thèse de doctorat
présentée à
Université de Lausanne
Faculté des Sciences
et
Università degli Studi di Milano
Facoltà di Scienze Matematiche, Fisiche e Naturali

par
Mario Giacobini
Laurea in Matematica
Università degli Studi di Torino
Torino, Italia
DEA en Mathématiques Discrètes et Fondements de l’Informatique
Université Aix-Marseille II
Marseille, France

Jury
Prof. Raymond Olivier, Président
Prof. Vincenzo Piuri, Rapporteur
Prof. Marco Tomassini, Codirecteur de thèse
Prof. Andrea Tettamanzi, Codirecteur de thèse
Prof. Marc Schoenauer, Expert
Prof. Moshe Sipper, Expert

LAUSANNE, 2005
Vu le rapport présenté par le jury d'examen, composé de

Président
Co-directeur de thèse
Co-directeur de thèse
Rapporteur
Experts

M. le Prof. Raymond Olivier
M. le Prof. Andrea G.B. Tettamanzi
M. le Prof. Marco Tomassini
M. le Prof. Vincenzo Piuri
M. le Prof. Marc Schoenauer
M. le Prof. Moshe Sipper

le Conseil de Faculté autorise l'impression de la thèse de

Monsieur Mario Giacobini
DEA en Mathématiques Université d'Aix – Marseille II

intitulée

Artificial Evolution on Network Structures: How Time and Space Influence Dynamics

Lausanne, le 17 novembre 2005

Pour le Doyen de la Faculté des Sciences

M. le Prof. Raymond Olivier
To Ebe and Dante,

who have been and will always be a part of me

and to Martina,

who is not yet but will soon be a part of me
Abstract

The main objective of this work is to show how the choice of the temporal dimension and of the spatial structure of the population influences an artificial evolutionary process. In the field of Artificial Evolution we can observe a common trend in synchronously evolving panmictic populations, i.e., populations in which any individual can be recombined with any other individual. Already in the '90s, the works of Spiessens and Manderick, Sarma and De Jong, and Gorges-Schleuter have pointed out that, if a population is structured according to a mono- or bi-dimensional regular lattice, the evolutionary process shows a different dynamic with respect to the panmictic case.

In particular, Sarma and De Jong have studied the selection pressure (i.e., the diffusion of a best individual when the only selection operator is active) induced by a regular bi-dimensional structure of the population, proposing a logistic modeling of the selection pressure curves. This model supposes that the diffusion of a best individual in a population follows an exponential law. We show that such a model is inadequate to describe the process, since the growth speed must be quadratic or sub-quadratic in the case of a bi-dimensional regular lattice. New linear and sub-quadratic models are proposed for modeling the selection pressure curves in, respectively, mono- and bi-dimensional regular structures. These models are extended to describe the process when asynchronous evolutions are employed. Different dynamics of the populations imply different search strategies of the resulting algorithm, when the evolutionary process is used to solve optimisation problems. A benchmark of both discrete and continuous test problems is used to
study the search characteristics of the different topologies and updates of the populations.

In the last decade, the pioneering studies of Watts and Strogatz have shown that most real networks, both in the biological and sociological worlds as well as in man-made structures, have mathematical properties that set them apart from regular and random structures. In particular, they introduced the concepts of small-world graphs, and they showed that this new family of structures has interesting computing capabilities. Populations structured according to these new topologies are proposed, and their evolutionary dynamics are studied and modeled. We also propose asynchronous evolutions for these structures, and the resulting evolutionary behaviors are investigated.

Many man-made networks have grown, and are still growing incrementally, and explanations have been proposed for their actual shape, such as Albert and Barabási’s preferential attachment growth rule. However, many actual networks seem to have undergone some kind of Darwinian variation and selection. Thus, how these networks might have come to be selected is an interesting yet unanswered question. In the last part of this work, we show how a simple evolutionary algorithm can enable the emergence of these kinds of structures for two prototypical problems of the automata networks world, the majority classification and the synchronisation problems.
Nous devons connaître, nous connaîtrons.

– Epitaphe sur le tombeau
de David Hilbert

L’objectif principal de ce travail est de montrer l’influence du choix de la dimension temporelle et de la structure spatiale d’une population sur un processus évolutionnaire artificiel. Dans le domaine de l’Evolution Artificielle on peut observer une tendance à évoluer d’une façon synchrone des populations panmictiques, où chaque individu peut être récombiné avec tout autre individu dans la population. Déjà dans les année ’90, Spiessens et Manderick, Sarma et De Jong, et Gorges-Schleuter ont observé que, si une population possède une structure régulière mono- ou bi-dimensionnelle, le processus évolutionnaire montre une dynamique différente de celle d’une population panmictique.

En particulier, Sarma et De Jong ont étudié la pression de sélection (c-à-d la diffusion d’un individu optimal quand seul l’opérateur de sélection est actif) induite par une structure régulière bi-dimensionnelle de la population, proposant une modélisation logistique des courbes de pression de sélection. Ce modèle suppose que la diffusion d’un individu optimal suit une loi exponentielle. On montre que ce modèle est inadéquat pour décrire ce phénomène, étant donné que la vitesse de croissance doit obéir à une loi quadratique ou sous-quadratique dans le cas d’une structure régulière bi-dimensionnelle. De nouveaux modèles linéaires et sous-quadratique sont proposés pour des structures mono- et bi-dimensionnelles. Ces modèles sont étendus pour décrire des processus évolutionnaires asynchrones. Différentes dynamiques de la population impliquent stratégies différentes de recherche de l’algorithme résultant lorsque le processus évolutionnaire est utilisé pour résoudre des problèmes d’optimisation. Un ensemble de problèmes discrets et continus est
utilisé pour étudier les caractéristiques de recherche des différentes topologies et mises à jour des populations.

Ces dernières années, les études de Watts et Strogatz ont montré que beaucoup de réseaux, aussi bien dans les mondes biologique et sociologique que dans les structures produites par l’homme, ont des propriétés mathématiques qui les séparent à la fois des structures régulières et des structures aléatoires. En particulier, ils ont introduit la notion de graphe *small-world* et ont montré que cette nouvelle famille de structures possède des intéressantes propriétés dynamiques. Des populations ayant ces nouvelles topologies sont proposées, et leurs dynamiques évolutionnaires sont étudiées et modélisées. Pour des populations ayant ces structures, des méthodes d’évolution asynchrone sont proposées, et la dynamique résultante est étudiée.

Beaucoup de réseaux produits par l’homme se sont formés d’une façon incrémentale, et des explications pour leur forme actuelle ont été proposées, comme le *preferential attachment* de Albert et Barabási. Toutefois, beaucoup de réseaux existants doivent être le produit d’un processus de variation et sélection darwinienne. Ainsi, la façon dont ces structures ont pu être sélectionnées est une question intéressante restée sans réponse. Dans la dernière partie de ce travail, on montre comment un simple processus évolutif artificiel permet à ce type de topologies d’émerger dans le cas de deux problèmes prototypiques des réseaux d’automates, les tâches de densité et de synchronisation.
Acknowledgements

We cannot even imagine a world where no one ever thanks anybody! Except for the doors! [...] A world similar to ours, it’s not even conceivable!

– Daniel Pennac

Acknowledging all those that have played a role in the five-years journey to this PhD dissertation is a hard task. Although it is easy to remember all of them, finding the good words and the right order is a tough job. To solve this latter problem, I’ve decided to follow the path traced by one of my favorite writers, Daniel Pennac, in his pamphlet *Merci*.

Like all literary genres, acknowledgement obeys its laws. It’s a centrifugal genre, in the undulatory sense of the term. Like a stone thrown in a pond, acknowledgement produces circles ... centrifugal, larger and larger ... farther and farther from the center.

Let me now throw a stone of my own in the pond and observe the different circles it will produce. I will explain my reasons for gratitude to each face that appears in every circle.

The rewarded first acknowledges the first circle: the notables, the important people, the jury, without whom the reward would have not been awarded;

Had a jury not been disposed to read, correct and judge it, this document would not be a PhD dissertation . All my thanks to Prof. Raymond Olivier of the University of Lausanne, who has dedicated part of his time to preside the works of the jury, and to Prof. Vincenzo Piuri, who has represented the University of Milano.

Special thanks to Prof. Marc Schoenauer of Paris–Sud University and to Prof. Moshe Sipper of Ben–Gurion University, the two experts of the jury, whose comments on the work presented in this manuscript have helped me improve its quality, and whose questions have
opened future directions for my research.

Prof. Marco Tomassini and Prof. Andrea Tettamanzi have also been members of the jury, nevertheless acknowledging this role only is not nearly sufficient. Along these five years they have been there at all times to guide my research, to answer all my questions, and to encourage me. It has surely not been easy to face all my doubts and questions, but they have always been there for me, building the master-disciple relationship that I’ve always dreamt of. Working with them has been, and still is, a pleasure, and the friendship that has grown during these years will surely last for a long time.

Even if he has not been part of the jury, let me thank Prof. Lemont B. Kier of the Virginia Commonwealth University for his comments and corrections on this manuscript, and for encouraging me in following my research.

then the second circle: the audience, you in this circumstance, who are here to rejoice for me, this evening, and it’s very kind of you, really, I thank you for that ...

A thesis would not exist without someone who reads it: I want to thank you, my reader, for dedicating your time to this dissertation. I hope that this work will interest and stimulate you.

I also want to acknowledge all those who have attended my presentations at various conferences. Their comments and questions have been useful to me in developing my research.

then the third circle: the team, without whom his work would not be as it is.

Had I been alone in doing my research, it would have certainly been much more difficult to write this dissertation. In particular, the work presented in the third part of this manuscript is the result of the work done with Christian Darabos: working with him is not only productive, but also a great pleasure. I hope that our collaboration and friendship will last for many further years.

When I arrived in Lausanne in 2001, I had the chance to meet Leonardo Vanneschi, who helped me in discovering this new reality. Moreover, we have quickly discovered a fruitful feeling in working together, and for these reasons I owe him my gratitude.
In the last two years my research interests have crossed those of Leslie Luthi and Enea Pestelacci: even if the work done with them is not part of this dissertation, surely our discussions have contributed to it.

A special acknowledgement goes to Prof. Enrique Alba: the main research line of this thesis started thanks to his collaboration with Prof. Marco Tomassini. His contribution to this dissertation is evident when looking at the related publications.

After the usual salutations [...] he dissertates on what he owes to the ones and the others, this soil so rich and so varied where his roots plunge, he gives thanks to what he has seen, read, heard, since his childhood ...

I owe a lot of gratitude to those who have guided me through my first research steps: Prof. Umberto Cerruti at the University of Torino, and Prof. Pierre Liardet at the University of Marseille, who helped me discover the field of Artificial Evolution.

The passion for this research area has grown also thanks to Tom Lenaerts and Jano van Hemert, colleagues and friends. Our discussions during the past seven years have been very fruitful and stimulating.

If it is true that I discovered this field only in the last years, my passion for research plunges its roots in my childhood. Even if they were working in a different field, my parents have showed me with their example how exciting such passion can be.

Then who else? ... His friends? The elective family ... those that we have chosen by ourselves, along the years, one by one, in full consciousness, the best bottles ...

A good environment is more than useful to succeed in one’s activity, and I owe all my gratitude to Prof. François Grize and Elisabeth Fournier who are the main persons to thank for the special atmosphere of the Information Systems Institute of the University of Lausanne where I have had the chance to work during the last five years.

Important ingredients of this unique environment are surely those who have worked in it during these years: I want to thank all my colleagues, and in particular Marc Mercier, Mathieu Capcarrère, Denis Rochat and Javier Iglesias, who have patiently stand my good and bad moods.

Last but surely not least, a really special thank you goes to Silvia, my wife, who has played a fundamental role in supporting and encouraging me during these years. Without
her it would have been much harder and less enjoyable to bring this dissertation to its end.

Oh yes, gratitude is a long history, like charity: you must not forget anybody. But we always forget someone. There is always somebody that passes through.

For those I might have forgotten to thank here, all my apologizes and acknowledgments for their contribution to this dissertation.

Mario Giacobini
Lausanne, November 2005
Overview of the Thesis

The White Rabbit put on his spectacles. “Where shall I begin, please your Majesty?” he asked.

“Begin at the beginning,” the King said, very gravely, “and go on till you come to the end: then stop.”

– Charles Lutwidge Dodgson
    (Lewis Carroll)

This work is divided into four parts: we start with the motivation of this thesis, and an introduction to Artificial Evolution and to those notions of space and time that will be used all through this dissertation. Then, we will pass to the main part, in which we will discuss the modeling of the dynamics of structured populations evolving using synchronous and asynchronous generational steps. In the third part, we will investigate how an artificial evolutionary process can enable networks with interesting computational capabilities to emerge. Finally, the last part will be devoted to drawing the final considerations and discussing the future research directions that this work opens.

Part I: Background and Motivation

Chapter 1. An introduction to Artificial Evolution with special attention to the historical origins of Evolutionary Computation is given in this chapter. The basic concepts of an Evolutionary Algorithm will be presented with emphasis on the selection mechanism, the key operator for this work.

Chapter 2. This chapter will focus on the spatial and time dimensions of an artificial evolution. The basic notions of graph theory, together with the different asynchronous generational updates, will be presented in order to introduce the reader to the main mechanisms that will be investigated in the dissertation.
Part II: Artificial Evolution on Structures

Chapter 3. Selection pressure is the key concept of this second part. It will be introduced in this chapter, together with its relation with the takeover time. We will define the mathematical notations and the general models used in this part. Finally, the logistic modeling of selection pressure for two-dimensional regular structures proposed by Sarma and De Jong will be discussed.

Chapter 4. The first topology to be studied is the one-dimensional regular lattice. We will present the different synchronous and asynchronous models for selection pressure curves concerning this topology. The models will be experimentally validated.

Chapter 5. We will extend the models of the previous chapter to two-dimensional regular lattices. Even though we will concentrate more on square lattices, rectangular lattices won’t be left aside, and general models for topologies with different radii neighborhoods will be presented. All models will be experimentally validated.

Chapter 6. The search capabilities of an evolutionary algorithm are directly linked to their dynamics. In this chapter we will present a well chosen set of discrete and continuous problems to study the different optimisation behaviors shown by a cEA with a two-dimensional structure when the grid shape and the time dimension are varied.

Chapter 7. This chapter is dedicated to the study of populations with irregular topologies. In a first part, random-graph structures are analysed and modeled, while in the second part, the attention will be focused on the evolution of population with small-world and scale-free topologies.

Part III: Artificial Evolution of Networks

Chapter 8. Cellular automata will be introduced here, focusing on their utilisation as computational systems. In particular, two prototypical problems will be described, the density classification and the synchronisation tasks. These problems will be the evolutionary environment of the next chapter.
Chapter 9. We will show how automata network structures with interesting computing capabilities using a simple evolutionary algorithm. We will also see how artificial evolution can be used to produce topologies in the small-world class as those Watts constructed by hand.

Part IV: Final Considerations

Chapter 10. In this last chapter we will draw the conclusions for the two main parts of this dissertation, the modeling of the selection pressure in evolving structured populations, and the emergence of structures in automata networks for computation. The main prospects together with hints on future directions of research opened by this work will be outlined.

Overview of the Related Publications

In this section we give an overview by chapter of the proceedings and journals in which the results presented in this dissertation have been published.

Chapter 3. Our study on selection pressure for structured population started with a publication at the Genetic and Evolutionary Computation Conference 2003 (GECCO 2003) (Giacobini et al., 2003): in this paper the criticism to the logistic modeling can be found. An introduction to the mathematical models can be found in IEEE Transaction on Evolutionary Computation (Giacobini et al., 2005b).

Chapter 4. The mathematical models for one-dimensional regular structures are published in the Proceedings of the 6th International Conference on Artificial Evolution (EA 2003) (Giacobini et al., 2004b), and in IEEE Transaction on Evolutionary Computation (Giacobini et al., 2005b).

Chapter 5. Selection pressure on two-dimensional regular structures are analysed and modeled in the Proceedings of the Genetic and Evolutionary Computation Conference 2003 (GECCO 2003) (Giacobini et al., 2003), the Proceedings of the Genetic
and Evolutionary Computation Conference 2004 (GECCO 2004) (Giacobini et al., 2004a). All the models, together with the extended models for rectangular lattices and different neighborhood structures, can be found in IEEE Transaction on Evolutionary Computation (Giacobini et al., 2005b).

**Chapter 6.** The analysis of the behavior of structured populations in optimisation problems can be found in the Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN VII) (Alba et al., 2002) and in the Proceedings of the 2004 Congress on Evolutionary Computation (CEC 2004) (Dorronsoro et al., 2004). This study is completed and extended to different lattice shapes in chapter 7 of the Handbook of Bioinspired Algorithms and Applications (Alba et al., 2005).

**Chapter 7.** Irregular structures are introduced and analysed in the Proceedings of the Genetic and Evolutionary Computation Conference 2005 (GECCO 2005) (Giacobini et al., 2005a).

**Chapter 9.** A preliminary study on the evolution of automata networks has been published in the Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII) (Tomassini et al., 2004). A complete survey of the results presented in this chapter can be found in Complex Systems (Tomassini et al., 2005).
# Table of Contents

Abstract iv
Synopsis vi
Acknowledgements viii
Overview of the Thesis xiii

**Part I  Background and Motivation** 1

Chapter 1 Artificial Evolution 3
  1.1 The Biological Metaphor .......................... 4
  1.2 Prehistory and Origins of Evolutionary Computation ........................ 7
  1.3 Basic Principles of an Evolutionary Algorithm .......................... 11
  1.4 Artificial Selection .................................. 13
    1.4.1 Fitness Proportionate Selection ......................... 14
    1.4.2 Linear Ranking Selection .............................. 15
    1.4.3 Local k-Tournament Selection ......................... 16
    1.4.4 Truncation Selection ................................. 16
    1.4.5 Classification of Selection Schemes ................... 17

Chapter 2 Space and Time 19
  2.1 Cellular Evolutionary Algorithms .......................... 21
    2.1.1 Useful Definitions for Graphs ........................ 24
  2.2 The Time Dimension .................................. 26

**Part II  Artificial Evolution on Structures** 29

Chapter 3 Selection Pressure 31
  3.1 Takeover Time ........................................ 32
  3.2 Mathematical Models ................................. 33
  3.3 Limitations of the Logistic Model ........................ 34

Chapter 4 One-Dimensional Structures 37
  4.1 Synchronous Takeover Time .............................. 38
  4.2 Asynchronous Fixed Line Sweep Takeover Time ............... 39
  4.3 Asynchronous Fixed and New Random Sweep Takeover Times ........... 40
# TABLE OF CONTENTS

4.4 Asynchronous Uniform Choice Takeover Time .................................. 41
4.5 Experimental Validation ............................................................. 42
4.5.1 Binary Tournament Selection .................................................. 43
4.5.2 Linear Ranking Selection ......................................................... 44
4.6 Summary ....................................................................................... 46

Chapter 5 Two-Dimensional Structures ................................................. 49
5.1 The Geometrical Approximation ...................................................... 50
5.2 Synchronous Takeover Time ............................................................ 51
5.3 Asynchronous Fixed Line Sweep Takeover Time ................................. 51
5.4 Asynchronous Fixed and New Random Sweep Takeover Time ............... 52
5.5 Asynchronous Uniform Choice Takeover Time ..................................... 53
5.6 Experimental Validation .................................................................. 54
5.6.1 Binary Tournament Selection .................................................... 55
5.6.2 Linear Ranking Selection ........................................................... 56
5.7 Rectangular Toroidal Structures ....................................................... 58
5.7.1 Rectangular Toroidal Models Validation ...................................... 61
5.8 Varying the Radius .................................................................... 62
5.9 Summary ....................................................................................... 64

Chapter 6 Optimisation by Evolution on Regular Structures .................. 71
6.1 Experiments in Discrete Optimization .............................................. 72
6.1.1 Massively Multimodal Deceptive Problem .................................... 72
6.1.2 Frequency Modulation Sounds ................................................... 73
6.1.3 Multimodal Problem Generator P-PEAKS .................................... 74
6.1.4 Error Correcting Code Design Problem ....................................... 74
6.1.5 Maximum Cut of a Graph ........................................................ 75
6.1.6 Minimum Tardy Task Problem .................................................. 76
6.1.7 Satisfiability Problem ............................................................... 77
6.1.8 Experimental Analysis in Discrete Optimization ............................. 77
6.2 Experiments in Continuous Optimization ......................................... 81
6.2.1 Rastrigin’s Function .................................................................. 82
6.2.2 Ackley’s Function .................................................................... 83
6.2.3 Fractal Function (FRAC) .......................................................... 83
6.2.4 Experimental Analysis in Continuous Optimization ....................... 84
6.3 Summary ....................................................................................... 86

Chapter 7 Evolution on Irregular Structures ......................................... 89
7.1 The General Model .................................................................... 90
7.2 The Random Graph Structure ....................................................... 91
7.3 Experimental Results on Random Graph Structures ......................... 94
7.4 Small-World Graph Topologies ...................................................... 97
7.4.1 The Watts-Strogatz Model ......................................................... 97
7.4.2 The Barabási-Albert Model ....................................................... 99
7.5 Experimental Results on Small Worlds ........................................... 100
TABLE OF CONTENTS

7.5.1 Watts-Strogatz Model ........................................ 100
7.5.2 Barabási-Albert Model ...................................... 102
7.6 Summary .......................................................... 104

Part III  Artificial Evolution of Networks  107

Chapter 8  Cellular Automata  109
  8.1 Introduction .................................................... 110
  8.2 Formal Definitions ............................................ 111
  8.3 Variations on the Original Model ............................ 114
    8.3.1 Non-Uniform Cellular Automata .......................... 114
    8.3.2 Non-Standard Architectures .............................. 114
    8.3.3 Asynchronous Cellular Automata .......................... 115
    8.3.4 Probabilistic Cellular Automata .......................... 115
  8.4 Cellular automata as computational systems .................. 116
    8.4.1 The majority task ......................................... 118
    8.4.2 The synchronization task .................................. 120

Chapter 9  Automata Network Evolution  121
  9.1 The Evolutionary Machinery ................................... 122
  9.2 Evolution for the density task from regular lattices ......... 125
  9.3 Evolution for the density task from random graphs ............ 127
  9.4 Evolution of automata networks for the synchronization task .. 132
  9.5 Limiting the number of shortcuts ............................ 134
  9.6 Task flexibility of the evolved networks ..................... 138
  9.7 Robustness in the presence of random faults .................. 139
  9.8 Summary ...................................................... 142

Part IV  Final Considerations  151

Chapter 10  Discussion of the Results  153
  10.1 Part II: Artificial Evolution on Structures .................. 153
  10.2 Part III: Artificial Evolution of Networks ................... 154

Part V  Annexes  157

Statistical Comparison of the Studied Algorithms  159
Adresses of the Jury Members  163
List of Publications  165
List of Attended Conferences  169
References  171
Part I

Background and Motivation
Chapter 1

Artificial Evolution

The relation of the scientist with the reality that surrounds him is, as for the artist, a continuous dialogue of (sometimes correct) anticipations, (frequent) errors and (necessary) connections, a dialogue that has many characteristics of an evolutionary process.

– Jean-Pierre Changeux

In Artificial Intelligence (AI), and in particular for those approaches called *bottom-up*, researchers have often looked at biology as a metaphor for designing intelligent systems. The biological world is in fact full of examples of emergent intelligent behaviors, in which the interactions of simple micro-components enable complex behaviors to emerge. These behaviors can be considered intelligent at the macro-level of the ensemble of the micro-components.

Since the beginning of AI researches in the 40s, one of the first metaphors that attracted AI researchers has been the neural intelligence, in which the interaction by chemical signal transmission of simple neuronal cells perform complicated tasks as image processing and recognition, language understanding, and many stimulus-reaction behaviors. The artificial simplified reproduction of this paradigm, the Artificial Neural Networks (ANNs), is still nowadays a very prolific research area. Other more recent examples of mimicking biological emergent intelligent systems are the Artificial Ant Colonies (ACCs), in which artificial ants are used to find optimal paths on graphs, and Particle Swarm Optimisation (PSO), where an ensemble of particles traverse a search space in order to find solutions to a given problem.
It’s not our aim here to describe these systems in detail: a good perspective on the role of ANNs in AI can be found in Russell and Norvig (2003), and an introduction to ANNs can be found in Tettamanzi and Tomassini (2001). Concerning the AACs, their origins can be traced back to the article by Deneubourg et al. (1983) who described the algorithmic behavior of natural ants, and to the paper by Manderick and Moyson (1988), where the first artificial ant is described. The first optimisation system based on AACs was proposed by Colorni et al. (1992), and a good introduction to these algorithms can be found in Dorigo and Stutzle (2004). A good reference on PSO is Bonabeau et al. (1999).

In this chapter we will introduce and discuss another class of biologically inspired systems, the Artificial Evolutionary Systems, the research environment in which the results presented in this dissertation are developed. Evolutionary Algorithms (EAs) are a broad class of stochastic optimisation algorithms, inspired by those biological processes that allow populations of organisms to adapt to their surrounding environment: genetic inheritance and survival of the fittest. An interesting introduction to EAs as a bottom-up approach in AI is David Fogel’s *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence* (Fogel, 2000).

The chapter is organized as follows: in the next section we will outline some principles of the biological evolution by natural selection that have been of inspiration to EAs. Then, in Section 1.2, the history of EAs will be briefly sketched (a complete survey of the history of Evolutionary Computation can be found in Fogel (1998)). The basic ingredients of an EA will be presented in Section 1.3, and in the next section the operator that plays a central role in this dissertation, the selection operator, will be discussed. A particular attention will be devoted to the four more commonly used selection operators: fitness proportionate (Section 1.4.1), local k-tournament (Section 1.4.3), linear ranking (Section 1.4.2), and truncation (Section 1.4.4).

### 1.1 The Biological Metaphor

Talking about biological evolution during these years of hard debates is not an easy task. Our intent here is not to give a complete overview of the different theories on evolution that
have been recently proposed and opposed to one another. Entering in such a debate would be presumptuous and we couldn’t add any significant points to it. In this dissertation, what really interests us are those concepts that are nowadays accepted as a common factor of every evolutionary theory, and that have been the inspiration points of the early AI researchers on Artificial Evolution when designing the first EAs. In the following sections we will see how these biological mechanisms have been translated into computational operators in order to implement Artificial Evolutionary Systems.

The publication in 1859 of Charles Darwin’s *On the Origin of Species by Means of Natural Selection* (Darwin, 1859) has marked a revolution in biological thought, and of course in human philosophy. The importance of the role this theory plays cannot be denied: the most prominent researchers in the field have clearly recognized it, from Dobzhansky and colleagues, who argued that “Nothing in biology makes sense except in the light of evolution” (Dobzhansky et al., 1977), to Mayr, who claimed that “There is no area in biology which [evolution] has not served as an ordering principle” (Mayr, 1963).

This theory, combined with the selectionism of Weismann and the genetics of Mendel, is now known as Neo-Darwinism, and “asserts that the history and the vast majority of life is fully accounted for by only a very few statistical processes acting on and within population of species” (Fogel, 2000). These basic processes are reproduction, mutation, competition and selection, and act on populations of individuals, each one being the product of the interaction of a genetic coding with the environment.

To illustrate the distinction between an individual (phenotype), its genetic coding (genotype) and the action of the different evolutionary processes on them, we have depicted in figure 1.1 the model proposed by Lewontin (1974) and modified by Atmar (1992). The evolutionary process is modeled as a map between the spaces \( G \) of the population genotypes, \( P \) of the population phenotypes, and \( I \) of the environmental interaction with the genotypes’ space for the development of the individuals’ phenotypes. A generational step is then seen as the composition of the four following maps:

- \( f_1 : G \times I \to P \), called *epigenesis*, that maps an ensemble of genotypes \( g_1 \in G \) to an ensemble of phenotypes \( p_1 \in P \) whose development is modified by the environment via the interaction between \( g_1 \) and the space \( I \);
• $f_2 : \mathbf{P} \to \mathbf{P}$, called selection, that maps a population of phenotypes $p_1 \in \mathbf{P}$ into another population of phenotypes $p_2 \in \mathbf{P}$; note that since natural selection operates only on phenotypes, the genotypical expressions of the individuals belonging to the populations is not involved by the action of map $f_2$;

• $f_3 : \mathbf{P} \to \mathbf{G}$, called genotypic survival, that describes the effects of natural selection (and migration) on the genotypes’ space $\mathbf{G}$;

• $f_4 : \mathbf{G} \to \mathbf{G}$, called mutation, that models the effects of recombination and mutation, encompassing all the genetic changes in a generational step.

Figure 1.1: A generation step in the model of Lewontin and Atmar: a generation is seen as the the composition of four maps between the population genotypes’ space $\mathbf{G}$, the population phenotypes’ space $\mathbf{P}$, and the space $\mathbf{I}$ of the environmental interaction with the genotypes’ space for the development of the individuals’ phenotypes.
Natural selection is often seen as the predominating evolutionary force, that leads to the maintenance or increase of the populations’ fitness, where fitness is the measure of the ability of an ensemble of individuals to adapt and survive in a specific environment (Hartl and Clark, 1989). Selection operates in the face of phenotypic variation that is mainly given by replication, reproduction and mutation of the individuals belonging to a population.

To summarise the interaction of the principal concepts in the Neo-Darwinist theory (Mayr, 1963) we cite David B. Fogel (2000): “Evolution most aptly refers to changes in gene frequencies. Gene frequencies persist only as the result of selection on the phenotype. The genetic variation of individuals is largely a chance phenomenon, primarily a product of recombination and ultimately of mutation. Selection is probabilistic, and its primary target is the individual. Sexual reproduction allows for the rapid phenotypic exploration of an underlying genetic theme”.

1.2 Prehistory and Origins of Evolutionary Computation

Since the early beginning of AI, simulated evolution has been proposed as a process to generate intelligent behaviors and machine learning. Already Alan Turing in 1950 stated that there is “an obvious connection between [machine learning] and evolution” (Turing, 1950). Friedman (1959) argued that thinking machines could be obtained by simulating mutation and selection dynamics; in particular, he remarked that chess-playing programs could be derived in this way.

The first attempts of implementing these ideas on a computing device can already be found in the late 1950s. George Box (1957) proposed a technique called Evolutionary Operation (EVOP) for manufacturing plants as a management process. The basic idea was to replace the static operation of a process by a continuous and systematic scheme of slight perturbations in the control variables. The effect of these perturbations was evaluated by the votes of a committee of technical managers and the process was shifted in the direction of improvement.

In 1958 Friedberg attempted to solve fairly simple problems by teaching a computer
to write computer programs. His system, called Learning Machine, learned by using what looks a lot like a modern mutation operator, random initialization of the individual solutions and random changes in the instructions (Friedberg, 1958; Friedberg et al., 1959). This approach was strongly criticised by Minsky (1961), who called Friedberg’s research “a comparable failure. [...] The machine did learn to solve some extremely simple problems. But it took on the order of 1000 times longer than pure chance would expect”. Even if this criticism was exaggerated (in fact Friedberg’s system performances were only 10 times longer than pure chance would expect and it could consistently outperform random search) and if Minsky himself admitted that “in the last section we see some real success obtained by breaking the problem into parts and solving them sequentially”, Friedberg’s subsequent successes were overshadowed by the criticism of his early research.

The first to recognise artificial evolution as an optimisation process was Bremermann (1958), who also argued that “the principle of evolution [...] is most useful as a key to the understanding of creative thinking and learning”. His optimisation technique searched over a response surface determined by the fitness of each possible arrangement of the considered genotype. He used a binary encoding of the possible solution, and calculated the fitness of a tentative solution as the number of bits in its genotypical representation that differed from the target bit string. Bremermann used probabilistic binary mutation to perturb a tentative solution, and argued that the mutation rate that achieves the most rapid improvement is obtained when the mutation was such that “on the average in each generation in each individual one bit is changed” (Bremermann, 1958). He also conjectured that an operator of sexual mating of individuals would overcome those situations in which asexual evolution is stagnant. He even tried some mating schemes, without obtaining any “spectacular result” (Bremermann, 1962).

The first successful attempts to use a recombination operator in a simulated evolutionary process were done by Barricelli (1954) and Reed et al. (1967). They conducted a series of evolutionary gaming experiments that centered around a simplified game of poker in which individuals were coding probabilistic betting strategies. They concluded that recombination “greatly enhances the speed of selective adaptation, particularly if interactions between the expressions of different hereditary factors is avoided” (Reed et al.,
1967).

In these early attempts we can see how some evolutionary concepts have been employed. But it’s only in the mid-1960s that we observe the birth of EAs, when John Holland (1969, 1975), of the University of Michigan, introduced Genetic Algorithms (GAs), Lawrence Fogel and his colleagues (Fogel, 1962; Fogel et al., 1966), of the University of California in San Diego, started their experiments on Evolutionary Programming (EP), and Ingo Rechenberg and Hans-Paul Schwefel (Rechenberg, 1965; Schwefel, 1965; Rechenberg, 1973), of the Technical University of Berlin, independently began to work on Evolution Strategies (ESs). Their pioneering works eventually gave rise to a broad class of optimisation methods particularly well suited for hard problems where little is known about the underlying search space. The last development of this research thread is the Genetic Programming (GP) introduced by John Koza (1992), of Stanford University, at the beginning of the 1990s. It’s only in the late 1990s, with the birth of common initiatives and conferences like the Genetic and Evolutionary Conference (GECCO) and the Congress on Evolutionary Computation (CEC) that all these approaches were grouped under the common name of Evolutionary Algorithms.

Lawrence Fogel conceived intelligent behavior as requiring the composite ability to predict the environment coupled with a translation of the predictions into a suitable response in light of a given goal. For the sake of generality, a sequence of symbols taken from a finite alphabet was used as the environment, and a population of finite state machines was evolved in order to operate on the sequence of symbols so far observed. These machines produce an output symbol that is likely to maximise the prediction of the next symbol in the sequence. New tentative solutions were produced by randomly mutating the population’s individuals by means of five different operators randomly chosen with uniform distribution. The produced offspring were evaluated on the same environment sequence of their parents, and the best individuals between parents and offspring were retained for the population of the next generational step.

If Lawrence Fogel’s EP was an attempt to produce intelligent behaviors, Rechenberg and Schwefel’s ESs were proposed to solve continuous function optimisation problems, in particular for aeronautical engineering problems. They addressed the problem of finding
the minima of a continuous real-valued function by evolving a population of points in the domain space. New offspring were produced randomly mutating the selected parents by adding a Gaussian random variable with zero mean and preselected standard deviation to each coordinate of the point. The mutation standard deviation is included in the representation of an individual and can undergo mutation like all the other coordinates of the point. Even if in their first approach they focused on a single parent – single offspring search (the (1 + 1)-ES), Schwefel (1981) developed the use of multiple parents and offspring populations, with the (λ + μ)-ES and the (λ, μ)-ES, in which λ parents produce μ offspring by mutation and, respectively, they compete or not with them for creating the new population.

The use of a recombination operator was especially emphasized by John Holland when proposing the GAs, probably the most known family of EAs. His idea was to design an efficient search mechanism in artificially adaptive systems, by evolving a binary encoded population of tentative solutions to a given problem. The evolution was done using probabilistic selection in the parents’ population, as well as recombination and random mutation of the selected individuals to produce the offspring population. The recombination was designed as a two-parent operator that produced two offspring by exchanging a randomly chosen part of the parents’ representations.

John Koza’s GP is the more recent evolutionary approach which extends the genetic model of learning to the space of possible computer programs. It is a major variation of GAs in which the evolving individuals are themselves computer programs instead of fixed length strings from a limited alphabet of symbols. Genetic programming is a form of program induction that can be used to automatically discover programs that solve or approximately solve a given task. Individual programs in GP may normally be expressed in any current programming language. However, the syntax of most languages is such that GP operators would create a large percentage of syntactically incorrect programs. For this reason, Koza chose a syntax in prefix form analogous to LISP and a restricted language with an appropriate number of variables, constants and operators defined to fit the problem to be solved. In this way syntax constraints are respected and the program search space is limited.
1.3 Basic Principles of an Evolutionary Algorithm

Employing the concepts of evolution by natural selection described in Section 1.1, EAs make use of a metaphor whereby an optimisation problem takes the place of the environment, feasible solutions are viewed as individuals living in that environment and an individual’s degree of adaptation to its surrounding environment is the counterpart of the objective function evaluated on a feasible solution, giving the fitness value of an individual. In the same way, a set of feasible solutions takes the place of a population of organisms.

Once an initial population has been created, the evolutionary algorithm enters a loop. At the end of each iteration a new population is created by applying a certain number of stochastic operators to the previous population. One such iteration is referred to as a generation or a time step. To reproduce the dynamics of biological evolution, three main operators are applied to a population in a generation: selection, variation, and replacement. These operators usually work on the genotypical representation of the individuals, as depicted in Figure 1.2, where the Lewontin-Atmar model for biological evolution (Section 1.1 and Figure 1.1) is modified to describe artificial evolution.

A generational step is therefore the result of the composition of three genetic operators: first a parents’ population is selected according to the fitness values of the individuals. Once the population of parents has been extracted, offspring for the next generation is produced through the application of a number of reproduction operators. The latter can involve just one parent (thus simulating asexual reproduction), in which case we speak of mutation, or more parents (thus simulating sexual reproduction), in which case we speak of recombination.

Recombination produces new individuals in combining the information contained in the parents. Depending on the representation of the individuals’ encoding, there can be two kinds of recombinations: binary valued and real valued. To give some examples of recombination operators, we can mention the following operators working on binary representations:

- **one-point crossover**, where a unique one position is selected uniformly at random and the genotypes are exchanged between the two individuals from this point;
Figure 1.2: A generational step in the model of Lewontin and Atmar modified for artificial evolution: a generation is seen as the composition of three operators selection, variation, and replacement working in the space $G$ containing the populations of genotypes; each population in $G$ is mapped to a point in the space $P$ of the populations of phenotypes.

- multi-point crossover, where several crossover positions are chosen at random with generally no duplicates and sorted in ascending order. The genotypes between successive crossover points are exchanged between the two parents to produce two new offspring. The first section is usually not exchanged between individuals. The idea behind multi-point crossover and many other 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 (Booker, 1987). Furthermore, the disruptive nature of multi-point
crossover seems 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 (Spears and Jong, 1991).

- **uniform crossover.** The scheme explained above is generalized to make every position of the genotype a potential crossover point (Syswerda, 1987). A *crossover mask*, having the same length as the individual structure, is created at random and the parity of each bit in the mask indicates which parent will supply the offspring with which bits. 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 overcome the bias in single-point crossover towards short substrings without requiring precise understanding of the significance of the specific bits in the individuals’ representation.

Mutation is ruled by the *mutation probability* or *mutation rate*. Two different approaches exist; during the EA generations, this mutation rate can either stay constant or change (i.e., increase or decrease). There is no general rule. In nature this can be spontaneous or depend on environmental factors such as pollution or radiation. In EAs 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 flipping variable values for every variable has only two states. For every individual, the variable value to change is randomly chosen. The mutation of a position will of course depend on the previously defined mutation probability.

As seen in Section 1.1, natural selection plays a central role in all modern evolutionary theories. In this dissertation we’ll mainly concentrate on the effects of this operator, which will be presented and discussed in detail in the next section.

### 1.4 Artificial Selection

In an evolutionary process, the variation operators are in charge of producing new tentative solutions to the environment, breeding and/or mutating individuals in the population
under evolution. The choice of the individuals that have to undergo variation is done by selection in both natural and artificial systems. This operator clearly plays a central role in the evolutionary process by inducing a direction to the search of the tentative solutions to the environment by means of the fitness values of the population’s individuals.

Selection is therefore the only operator in which the fitness values of the individuals influence the dynamical process. In biological systems the fitness of the individuals can only be measured indirectly by the number of offspring that they produce for the next generational step. In artificial evolutionary systems, fitness is a direct, well defined and measurable property of the individuals. We can actually argue that “in Evolutionary Algorithms fitness is measurable and implies the survival and reproduction behavior, which is just the opposite in biological reality” (Bäck, 1996).

The influence of this operator on the evolutionary dynamics has always deserved a great deal of attention since it determines the explorative/exploitative character of the process. Bäck (1996) differentiates between volume-oriented and path-oriented search processes, corresponding to low, respectively high, directedness in biological terminology, when the search character induced by the selection mechanism results in a more explorative or exploitative dynamics of the evolutionary process. He also points out that selection pressure, that will be introduced and discussed in Chapter 3, is “the instrument which enables the user of an EA to influence these characteristics of the search” (Bäck, 1996).

1.4.1 Fitness Proportionate Selection

Fitness proportionate selection was derived by Holland as the optimal trade-off between exploration and exploitation using an analogy with the $k$-armed bandit (Holland, 1975). Also known as stochastic sampling with replacement (Baker, 1987) and roulette-wheel selection, it is the simplest selection scheme. This algorithm maps each individual on a roulette-wheel, with share sizes proportional to the individual’s fitness. The roulette is then “spun”, offering each individual a chance to be selected for recombination. The process is repeated until the desired number of individuals is obtained (called mating population). Despite being rigorously derived and having a deep justification in Decision Theory, fitness proportionate selection has some drawbacks.
For instance, consider two fitness functions $f_1(x)$ and $f_2(x) = f_1(x) + c$, for all individuals $x$ in the search space. Since they appear to be substantially equivalent, one would expect the behavior of an evolutionary algorithm not to change when replacing one with the other; however, if the selection scheme is fitness proportionate this is obviously false, depending on the value of the constant $c$ with respect to the maximal and the minimal fitness values of the possible solutions in the search space.

Another difficulty is represented by so-called superindividuals. A superindividual in a population is an individual $y$ such that $f(y) \gg f(x)$ for all individuals $x$ in the population. A superindividual is allocated by fitness proportionate selection a prominent slice of copies in the next generation and, in a matter of a few generations, ends up overwhelming any other genotypes initially in the population, thus causing convergence. If the superindividual corresponds to the problem's global optimum, this is exactly what is desired. However, if it is associated with a local optimum, this leads to a failure of the algorithm, called premature convergence.

The list of problems does not end here. The push toward improvement provided by fitness proportionate selection asymptotically tends to zero as the individuals in a population approach the optimum. While this in fact reflects what is observed in nature, in the framework of optimisation it amounts to an actual drawback.

In order to solve these difficulties, two approaches are possible: either appropriately modifying the fitness function or, more simply, resorting to alternative selection schemes.

### 1.4.2 Linear Ranking Selection

Linear ranking selection (Baker, 1985) is based on a sorting of the individuals in the selection pool by decreasing fitness. It is similar to the roulette-wheel selection, but this time, share sizes on the roulette-wheel are not proportional to the individuals’ fitnesses anymore, but rather to their rankings with respect to the other individuals’ fitness values in the population. Once again, the roulette is “spun”, offering each individual a chance to be selected for recombination. The process is repeated until the desired number of individuals is obtained (called mating population). The probability the $i$th individual in
the ranking is extracted is thus defined, for $i = 1, \ldots, n$, as

$$p(i) = \frac{1}{n} \left[ \beta - 2(\beta - 1) \frac{i - 1}{n - 1} \right],$$

where $0 \leq \beta \leq 2$ is a parameter that can be interpreted as the expected sampling rate of the best individual across $n$ independent extractions with re-insertion.

### 1.4.3 Local k-Tournament Selection

Local k-tournament selection (Brindle, 1981) extracts $k$ individuals from the population with uniform probability but without re-insertion and makes them play a “tournament”. In the deterministic case, the winner is the fittest individual among the participants. However, the tournament may be probabilistic as well, in which case the probability for an individual to win is generally proportional to its fitness.

Selective pressure is directly proportional to the number $k$ of participants in a tournament: for $k = n$, the population size, deterministic local tournament selection degenerates into truncation selection with parameter $\tau = \frac{1}{n}$ (see below), whereas probabilistic local tournament selection degenerates into fitness proportionate selection.

### 1.4.4 Truncation Selection

Truncation selection (Mühlenbein and Schlierkamp-Voosen, 1993) has its inspiration in the science of breeding, a branch of applied statistics, and the main concepts that it relies on are the correlation between parent and offspring and the inheritance coefficient.

As Mühlenbein and Schlierkamp-Voosen point out, there is no major difference between breeding natural organisms and solutions to a problem. A minor difference is that, in the latter case, it is possible to control the genetic operators of mutation and recombination and modify them in order to get the greatest advantage out of them.

The most interesting aspect of selection for a breeder is the response to selection $R$, defined as the difference between the average fitness in two subsequent generations:

$$R_t = f(P_{t+1}) - f(P_t).$$
Breeders measure selection through the *selective differential* $S$, defined as the difference between the average fitness of the individuals selected for reproduction and the average fitness of the entire population.

Truncation selection consists in selecting just the best individuals and discarding the rest. The selective differential depends on the proportion $\tau$ of the population that gets selected: the smaller $\tau$, the greater $S_t$.

### 1.4.5 Classification of Selection Schemes

A possible taxonomy of selection schemes is the following, whereby a selection scheme can be classified according to at least four independent axes (Bäck and Hoffmeister, 1991):

- **dynamic – static**, depending on whether or not the selection probabilities take into account the fitness values actually present in the population, varying across generations: fitness proportionate selection is thus dynamic, while linear ranking, local $k$-tournament and truncation selection are static;

- **preservative – extinctive**, depending on whether or not it guarantees a non-zero probability of being selected to every individual: thus, fitness proportional selection is preservative, truncation and deterministic local $k$-tournament selection are extinctive, and linear ranking selection is preservative for $\beta < 2$ and extinctive for $\beta \geq 2$;

- **elitist – pure**, depending on whether or not it guarantees the survival of the best individual unchanged into the next generation.

- **generational – steady-state**, depending on whether or not the set of parents is determined once and remains fixed until a new population of offspring has been produced, or the parents are extracted at different times and their offspring are introduced into the population as they are produced.

This last classification axis introduces one of the two main subjects under investigation in this thesis: the temporal dimension in evolutionary processes. This item, together with spatial dimension, will be introduced in the next chapter.
Chapter 2

Space and Time

If what you have to deal with becomes too big, you cannot avoid structure.
– Frank Plumpton Ramsey

When designing an artificial evolutionary system, most of the time researchers focus their attention on setting the parameters of the variation and selection operators. In particular, a lot of care is devoted to the different crossover and mutation probabilities, and to the choice of the selection mechanism. This attention is mainly due to the observation that the set of parameters in an evolutionary algorithm drastically influences the dynamics of the system.

Moreover, both theory and applications have tended to focus mainly on mixing populations, also called panmictic populations. These are populations in which there is no particular structure, or even better, the underlying structure is a complete graph: any member of the population is equally likely to “meet” any other member. But Darwin realised long ago that populations may have a spatial structure and that the latter may have an influence on population dynamics. For instance, he remarked how some species, when isolated on islands, evolved differently from others that lived in more open environments. If we look around us, we too find that geographical separation factors have helped shape evolution. There are many examples, but one well-documented case is the spread of genomic traits in human populations due to geographical separation followed by migrations and mixing (Cavalli-Sforza et al., 1994). Also, as early as the 1960s, the book
by Macarthur and Wilson (1967) used models of geographical separation and migration to explain the spread and extinction of species. Thus, complete panmixia, although it can be achieved in laboratory, appears only as a limiting case in nature, where spatial-separation effects seem to play an important role.

Today, it appears very clearly that topological structure largely determines the dynamical processes that can take place in complex systems. The study of both the structure and the dynamics of a given system is thus mandatory if one wants to understand and possibly exploit all the possibilities. Why then has evolutionary computation concentrated on mixing populations? One possible answer is that using a mixing population is both easier and good enough for many purposes and the mathematical analysis of the dynamics of such populations is also usually easier. The next question is, thus: do spatially structured populations have a role in Evolutionary Computation (EC), i.e., are they a worthy object of study? We believe that the answer to this question should be positive. On the one hand, spatially structured populations are not new in EC: they have been much less studied than panmictic ones, but the empirical results available to date show beyond doubt that they can have beneficial effects. Secondly, there is actually only a very low cost associated with setting up and running a structured population instead of a panmictic one. Sometimes the cost of some genetic operations is even lower than that for a mixing population. So, experimenting with structured populations is generally easy and does not entail fundamental changes to the EC paradigm. On the other hand, spatially structured population models and their dynamical behaviors are interesting objects in themselves; their interest goes beyond their mere utility as an empirical way of improving results in EC. This will be seen particularly in conjunction with some irregular population structures that have been found to be extremely significant in many fields.

Often, the main motivation for using structured evolutionary algorithms (EAs) is the possibility of physically distributing computational tasks over different machines, having these tasks executed in parallel. This is common practice and is an advantage as far as computing times are concerned, especially for heavy-duty computations. However, the models and their implementations are two orthogonal factors. One may very well have a potentially parallelizable structured EA model and run it on standard sequential hardware.
2.1 Cellular Evolutionary Algorithms

Structured populations are just populations in which any given individual has its own neighborhood, which is smaller, often much smaller, than the size of the population. In other words, instead of all the other individuals in the population being considered as potential mates as it is the case in panmictic populations, only those that are in the same neighborhood can interact. Although this “isolation by distance” is often associated with geographical separation, this is not strictly required in evolutionary-algorithm (EA) models where only the “relational aspect” matters. In fact, there are many examples of biological niches and isolated or semi-isolated populations in biology where physical distance is the key factor keeping these demes nearly independent of one another. And many such biologically inspired models have been proposed for EAs. However, what counts is the neighborhood relationship, and this can be of any type, as long as it makes algorithmic sense. We shall see many examples of this in the following chapters. Thus, we are led to the conclusion that the important idea is the ensemble of relations among individuals, be they truly spatial or not. The mathematical objects that are required for describing this state of affairs are graphs. This means that we do not always need the concept of a metric space and an associated distance, such as the Euclidean distance. More often, distances between two individuals will be given by the network itself, as measured along the path that links them in the graph. In the realm of population graphs, and with
a slight abuse of language, we will often call these population structures *topologies*, be they truly spatial or not.

Cellular evolutionary algorithms (cEAs) use populations that are structured according to a topology. The structure may be an arbitrary graph, but more commonly it is a one- or two-dimensional regular lattice. This kind of spatially structured EA has been introduced by Gorges-Schleuter (1989), and Manderick and Spiessens (1989). A cEA maintains a population whose individuals are spatially distributed in cells. Each cell is occupied by one individual; therefore, the terms *cell* and *individual* may be used interchangeably without possibility of confusion. A cEA starts with the cells in a random state and proceeds by successively updating them using evolutionary operators until a termination condition is met. Updating a cell in a cEA means selecting two parents in the individual’s neighborhood, applying variation operators to them, and finally replacing the individual if the obtained offspring has a better fitness (other replacement policies can also be used).

Let us call $S$ the (finite) set of states that a cell can be in or, equivalently, the set of different individuals that can occupy a cell at any given time: this is the set of points in the (discrete) search space of the problem. Let $N_i$ be the set of neighbors of a given cell $i$, and let $|N_i| = N$ be its size. Note that all the cells in the lattice have identical neighborhood geometry and size. The local transition function $\phi(\cdot)$ can then be defined as:

$$\phi : S^N \rightarrow S,$$

which maps the state $s_i \in S$ of a given cell $i$ into another state from $S$, as a function of the states of the $N$ cells in the neighborhood $N_i$. Thus, the implicit form of the stochastic transition function $\phi(\cdot)$ is:

$$\phi(\cdot) = P\{x_i(t+1) \mid x_j(t) \in N_i\},$$

where $P$ is the conditional probability that cell $x_i$ will assume a certain value from the set $S$ at the next time step $t+1$, given the current (time $t$) values of the states of all the cells in the neighborhood. We are thus dealing with probabilistic automata (Tomassini, 1993; Whitely, 1993), and the set $S$ should be seen as a set of values of a random variable. The
2.1. CELLULAR EVOLUTIONARY ALGORITHMS

probability $P$ will be a function of the particular selection and variation methods.

This evolutionary system can be described in a more formal way: let's consider a
discrete time $t$ and a population $P(t)$ of $n$ individuals. To each individual are associated
a state and a location: $P(t) = \{a_1(t), a_2(t), \ldots, a_n(t)\}$, with $a_i = (s_i(t), l_i)$, where $s_i(t) \in S = \{s_1, s_2, \ldots, s_m\}$, the set of the possible states of the individuals, and $l_i \in L = \{l_1, l_2, \ldots, l_n\}$, the set of the locations of the individuals in the structure of the population.

If we denote with $T$ the product space between the space of the possible states and the
space of the possible locations of the individuals ($T = S \times L$), a population of size $n$ is an
element of $T^n$.

A fitness function $F : T^n \rightarrow \mathbb{R}^n$ is given such that each population $P(t) = \{a_1(t), a_2(t), \ldots, a_n(t)\}$ is associated to a vector $f(t) = F(P(t)) = (f_1(t), f_2(t), \ldots, f_n(t))$ with $f_i(t)$
being the fitness value of individual $a_i(t)$.

The selection mechanism is described by a function $Sel : T^n \times \mathbb{R}^n \rightarrow S^n$ such that
$(s'_1(t), s'_2(t), \ldots, s'_n(t)) = Sel(P(t), F(P(t)))$. For each individual in the population, it
selects the state of an individual in his neighborhood. Note that only the state of an
individual is selected, since the location of the selected individual doesn't influence the
successive crossover. On the contrary, the location of the selected individual influences the
function $Sel$, since it determines the selection pool for each location in the structure.

The topology of the structure thus affects the selection function, but not the successive
variation operators.

The state of the individual in the considered location is then combined with the selected
state by a function $Op : S \times S \rightarrow S$, producing the state of the individual in the next
generation for the considered location. If only a crossover operator is used the function
$Op$ can be represented in the form of an $n \times n$ matrix of elements of $S$.

Given the topology of the structure, the set $L$ of the possible locations of the individuals, the set $S$ of the possible states of the individuals, the fitness function $F$, the
selection function $Sel$, the recombination function $Op$, and the population $P(t)$, the population $P(t+1) = \{a_1(t+1), a_2(t+1), \ldots, a_n(t+1)\}$ at the next generation is formed by
individuals $a_i(t+1) = (s_i(t+1), l_i)$ such that $s_i(t+1) = Op(s_i(t), s'_i(t))$.

Since graphs are a suitable mathematical description for structured populations, in
the next section we will give a short introduction to the relevant concepts and definitions that will be used in the rest of the dissertation. Graph theory is a well-developed branch of discrete mathematics and it would be impossible, and also useless, to try to give an account of it here. Instead, we will limit to the introduction of the concepts that are really useful to us.

2.1.1 Useful Definitions for Graphs

For ease of reference, we collect here a few definitions and some nomenclature for graphs that are used throughout this work: a more detailed account can be found for example in Newman (2003) and Watts (1999).

Let \( V \) be a non-empty 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. In this thesis we only deal with undirected graphs.

When two 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 composing it. The shortest path between two vertices \( u \) and \( v \) is the path with the smallest length joining \( u \) to \( v \).

A graph is said to be connected if a path exists between any two vertices. A completely connected undirected graph \( G \) with \(|V| = n\) vertices has an edge between any two vertices. In this case, 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 by Albert and Barabasi (2002).
Four statistics are particularly useful for the graphs we’ll have to deal with in this dissertation: the average degree described above, the \textit{clustering coefficient}, the \textit{characteristic path length}, and the \textit{degree distribution}. They are briefly described below and in more detail by Albert and Barabasi (2002).

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 $C_j$ of a node $j$ is defined as the ratio between the $E$ edges that actually exist between the $k$ neighbors and the number of possible edges between them:

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

The cluster coefficient $C$ of a graph is then the mean of the clustering coefficients $C_j$ of all nodes $j$ in the graph.

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 one-dimensional structure, $C$ is given by the following formula:

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

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 Watts (1999) 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 $k$ incident edges. For a random graph $P(k)$ is a binomial peaked at $P(\langle k \rangle)$. However, most real networks do not show this kind of behavior. In particular, in scale-free graphs which seem to be common in real-life (Albert and Barabasi, 2002), $P(k)$ follows a power-law distribution: $P(k) = c k^{-\gamma}$, with $c$ and $\gamma$ being positive constants.
2.2 The Time Dimension

Updating a cell (individual) in a cellular EA means selecting two parents in the individual’s neighborhood (including the individual itself), applying variation operators to them, and finally replacing the individual with the best offspring. In a conventional synchronous cEA, all the individuals in the grid are updated simultaneously. This step makes up a generation, and the process is repeated until a termination condition is reached.

There exist many ways for sequentially updating the cells of a cEA. In this dissertation we’ll employ step-driven updates and ignore the so-called time-driven methods in which (real) time is explicit. Time-driven methods are more realistic for physical simulation but are not needed in the EA case (an excellent discussion of asynchronous update in CAs is available in (Schönfisch and de Roos, 1999)).

The most general update scheme is independent random ordering of updates in time, which consists of randomly choosing the next cell to be updated with replacement. This corresponds to a binomial distribution for the update probability. This update policy will be called uniform choice (UC) in the following and it is similar to the time-driven Poisson update in the limit of large $n$, $n$ being the population size.

In our study we also consider three other update methods: fixed line sweep, fixed random sweep, and new random sweep (we employ the same terminology as in (Schönfisch and de Roos, 1999)).

- In fixed line sweep (LS), the $n$ cells are updated sequentially from left to right and line after line starting from the upper left corner cell. This update can be used only for regular lattice structured populations.

- In fixed random sweep (FRS), the next cell to be updated is chosen with uniform probability without replacement; this will produce a certain update sequence $(c^1_q, c^2_q, \ldots, c^n_q)$, where $c^p_q$ means that cell number $p$ is updated at time $q$ and $(j, k, \ldots, m)$ is a permutation of the $n$ cells. The same permutation is then used for all update cycles.

- The new random sweep method (NRS) works like FRS, except that a new random
cell permutation is used for each sweep through the array.

A *time* or *generational step* is defined as \( n \) sequential updates, which corresponds to updating *all* the \( n \) cells in the grid for the synchronous and the asynchronous LS, FRS and NRS, and possibly less than \( n \) different cells in the asynchronous UC method, since some cells might be updated more than once.
Part II

Artificial Evolution on Structures
Chapter 3

Selection Pressure

If a man will begin with certainties, he shall end in doubts; but if he will be content to begin with doubts, he shall end with certainties.

– Francis Bacon

Changing time and space dimensions in an evolutionary algorithm has a main influence on the selection operator dynamics. In fact, the variation and mutation mechanisms operate on the individuals that have been previously selected, while selection is directly influenced by the selection pool of each individual, which is in turn determined by the topology of the population, and by the time dimension of the evolution. In the artificial evolution environment, the selection mechanism results in a key factor in the dynamics of the system, and different selection operators have been characterised by the selection pressures they induce.

In the next section we will introduce the concept of takeover time and its relation with selection pressure. Then, in Section 3.2, the mathematical notations used in the next chapters to model the selection pressure curves for different topologies and different update methods will be defined. Finally, in Section 3.3, the logistic modeling of selection pressure curves will be presented. We will also show why such a model is inadequate in the case of regularly structured populations.
3.1 Takeover Time

The takeover time is defined as being the time it takes for a single best individual to take over the entire population. It can also be seen as a simplified infection process, where an individual is infected when it is replaced by a copy of the best individual and where infected individuals cannot recover and stay infected forever. Takeover time can be estimated experimentally by measuring the propagation of the proportion of the best individual under the sole effect of selection, without any variation operator. A shorter takeover time indicates a higher selection pressure and thus more exploitative algorithms. By lowering the selection intensity\(^1\) the algorithm becomes more explorative. Theoretical takeover times have been derived by Goldberg and Deb (1991) for panmictic populations and for the standard selection methods. These times turn out to be logarithmic in the population size, except in the case of proportional selection, where it is slower by a factor of \(n\), \(n\) being the population size.

It has been empirically shown by Sarma and DeJong (1996) that the selection pressure induced on the entire population becomes weaker as we move from a panmictic to a square grid population identical in size, and where synchronous updating of the cells is used.

A study on the selection pressure in the cases of ring and array topologies in one-dimensional cEAs has been done by Rudolph (2000). Abstracting from specific selection methods, he splits the selection procedure into two stages: in the first stage, for each individual a neighbor is chosen, and then in the second stage, for each individual it is decided whether the previously chosen neighbor will replace it in the next time step. Using only replacement methods where extinction of the best by chance cannot happen, i.e., non-extinctive selection, Rudolph derives the expected takeover times for the two topologies as a function of the population size and the probability that in the selection step the individual with the best fitness is selected in the neighborhood.

In this part we complete and extend the previous investigations to comprise synchronous and asynchronous cell update modes for regular and irregular topologies, intro-

---

\(^1\)Note that the term selection intensity is used here rather informally, to express the degree of explorative or exploitative character of the algorithm, while in population genetics it is a rigorous concept (Mühlenbein and Schlierkamp-Voosen, 1993).
3.2 MATHEMATICAL MODELS

Let us consider the random variables $V_i(t) \in \{0, 1\}$ indicating the presence in cell $i$ ($1 \leq i \leq n$) of a copy of the best individual ($V_i(t) = 1$) or of a worse one ($V_i(t) = 0$) at time step $t$, where $n$ is the population size. The random variable

$$N(t) = \sum_{i=1}^{n} V_i(t)$$

denotes the number of copies of the best individual in the population at time step $t$. Initially, $V_i(1) = 1$ for a certain individual $i$, and $V_j(1) = 0$ for all $j \neq i$.

Following Rudolph’s definition (Rudolph, 2000), if the selection mechanism is non-extinctive, the expectation $E[T]$ with

$$T = \min\{t \geq 1 : N(t) = n\}$$

is called the takeover time of the selection method. In the case of spatially structured populations, the quantity $E_i[T]$, denotes the takeover time if cell $i$ contains the best individual at time step 1 and is called the takeover time with initial cell $i$. Assuming a uniformly distributed initial position of the best individual among all cells, the takeover time is therefore given by

$$E[T] = \frac{1}{n} \sum_{i=1}^{n} E_i[T].$$

In the following chapters, we give the recurrences describing the growth of the random variable $N(t)$ in a cEA with different regular and irregular topologies for the synchronous and the asynchronous update policies described in Section 2.2. We consider non-extinctive selection mechanisms that select the best individual in a given neighborhood with probabilities in the interval $(0, 1)$. 

ducing quantitative models for the growth of the best individual in the form of difference stochastic equations.
3.3 Limitations of the Logistic Model

It is well known since the work of Verhulst (1845) that the assumption of logistic growth is a reasonable model for biological populations within bounded resources (Murray, 2002). It is easy to see that this behavior also holds for the best individual growth in the artificial evolution of a finite panmictic population (Goldberg and Deb, 1991). In fact, if we consider a population of size $n$, the number $N(t)$ of copies of the best individual in the population at time step $t$ is given by the following recurrence:

$$
\begin{align*}
N(0) &= 1 \\
N(t) &= N(t - 1) + p_{sel} N(t - 1) (n - N(t - 1)),
\end{align*}
$$

where $p_{sel}$ is the probability that the best individual is chosen. This recurrence can be easily transformed into one that describes a discrete logistic population growth in discrete time:

$$
\begin{align*}
N(0) &= 1 \\
N(t) &= N(t - 1) + \left( p_{sel} n \right) N(t - 1) \left( 1 - \frac{1}{n} N(t - 1) \right).
\end{align*}
$$

Such a recurrence can be approximated in analytical form by the standard continuous logistic equation:

$$
N(t) = \frac{n}{1 + \left( \frac{n}{N(0)} - 1 \right) e^{-\alpha t}},
$$

where the growth coefficient $\alpha$ depends on the probability $p_{sel}$. This happens to be the approach taken in (Sarma and DeJong, 1997) for synchronous cEAs in order to fit the measured growth curves as a function of a single structural parameter.

This can be useful as a first approximation but, as suggested by Spiessens and Manderick (1991), the growth of individuals propagation in a two-dimensional grid under local fitness-proportionate selection should follow a quadratic law. Gorges-Schleuter (1999) made similar remarks, noting that in the artificial evolution of locally interacting, spatially structured populations, the assumption of a logistic growth does not hold anymore.

---

Note that this is not true in a rigorous sense. The discrete logistic map can give rise to chaotic behavior for a range of parameter values (Murray, 2002). This is ignored in the previous qualitative discussion.
Indeed, in these locally interacting structures, although the curves do have the familiar "S-shape" denoting growth followed by saturation, they are not exponential but rather polynomial with a time dependence $\propto t^d$, $d$ being the lattice dimension.

In fact, in the case of a ring or a torus structure, we have respectively a linear and a quadratic growth. In the next two chapters, we complete here their analysis which holds for unrestricted growth, extending it to bounded spatial populations synchronously and asynchronously updated.

For a structured population, let us consider the limiting case which represents an upper bound on growth rate and where the selection mechanism is deterministic (i.e. where $p_{sel} = 1$), and a cell always chooses its best neighbor for updating. If we consider a population of size $n$ with a ring structure where the two cells on the borders are linked and the neighborhood radius is $r$ (i.e., a neighborhood of a cell contains $2r + 1$ cells) the following recurrence describes the growth of the number of copies of the best individual:

$$\begin{cases} 
N(0) = 1 \\
N(t) = N(t-1) + 2r.
\end{cases}$$

This recurrence can be described by the closed equation $N(t) = N(0) + 2rt$, which clearly shows the linear character of the growth rate.

In the case of a population of size $n$ disposed on a toroidal grid of size $\sqrt{n} \times \sqrt{n}$ (assuming $\sqrt{n}$ odd) and a von Neumann generalized neighborhood structure of radius $r$, the growth of the number of copies of the best individual can be described by the following recurrence:

$$\begin{cases} 
N(0) = 1 \\
N(t) = N(t-1) + 4\sum_{i=0}^{r-1}(rt - i), \quad \text{for } 0 \leq t \leq \frac{\sqrt{n}-1}{2} \\
N(t) = N(t-1) + 4\sum_{i=0}^{r-1}(\sqrt{n} - rt - i), \quad \text{for } t > \frac{\sqrt{n}-1}{2},
\end{cases}$$

$$\begin{cases} 
N(0) = 1 \\
N(t) = N(t-1) + 4\sum_{i=0}^{r-1}(\sqrt{n} - rt - i), \quad \text{for } 0 \leq t \leq \frac{\sqrt{n}-1}{2} \\
N(t) = N(t-1) + 4\sum_{i=0}^{r-1}(\sqrt{n} - rt - i), \quad \text{for } t > \frac{\sqrt{n}-1}{2},
\end{cases}$$
which reduces to the following:

\[
\begin{align*}
N(0) &= 1 \\
N(t) &= N(t-1) + 4r^2t - 2r(r+1), \quad \text{for } 0 \leq t \leq \frac{\sqrt{n}-1}{2} \\
N(t) &= N(t-1) - 4r^2t + 4r\sqrt{n} - 2r(r+1), \quad \text{for } t > \frac{\sqrt{n}-1}{2}.
\end{align*}
\]

This growth is described by a convex quadratic equation followed by a concave one, as clearly shown by the two closed forms of the recurrence:

\[
\begin{align*}
N(t) &= 2r^2t^2 + 2r(2r+1)t + 1, \quad \text{for } 0 \leq t \leq \frac{\sqrt{n}-1}{2} \\
N(t) &= -2r^2t^2 + 2r(2\sqrt{n} - 3r - 1)t + 1, \quad \text{for } t > \frac{\sqrt{n}-1}{2}.
\end{align*}
\]

Figure 3.1 graphically depicts the growth described by the equations above for a population of 81 individuals disposed on a 9 × 9 torus structure using a radius 1 von Neumann neighborhood.

Thus, a more accurate modeling should take into account the non-exponential growth followed by saturation (crowding effect). In the following chapters, we address such study for the cases of one- and two-dimensional regular lattice topologies using synchronous and asynchronous evolutions.
Chapter 4

One-Dimensional Structures

Instead of asking a complicated question (as all psychologically relevant questions must be) and to find a simple answer (often in the form of ‘yes’, ‘no’, ‘perhaps’), one could try to ask a very simple question (as ‘what will a person do when facing two alternatives’) and derive a rich and complex avalanche of answers.

– Anatol Rapoport

In a one-dimensional cEA the individuals are arranged along a line. Depending on whether the last and the first ones communicate or not we have either a ring or an array topology. Here we consider the first case (ring), which is more common. Each of the $n$ individuals has the same number of neighbors on both sides, and this number depends on the radius $r$. We will first consider the simplest case, $r = 1$, which means that in each neighborhood there are three neighbors, including the central cell itself. In the next chapter, we will generalise the models described here to radius $r > 1$. For this topology, at any given time step $t$ the expected number of copies $N(t)$ of the best individual is independent from its initial position. Therefore, the expected takeover time is $E[T] = E_t[T]$, $\forall i$.

In the next four sections, we will present the models for the synchronous (Section 4.1), the asynchronous fixed line sweep (Section 4.2), the asynchronous fixed and new sweep (Section 4.3), and the asynchronous uniform choice (Section 4.4) updates. These models will be experimentally validated in Section 4.5 when using a binary tournament selection mechanism (Section 4.5.1) and a linear ranking selection operator (Section 4.5.2). Finally,
in the last section, we summarise the result and we will present the future perspectives this investigation opens.

4.1 Synchronous Takeover Time

Since we consider neighborhoods of radius 1, the set of cells containing a copy of the best individual will always be a connected region of the ring. Therefore at each time step, only two more cells (the two adjacent to the connected region of the ring) will contain a copy of the best individual with probability $p$ depending on the selection mechanism. The growth of the quantity $N(t)$ can be described by the following recurrence:

$$
\begin{cases}
N(0) &= 1 \\
E[N(t)] &= \sum_{j=1}^{n} P[N(t-1) = j] (j + 2p),
\end{cases}
$$

where $P[N(t-1) = j]$ is the probability that the random variable $N$ has the value $j$ at time step $t-1$. Since $\sum_{j=1}^{n} P[N(t-1) = j] = 1$, and the expected number $E[N(t-1)]$ of copies of the best individual at time step $t-1$ is by definition $\sum_{j=1}^{n} P[N(t-1) = j] j$, the previous recurrence is equivalent to

$$
\begin{cases}
N(0) &= 1 \\
E[N(t)] &= E[N(t-1)] + 2p.
\end{cases}
$$

The closed form of this recurrence is trivially

$$
E[N(t)] = 2pt + 1,
$$

therefore the expected takeover time $E[T]$ for a synchronous ring cEA with $n$ cells is

$$
E[T] = \frac{1}{2p} (n - 1).
$$

Rudolph (2000) gave analytical results for the ring with synchronous updating for a generic probability of selection $p$. Although obtained in a different way, the previous
expression and his equation give nearly the same results for large population sizes $n$. In fact, for large $n$ his equation reduces to $\frac{p}{2p} - \frac{1}{2}$, while our equation gives $\frac{p}{2p} - \frac{1}{2p}$. Given that the first term quickly dominates the second one when $n$ is large, the two expressions are considered equivalent.

4.2 Asynchronous Fixed Line Sweep Takeover Time

Let us consider the general case of an asynchronous fixed line sweep cEA, where the connected region containing the copies of the best individual at time step $t$ is $B(t) = \{l, \ldots, k\}$, with $1 < l \leq k < n$. At each time step, the cell $l-1$ will contain a copy of the best individual with probability $p$, while the cells $k+j$ (with $j = 1, \ldots, n-k$) will contain a copy of the best individual with probability $p^j$. The recurrence describing the growth of the random variable $N(t)$ is therefore

$$
\begin{align*}
N(0) &= 1 \\
E[N(t)] &= \sum_{j=1}^{n} P[N(t-1) = j] \left( j + p + \sum_{i=1}^{n-j} p^i \right).
\end{align*}
$$

Since $\sum_{i=1}^{n-j} p^i$ is a geometric progression, for large $n$ we can approximate this quantity by the limit value $p/(1-p)$ of the summation. The recurrence is therefore equivalent to the following one:

$$
\begin{align*}
N(0) &= 1 \\
E[N(t)] &= E[N(t-1)] + p + \frac{p}{1-p} = E[N(t-1)] + \frac{2p-p^2}{1-p}.
\end{align*}
$$

The closed form of the previous recurrence being

$$
E[N(t)] = \frac{2p-p^2}{1-p} t + 1,
$$

we conclude that the takeover time for an asynchronous fixed line sweep cEA with a population of size $n$ is

$$
E[T] = \frac{1-p}{2p-p^2} (n-1).
$$
4.3 Asynchronous Fixed and New Random Sweep Takeover Times

The mean behaviors of the two asynchronous fixed and new random sweep update policies among all the possible permutations for the sweeps are equivalent. We therefore give only one model describing the growth of the random variable $N(t)$ for both policies.

Let us again consider the general case for which the connected region containing the copies of the best individual at time step $t$ is $B(t) = \{l, \ldots, k\}$, with $1 < l \leq k < n$. The cells $l-1$ and $k+1$ have a probability $p$ of containing a copy of the best individual at the next time step. For symmetry reasons, let’s consider only the part of the ring at the right side of the connected region. The cell $k+2$ has a probability $1/2$ to be contained in the set of cells after cell $k+1$ in the sweep, so it has a probability $(p/2)p$ to contain a copy of the best individual in the next time step. In general, a cell $k+j+1$ has a probability $1/2$ to be after cell $k+j$ in the sweep, so it has a probability $(p/2)^j p$ to contain a copy of the best individual in the next time step. The recurrence describing the growth of the random variable $N(t)$, is therefore

$$
\begin{align*}
N(0) &= 1 \\
E[N(t)] &= \sum_{j=1}^{n} P[N(t-1) = j] \left( j + 2 \sum_{i=1}^{n-j} p \left( \frac{p}{2} \right)^{i-1} \right),
\end{align*}
$$

which can be transformed into the recurrence

$$
\begin{align*}
N(0) &= 1 \\
E[N(t)] &= \sum_{j=1}^{n} P[N(t-1) = j] \left( j + 4 \sum_{i=1}^{n-j} \left( \frac{p}{2} \right)^{i} \right).
\end{align*}
$$

Since $\sum_{i=1}^{n-j} (p/2)^i$ is a geometric progression, for large $n$ we can approximate this quantity by the limit value $p/(2-p)$ of the summation. The recurrence is thus equivalent to the following one:

$$
\begin{align*}
N(0) &= 1 \\
E[N(t)] &= E[N(t-1)] + \frac{4p}{2-p},
\end{align*}
$$
The closed form of the previous recurrence is

\[ E[N(k)] = \frac{4p}{2-p} k + 1, \]

and we conclude that the expected takeover time for a fixed (or new) random sweep asynchronous cEA with a population of size \( n \) is

\[ E[T] = \frac{2-p}{4p} (n-1). \]

### 4.4 Asynchronous Uniform Choice Takeover Time

To model takeover time for asynchronous uniform choice cEAs, it is preferable to use cell update steps \( u \) instead of time steps in the recurrences. As for the other update policies, the region containing the copies of the best individual at update step \( u \) is a connected part of the ring \( B(u) = \{l, \ldots, k\} \), with \( 1 < l \leq k < n \). At each update step, the two cells \( l-1 \) and \( k+1 \) have probability \( 1/n \) of being selected, and, if selected, each cell has a probability \( p \) to contain a copy of the best individual after the selection and the replacement phases. The recurrence describing the growth of the random variable \( N(u) \) counting the number of copies of the best individual at update step \( u \) thus becomes:

\[
\begin{align*}
N(0) &= 1 \\
E[N(u)] &= \sum_{j=1}^{n} P[N(u-1) = j] \left( j + 2 \frac{1}{n} p \right),
\end{align*}
\]

which can be transformed into

\[
\begin{align*}
N(0) &= 1 \\
E[N(u)] &= E[N(u-1)] + 2 \frac{1}{n} p.
\end{align*}
\]

We can easily derive the closed form of the previous recurrence:

\[ E[N(u)] = \frac{2}{n} p u + 1. \]
Since a time step is defined as \( n \) update steps, where \( n \) is the population size, the expected takeover time for a uniform choice asynchronous cEA in terms of time steps is

\[
E[T] = \frac{1}{2p} (n - 1).
\]

We notice that the expected takeover time for a uniform choice asynchronous cEA is equal to the expected takeover time for a synchronous cEA.

It should also be noted that the present asynchronous uniform choice update model is very similar to what goes under the name of nonlinear voter model in the probability literature (Durrett, 1988). As well, it can be considered analogous to a steady-state cellular EA with a generation gap of \( 1/n \).

### 4.5 Experimental Validation

In this section we provide a set of validation tests intended to demonstrate the accuracy of the developed mathematical models. Since cEAs are good candidates for using selection methods that are easily extensible to small local pools, we use binary tournament and linear ranking in our experiments. Fitness-proportionate selection could also be used but it suffers from stochastic errors in small populations (e.g., at a neighborhood level), and it is more difficult to model theoretically since it requires knowledge of the fitness distribution function.

We have used the binary tournament selection mechanism described by Rudolph (2000): two individuals are randomly chosen with replacement in the neighborhood of a given cell, and the one with the better fitness is selected for the replacement phase.

In linear ranking selection the individuals in the neighborhood of a considered cell are ranked according to their fitness: each individual then has a probability of

\[
\frac{2(s - i)}{s(s - 1)}
\]

of being selected for the replacement phase, where \( s \) is the number of cells in the neighborhood and \( i \) is its relative rank in the neighborhood.
For this study, the cEA structure has a ring topology of size 1024 with a neighborhood of radius 1. Only the selection operator is active: for each cell it selects one individual in its neighborhood (the cell itself and the two adjacent cells situated at its immediate right and left). The selected individual replaces the old one only if it has a better fitness.

### 4.5.1 Binary Tournament Selection

Figure 4.1 shows the experimental growth curves of the best individual for the synchronous and the four asynchronous update methods. We can notice that the mean curves for the two asynchronous fixed and new random sweep show a very similar behavior. The graph also shows that the asynchronous update methods give an emergent selection pressure greater than that of the synchronous case, growing from the uniform choice to the line sweep, with the fixed and new random sweep in between.

Figure 4.1: Takeover times with binary tournament selection: mean values over 100 runs. The vertical axis represents the number of copies $N(t)$ of the best individual in each population as a function of the time step $t$.

The numerical values of the mean takeover times for the five update methods, along with their standard deviations, are shown in Table 4.1, where one can see that the asynchronous fixed random sweep and new random sweep methods give results that are statistically indistinguishable, and can therefore be described by a single model, as assumed in Section 4.3. The same can be said for the synchronous and the asynchronous uniform choice methods, as our models predicted (see Sections 4.1 and 4.4).

Since we use a neighborhood of radius 1, at most one individual with the best fitness


Table 4.1: Actual mean takeover time and standard deviation of the tournament selection for the five update methods. Mean values over 100 independent runs.

<table>
<thead>
<tr>
<th></th>
<th>Synchro</th>
<th>LS</th>
<th>FRS</th>
<th>NRS</th>
<th>UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Takeover Time</td>
<td>925.03</td>
<td>569.82</td>
<td>666.18</td>
<td>689.29</td>
<td>920.04</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>20.36</td>
<td>24.85</td>
<td>17.38</td>
<td>20.27</td>
<td>26.68</td>
</tr>
</tbody>
</table>

will be present in the neighborhood of a considered cell, except for the last update when there are two of them. It turns out that the probability for an individual having a copy of the best individual in its neighborhood to select it is $p = 5/9$. Using this probability in the models described in Sections 4.1 to 4.4, we calculated the theoretical growth curves. Figure 4.2 shows the predicted and the experimental curves for the five update methods, as well as the mean square error (m.s.e.) between them.

Looking at the curves, and taking into account the small value of the m.s.e., it is clear that the models faithfully predict the observed takeover times and selection pressure curves. Moreover, the equivalence between asynchronous new and fixed random sweeps, as well as that of synchronous and asynchronous uniform choice, are fully confirmed.

4.5.2 Linear Ranking Selection

Figure 4.3 shows the experimental growth curves of the best individual for the synchronous and the four asynchronous update methods. We can observe in the linear ranking case the same behavior that previously emerged in the binary tournament case: the mean curves for the synchronous and the asynchronous uniform choice cases are superposed, and the mean curves for the two asynchronous fixed and new random sweep show very similar behaviors. The graph shows that the asynchronous update methods give an emergent selection pressure greater than that of synchronous one, growing from the uniform choice to the line sweep, with the fixed and new random sweep in between.

The numerical values of the mean takeover times for the five update methods, along with their standard deviations, are shown in Table 4.2. Once again, the results show that the two random sweep methods are statistically equivalent, which is also the case for the synchronous and uniform choice methods.
4.5. EXPERIMENTAL VALIDATION

Figure 4.2: Comparison of the experimental takeover time curves (full) with the model (dashed) in the case of binary tournament selection for four update methods: synchronous (a), asynchronous line sweep (b), asynchronous fixed random sweep (c), asynchronous new random sweep (d). Asynchronous uniform choice gives the same curve as the synchronous update, therefore it is omitted. In each figure the mean square error between the predicted and the actual curves is shown.

With this linear ranking selection method, a cell having a copy of the best individual in its neighborhood has a probability \( p = \frac{2}{3} \) of selecting it. Using this value in the models described in Sections 4.1 to 4.4, we can calculate the theoretical growth curves. Figure 4.4 shows the predicted and the experimental curves for the five update methods, and the mean square error between them. As it can be seen, the agreement between theory and experiment is excellent.
Figure 4.3: Takeover times with linear ranking selection: mean values over 100 runs. The vertical axis represents the number of copies $N(t)$ of the best individual in each population as a function of the time step $t$.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean Takeover Time</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchro</td>
<td>768.04</td>
<td>17.62</td>
</tr>
<tr>
<td>LS</td>
<td>387.09</td>
<td>19.21</td>
</tr>
<tr>
<td>FRS</td>
<td>519.92</td>
<td>14.26</td>
</tr>
<tr>
<td>NRS</td>
<td>541.14</td>
<td>14.48</td>
</tr>
<tr>
<td>UC</td>
<td>766.5</td>
<td>25.44</td>
</tr>
</tbody>
</table>

Table 4.2: Mean takeover time and standard deviation of the linear ranking selection for the five update methods. Mean values over 100 independent runs.

4.6 Summary

In this chapter, we have presented quantitative models for the takeover time in cellular evolutionary algorithms structured as a ring with nearest neighbor interactions only. New results have been obtained for asynchronous cell update policies. The models are based on simple difference probabilistic equations. We have studied two types of selection mechanisms that are commonly used in cEAs: binary tournament and linear ranking. With these selection methods, our results show that there is a good agreement between theory and experiment; in particular, we showed that asynchronous cell update methods enable an easier control on the selection intensity without using ad hoc parameters.
4.6. SUMMARY

Figure 4.4: Comparison of the experimental takeover time curves (full) with the model (dashed) in the case of linear ranking selection for four update methods: synchronous (a), asynchronous line sweep (b), asynchronous fixed random sweep (c), asynchronous new random sweep (d). Asynchronous uniform choice gives the same curve as the synchronous update, therefore it is omitted. In each figure the mean square error between the predicted and the actual curves is shown.
You often hear that the purpose of a scientific theory is to predict. That’s not correct. The purpose is understanding. Prediction is one way to test whether our understanding is correct. Simplicity, scope, and beauty are as important as prediction in judging whether a theory leads to understanding.

– David H. Sharp

We consider cEAs defined on a square lattice of finite size $\sqrt{n} \times \sqrt{n}$ with periodic boundary condition (i.e., a toroidal structure). The neighborhood we consider in this paper is the von Neumann neighborhood, which is constituted by a central cell plus the four first neighbor cells in the directions north, east, south, and west. Because of the torus wrap-up properties, at each time step $t$ the expected number of copies $N(t)$ of the best individual is independent from its initial position. Therefore, the expected takeover time is $E[T] = E_i[T], \forall i$.

When passing to regular structures with dimension greater than 1, the recurrences describing the selection pressure curves become quickly too complicated to be derived. In the next section we will introduce the geometrical approximations that we’ll use in this chapter to model the selection pressure curves for the synchronous (Section 5.2), the asynchronous fixed line sweep (Section 5.3), the asynchronous fixed and new sweep (Section 5.4), and the asynchronous uniform choice (Section 5.5) updates. These models will be experimentally validated in Section 5.6 when using a binary tournament selection mechanism (Section 5.6.1) and a linear ranking selection operator (Section 5.6.2).
two next section will be devoted to the extension of the models to two-dimensional regular structures with rectangular shapes (Section 5.7), and to the generalisation of the one- and two-dimensional models when the neighborhood radius is augmented (Section 5.8). Finally, in the last section we will summarise the result and we will present the future perspectives opened by this investigation.

5.1 The Geometrical Approximation

We have seen in section 3.3 the limiting case of the growth with a deterministic selection (i.e., a mechanism that selects the best individual in the neighborhood with probability \( p = 1 \)). It should be noted that, in that case, the time variable \( t \) in the equations determines the measure of the half diagonal of the 45 degrees rotated square (see Figure 3.1). When modeling a probabilistic selection method, the exact recurrences, as derived for the ring topology in the Chapter 4, become very complicated. In fact, as it can be seen in Figure 5.1, the phenomenon that has to be modeled implies different selection probabilities in different locations in the grid.

![Figure 5.1: Example of a probabilistic selection growth of \( N(t) \) for a population of 81 individuals on a 9 × 9 torus structure.](image)

Since we wanted to keep the models simple and easily interpretable, we decided to approximate the geometry of the propagation as the growth of a rotated square in the torus (see Figure 5.2).

![Figure 5.2: Geometric approximation of a probabilistic selection growth in a torus structured population: a rotated square grows as a function of time, starting as an unrestricted growth until the square reaches the edges of the grid, and then saturating the population.](image)

Using this geometric growth, we can approximate the measures of the side \( s \) and the
5.2 Synchronous Takeover Time

Assuming that the region containing the copies of the best individual expands keeping the shape of a 45 degrees rotated square, we can model the growth of $N(t)$ with the following recurrence:

$$
\begin{cases}
N(0) = 1 \\
N(t) = N(t-1) + 4p_2 \frac{N(t-1)}{\sqrt{2}}, \text{ for } N(t) \leq \frac{n}{2} \\
N(t) = N(t-1) + 4p_2 \sqrt{n - N(t-1)}, \text{ for } N(t) > \frac{n}{2}.
\end{cases}
$$

It is practically impossible to find the closed analytical form of these recurrences, as it will be the case for the asynchronous models of the next sections. Therefore, we only give the explicit recurrences in each case.

5.3 Asynchronous Fixed Line Sweep Takeover Time

This update method, which is meaningful in a ring topology, in the case of a toroidal topology can be criticized. In fact, there is no biological parallelism for this update mechanism. A precise model for such update would be very complicated, since it is difficult to approximate the shape of the region containing the copies of the best individual. We have therefore decided, to keep the model simple and understandable, to roughly approximate the shape of the region with a square stretched to the south-east direction.
growing with probability $p_1$ on the north-east side, $p_2$ on the south-east side, and $p_1$ in the south direction.

Let us suppose that in any line the cells containing a copy of the best individual at time step $t$ have index $l$ to $k$. In the next time step, the cell $l - 1$ will contain a copy of the best individual with probability $p$, while the cells $k + j$ (with $j = 1, \ldots, n - s$) will contain a copy of the best individual with probability $p^j$. The number of copies of the best individual in the considered line in the next time step is

\[
p + \sum_{i=1}^{\sqrt{n-j}} p^i.
\]

For large $n$ we can approximate this quantity by the limit $(2p - p^2)/(1 - p)$. Therefore, we can model the growth of $N(t)$ with the following recurrence:

\[
\begin{aligned}
N(0) &= 1 \\
N(t) &= N(t - 1) + \left(\frac{2p_2 - p_2^2}{1 - p_2} + 2\frac{2p_1 - p_1^2}{1 - p_1}\right) \sqrt{n - N(t - 1)}, \quad \text{for } N(t) \leq \frac{n}{2} \\
N(t) &= N(t - 1) + \left(\frac{2p_2 - p_2^2}{1 - p_2} + 2\frac{2p_1 - p_1^2}{1 - p_1}\right) \sqrt{n - N(t - 1)}, \quad \text{for } N(t) > \frac{n}{2}.
\end{aligned}
\]

### 5.4 Asynchronous Fixed and New Random Sweep Takeover Time

The behaviors of fixed random sweep and new random sweep averaged over all the possible permutations of the individuals on the grid are equivalent also in the toroidal case. We therefore give only one model describing the growth of the random variable $N(t)$ for both policies.

In one time step, following the geometrical approximation, the probability of one individual on the border of the region being taken over by the best is $p_2$, while an individual at distance 2 from the region can be replaced by the best if one or two of its neighbors have already been replaced during the sweep. One of its neighbors is replaced if:

- only one neighbor comes before in the sweep (and it has been replaced)
- two neighbors come before in the sweep but just one has been replaced
In this latter case, two of its neighbors are replaced if the two come before in the sweep and the two have been replaced. The average probability for a couple of individuals of being before one another in a sweep is $\frac{1}{2}$; therefore, an individual at distance 2 from the region is replaced with probability

$$2 \left( \frac{1}{2} \left( 1 - \frac{1}{2} \right) p_2 p_1 \right) + 2 \left( \frac{1}{2} \frac{1}{2} p_2 (1 - p_2) p_2 \right) + \frac{1}{2} \frac{1}{2} p_2^2 p_2 = p_2 p_1 + \frac{1}{4} p_2 - 2 p_1 p_2^2.$$ 

At distance 3 or more the same reasoning can be done, but we have decided to model the growth up to distance 2 because, as it can been seen in Figure 5.3, the probability at distances $\geq 3$ becomes negligible.

Thus, we can model the growth of $N(t)$ with the following recurrence:

$$\begin{cases} 
N(0) = 1 \\
N(t) = N(t-1) + 4 \left( p_2 p_1 + \frac{1}{4} (p_2 - 2 p_1) p_2^2 \right) \left( \sqrt{N(t-1)} - 1 \right) + 4 p_1, & \text{for } N(t) \leq \frac{n}{2} \\
N(t) = N(t-1) + 4 \left( p_2 p_1 + \frac{1}{4} (p_2 - 2 p_1) p_2^2 \right) \left( \sqrt{n - N(t-1)} - 1 \right) + 8 p_3, & \text{for } N(t) > \frac{n}{2}. 
\end{cases}$$

### 5.5 Asynchronous Uniform Choice Takeover Time

The ways in which an individual can be replaced in a time step for this update case are the same as for fixed and new random sweep (see before). In the present case, the average probability of an individual coming before a given other individual in a time step is $1/n$;
therefore, an individual at distance 2 from the region is replaced with probability:

\[
\frac{1}{n} p_2 p_1 + \frac{1}{n^2} (p_2 - 2p_1)p_2^2.
\]

The probability is already very small at distance 2 (see Figure 5.4). Thus, in our model we only take into account individuals at distance 1 from the region.

![Figure 5.4: Probability of an individual being replaced by a copy of the best individual (y axis) with respect to distance (x axis) from the region formed by copies of the best for Uniform Choice. Note that the curve is traced continuously for clarity but the probability is calculated only at discrete points.](image)

In terms of time steps, the growth of \( N(t) \) can be modeled with the following recurrence:

\[
\begin{cases} 
N(0) = 1 \\
N(t) = N(t-1) + 4p_2\sqrt{N(t-1)}, & \text{for } N(t) \leq \frac{n}{2} \\
N(t) = N(t-1) + 4p_2(\sqrt{n - N(t-1)} - 1) + 8p_3, & \text{for } N(t) > \frac{n}{2}.
\end{cases}
\]

5.6 Experimental Validation

In this section we provide a set of validation tests intended to demonstrate the accuracy of the developed mathematical models. Since cEAs are good candidates for using selection methods that are easily extensible to small local pools, we use binary tournament and linear ranking in our experiments. Fitness-proportionate selection could also be used but it suffers from stochastic errors in small populations (e.g., at a neighborhood level), and it is more difficult to model theoretically since it requires knowledge of the fitness distribution function.

We have used the binary tournament selection mechanism described by Rudolph
two individuals are randomly chosen with replacement in the neighborhood of a given cell, and the one with the better fitness is selected for the replacement phase.

In linear ranking selection the individuals in the neighborhood of a considered cell are ranked according to their fitness: each individual then has probability

\[
\frac{2(s - i)}{s(s - 1)}
\]

to be selected for the replacement phase, where \(s\) is the number of cells in the neighborhood and \(i\) is its rank in the neighborhood.

We now describe the validation of the models for the torus structure in the same conditions that we used for the ring topology in Chapter 4. The cEA structure has torus topology of size 32 × 32 with von Neumann neighborhood. Only the selection operator is active: for each cell it selects one individual in the cell neighborhood, and the selected individual replaces the old individual only if it has a better fitness. This study is addressed separately for the two selection methods, binary tournament and linear ranking, in the forthcoming two subsections.

### 5.6.1 Binary Tournament Selection

Figure 5.5 shows the growth curves of the best individual for the panmictic, the synchronous and three asynchronous update methods. In all cases the same set of parameters has been used. The mean curves for the two asynchronous methods, fixed and new random sweep, show a very similar behavior, so we have decided to plot only the new random sweep results. The graph shows that the asynchronous update methods give an emergent selection pressure greater than that of the synchronous case, growing from the uniform choice to the line sweep, with the fixed and new random sweep in between (similar to our findings for the ring topology, see Section 4.5).

The numerical values of the mean takeover times for the five update methods, together with their standard deviations are shown in Table 5.1, where it can be seen that the fixed random sweep and new random sweep methods give results that are statistically indistinguishable. However, this time the differences between the uniform choice and the
Figure 5.5: Takeover times with binary tournament selection: mean values over 100 runs. The vertical axis represents the number of copies $N(t)$ of the best individual in each population as a function of the time step $t$.

synchronous update are meaningful in a torus.

<table>
<thead>
<tr>
<th></th>
<th>Synchro</th>
<th>LS</th>
<th>FRS</th>
<th>NRS</th>
<th>UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Takeover Time</td>
<td>44.06</td>
<td>21.8</td>
<td>27.21</td>
<td>28.26</td>
<td>35.73</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>1.6746</td>
<td>1.7581</td>
<td>1.5654</td>
<td>1.8996</td>
<td>2.4489</td>
</tr>
</tbody>
</table>

Table 5.1: Mean takeover time and standard deviation of the binary tournament selection for the five update methods. Mean values over 100 independent runs.

Since we use a von Neumann neighborhood, the probabilities $p_1$, $p_2$, and $p_3$ of selecting the best individual when there are respectively 1, 2, and 3 copies of it in the neighborhood are respectively $9/25$, $16/25$, and $21/25$. Using these probabilities in the models, we calculated the theoretical growth curves. Figure 5.6 shows the predicted and the experimental curves for the five update methods. It can be observed that the agreement between theory and experiment is very good, in spite of the approximations made in the models.

### 5.6.2 Linear Ranking Selection

Figure 5.7 shows the growth curves of the best individual for the panmictic, the synchronous and three asynchronous update methods, using the same parameter set in all cases. We can observe in the linear ranking case the same behavior that emerged in the binary tournament case: the average curves for the two asynchronous updates, fixed and
new random sweep, show a very similar behavior. We have therefore decided to plot only the new random sweep results. The graph shows that the asynchronous update methods give an emergent selection pressure greater than that of synchronous one, growing from the uniform choice to the line sweep, with the fixed random sweep in between.

The numerical values of the mean takeover times for the five update methods, together with their standard deviations are shown in Table 5.2. Again, the results show that the two random sweep methods are statistically equivalent while the uniform choice and synchronous are not.

Since we use a von Neumann neighborhood, the probabilities $p_1, p_2,$ and $p_3$ of selecting
the best individual when there are respectively 1, 2, and 3 copies of it in the neighborhood are respectively 2/5, 7/10, and 9/10. Using these probabilities in the models, we calculated the theoretical growth curves. Figure 5.8 shows the predicted and the experimental curves for the five update methods. The agreement between theory and experiment can be considered very good.

### 5.7 Rectangular Toroidal Structures

It has been shown in the literature (Alba and Troya, 2000; Dorronsoro et al., 2004; Alba and Dorronsoro, 2005) that varying the ratio of the grid axes in a two-dimensional cEA is another simple way for controlling the global induced selection pressure. In this section we address with our models the prediction of takeover regimes for cEAs whose population shape is toroidal but not square. Since different kinds of rectangular shapes could be used in a toroidal cEA, we here analyze the behavior of the mathematical models in such
scenarios.

Let us suppose a rectangular toroidal structure of size equal to $a \times b$, with $a \geq b$; the same geometrical approximation, done in the case of a square toroidal structure (see Figure 5.2), can be applied to this case. This time the models will describe the growth of a rotated square until its area is equal to $b^2/2$, followed by a composition of $b$ linear growths until the area of the region is $ab - b^2/4$, plus a final quadratic saturation (see Figure 5.9).

The recurrences modeling the synchronous and the three asynchronous evolutions will therefore be composed by the initial condition ($N(0) = 1$), followed by the equation describing the unrestricted growth of the square (until $N(t) = b^2/2$), then the composition

Figure 5.8: Comparison of the experimental takeover time curves (full) with the model (dashed) in the case of linear ranking selection for four update methods: synchronous (a), asynchronous line sweep (b), asynchronous fixed random sweep (c), and uniform choice (d).
of \( b \) linear growths (until \( N(t) = ab - b^2/4 \)), and finally the saturation equation.

For a synchronous evolution the recurrences are:

\[
\begin{align*}
N(0) &= 1 \\
N(t) &= N(t-1) + 4p_2 \frac{\sqrt{N(t-1)}}{\sqrt{2}} \\
N(t) &= N(t-1) + 2(b-1)p_2 + p_1 \\
N(t) &= N(t-1) + 4p_2 \sqrt{n} - N(t-1).
\end{align*}
\]

For an asynchronous Fixed Line Sweep update the model is:

\[
\begin{align*}
N(0) &= 1 \\
N(t) &= N(t-1) + \left( \frac{2p_2 - p_2^2}{1 - p_2} + 2\frac{2p_1 - p_2^2}{1 - p_1} \right) \sqrt{N(t-1)} \\
N(t) &= N(t-1) + \left( \frac{2p_2 - p_2^2}{1 - p_2} \right) (b - 1) + \frac{2p_1 - p_2^2}{1 - p_1} \\
N(t) &= N(t-1) + \left( \frac{2p_2 - p_2^2}{1 - p_2} + 2\frac{2p_1 - p_2^2}{1 - p_1} \right) \sqrt{ab - N(t-1)}.
\end{align*}
\]

For asynchronous Fixed and New Random Sweeps updates the recurrences are:

\[
\begin{align*}
N(0) &= 1 \\
N(t) &= N(t-1) + 4 \left( p_2 + p_2p_1 + \frac{1}{4}(p_2 - 2p_1)p_2^2 \right) \\
N(t) &= N(t-1) + \left( \frac{2p_2 - p_2^2}{1 - p_2} \right) (b - 1) + \frac{4p_1 - p_2^2}{1 - p_1} \\
N(t) &= N(t-1) + 4 \left( p_2 + p_2p_1 + \frac{1}{4}(p_2 - 2p_1)p_2^2 \right) \\
N(t) &= N(t-1) + 8 \left( p_3 + p_3p_1 + \frac{1}{4}(p_3 - 2p_1)p_3^2 \right).
\end{align*}
\]
For asynchronous Uniform Choice evolutions the model is:

\[
\begin{align*}
N(0) &= 1 \\
N(t) &= N(t - 1) + 4p_2\sqrt{N(t - 1)} \\
N(t) &= N(t - 1) + 2\sqrt{2}p_2(b - 1) + p_1 \\
N(t) &= N(t - 1) + 4p_2(\sqrt{ab - N(t - 1)} - 1) + 8p_3.
\end{align*}
\]

As it was the case for the models of the square toroidal structure, also for these recurrences the closed analytical form of the recurrences is practically impossible to be derived.

### 5.7.1 Rectangular Toroidal Models Validation

The cEA structures have now rectangular torus topologies of sizes $64 \times 16$ and $128 \times 8$ with von Neumann neighborhood. On the two topologies the cEA has been run, as in the previous experiments, using the binary tournament and the linear ranking selections. The results (summarized in table 5.3 for the binary tournament and in table 5.4 for the linear ranking) show a similar behavior as the one observed in the ring and torus topologies. In fact, the synchronous updates always correspond to the greater takeover time. The four asynchronous evolutions takeover times are ranked with the fixed line sweep being the faster, the uniform choice being the slowest, and the fixed and new random sweeps (that are statistically indistinguishable) in between.

<table>
<thead>
<tr>
<th></th>
<th>Synchro</th>
<th>LS</th>
<th>FRS</th>
<th>NRS</th>
<th>UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$64 \times 16$</td>
<td>62.15 (2.4)</td>
<td>29.99 (2.3)</td>
<td>38.1 (1.9)</td>
<td>39.37 (2.0)</td>
<td>48.96 (2.9)</td>
</tr>
<tr>
<td>$128 \times 8$</td>
<td>117.07 (3.7)</td>
<td>55.37 (4.2)</td>
<td>70.57 (3.3)</td>
<td>73.26 (3.5)</td>
<td>89.48 (4.3)</td>
</tr>
</tbody>
</table>

Table 5.3: Mean takeover time with standard deviation in parenthesis of the binary tournament selection for the five update methods on the $64 \times 16$ and the $128 \times 8$ rectangular toroidal topologies. Mean values over 100 independent runs.

As expected, the selection pressure induced by the different sizes of the grids reduces as the grid gets thinner (Alba and Troya, 2000; Dorronsoro et al., 2004): Figure 5.10 shows the different growth curves that result in changing the grid axis values from those of a square to a thin rectangle, when using the binary tournament and the linear ranking.
Table 5.4: Mean takeover time with standard deviation in parenthesis of the linear ranking selection for the five update methods on the $64 \times 16$ and the $128 \times 8$ rectangular toroidal topologies. Mean values over 100 independent runs.

<table>
<thead>
<tr>
<th></th>
<th>Synchro</th>
<th>LS</th>
<th>FRS</th>
<th>NRS</th>
<th>UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$64 \times 16$</td>
<td>57.42 (2.1)</td>
<td>24.6 (2.2)</td>
<td>33.89 (2.1)</td>
<td>35.3 (1.9)</td>
<td>45.05 (2.9)</td>
</tr>
<tr>
<td>$128 \times 8$</td>
<td>108.32 (3.1)</td>
<td>45.98 (3.5)</td>
<td>62.69 (2.6)</td>
<td>64.76 (3.0)</td>
<td>80.78 (4.6)</td>
</tr>
</tbody>
</table>

As it can be seen in Figures 5.11, 5.12, 5.13 and 5.14 the models successfully predict the experimental curves for both binary tournament and linear ranking selections.

5.8 Varying the Radius

Sarma and De Jong have shown convincingly that the neighborhood’s size and shape have an important influence on the induced global selection pressure in the grids (Sarma and DeJong, 1996, 1997). For this reason, and to complete our study, in this section we turn to change the basic value for the radius used until now. We therefore use generalized von Neumann neighborhood of radius 2 in both one- and two-dimensional regular lattices. Such a neighborhood is defined as containing all the individuals at distances smaller or equal to the radius, where the Manhattan distance is used on the two-dimensional grid.
The equation for a ring with radius 2 is as follows:

\[
\begin{cases}
N(0) = 1 \\
N(1) = N(0) + 4p_1 \\
N(t) = N(t-1) + 2p_2 + 2p_1,
\end{cases}
\]

where \(p_i\) \((i = 1, 2)\) is the probability that a copy of the best individual is selected when \(i\) copies of it are present in the neighborhood.

On the other hand, if we use the same geometrical approximation described at the beginning of this chapter, the model equations for a square torus with a radius 2 generalized von Neumann neighborhood are:

\[
\begin{cases}
N(0) = 1 \\
N(1) = N(0) + 12p_1 \\
N(t) = N(t-1) + 4p_5 \frac{\sqrt{N(t-1)}}{\sqrt{2}} + 4p_3 \frac{\sqrt{N(t-1)+1}}{\sqrt{2}} + 2p_1 \frac{\sqrt{N(t-1)+2}}{\sqrt{2}}, \\
\text{for } N(t) \leq \frac{n}{2} \\
N(t) = N(t-1) + 4p_5(\sqrt{n-N(t-1)}) + 4p_3(\sqrt{n-N(t-1)-2}) + \\
+2p_1(\sqrt{n-N(t-1)-2}), \\
\text{for } N(t) > \frac{n}{2},
\end{cases}
\]

where \(p_i\) \((i = 1, 3, 5)\) is the probability that a copy of the best individual is selected when \(i\) copies of it are present in the neighborhood.

The two previous models have been tested using a binary tournament selection mechanism. The comparisons between the predicted and the actual curves are shown in Figure 5.15. The models are still accurate, and could be extended to generalized von Neumann neighborhoods with larger radii.

Although this is not apparent from the explicit recurrences, looking at the curves one can infer that when the radius tends to \(n/2\) in the ring case, and when radius tends to \(\sqrt{n}\) in the grid case, then the curves tend to the panmictic limit.
5.9 Summary

We have presented quantitative models describing the growth of a single best individual in cellular evolutionary algorithms structured as a torus with von Neumann neighborhood. New results have been obtained for synchronous and some asynchronous cell update policies. The models are given as probabilistic recurrence equations. We have studied two types of selection mechanisms that are commonly used in cEAs: binary tournament and linear ranking. With these selection methods, our results show that there is a good agreement between theory and experiment; in particular, we confirmed that asynchronous cell update methods give rise to different global selection intensity. This should allow the control of selection pressure in an easy and principled way, without using ad hoc parameters.

Finally, we have mathematically explained the selection pressure present in cEAs when a non-square topology is used in toroidal cEAS, an important factor describing their performance. Just in the same line, different radius values are studied to offer a complete work on this topic.
5.9. SUMMARY

Figure 5.11: Experimental curves for rectangular $64 \times 16$ with binary tournament (a). Comparison of the experimental takeover time curves (full) with the model (dashed) in the case of linear ranking selection for four update methods: synchronous (b), asynchronous line sweep (c), asynchronous fixed random sweep (d), asynchronous new random sweep (e), uniform choice (f).
Figure 5.12: Experimental curves for rectangular $128 \times 8$ with binary tournament (a). Comparison of the experimental takeover time curves (full) with the model (dashed) in the case of linear ranking selection for four update methods: synchronous (b), asynchronous line sweep (c), asynchronous fixed random sweep (d), asynchronous new random sweep (e), uniform choice (f).
Figure 5.13: Experimental curves for rectangular $64 \times 16$ with linear ranking (a). Comparison of the experimental takeover time curves (full) with the model (dashed) in the case of linear ranking selection for four update methods: synchronous (b), asynchronous line sweep (c), asynchronous fixed random sweep (d), asynchronous new random sweep (e), uniform choice (f).
Figure 5.14: Experimental curves for rectangular $128 \times 8$ with linear ranking (a). Comparison of the experimental takeover time curves (full) with the model (dashed) in the case of linear ranking selection for four update methods: synchronous (b), asynchronous line sweep (c), asynchronous fixed random sweep (d), asynchronous new random sweep (e), uniform choice (f).
Figure 5.15: Comparison of the experimental takeover time curves (full) with the model (dashed) in the case of binary tournament selection for synchronous in case of a ring with radius 1 (a) and radius 2 (b) neighborhoods, and in a torus with radius 1 (c) and with radius 2 (d) generalized von Neumann neighborhoods.
Chapter 6

Optimisation by Evolution on Regular Structures

Theory is when nothing works but we know why.
Practice is when everything works and we don’t know why.
In this Institute we have re-united Theory and Practice:
nothing works and we don’t know why!
– Anonymous

As already noted in the first chapter, Artificial Evolution is often used as an optimisation technique. In particular, cEAs have been empirically reported as being useful in maintaining diversity and promoting slow diffusion of solutions through the grid. Such a behavior is mainly due to the lower selection pressure induced on the search process.

In this chapter we will present a set of discrete and continuous problems chosen to study the different optimisation behaviors showed by a cEA with a two-dimensional square structure when varying the grid shape (i.e., rectangular topologies, keeping a synchronous update policy fixed) or the time dimension (i.e., asynchronous update policies keeping the square shape of the structure fixed). The selected benchmark is representative because it contains many different interesting features found in optimization, such as epistasis, multimodality, deceptiveness, use of constraints, parameter identification, and problem generators. These are important ingredients in any work trying to evaluate algorithmic approaches with the objective of getting reliable results, as stated by Whitley et al. (1997).

The chapter is mainly divided in two parts: first the different topologies and update
methods will be tested on a set of discrete optimisation problems (Section 6.1), and then
the same spacial and temporal settings will be compared on a set of continuous optimi-
sation problems (Section 6.2). In each section, the problems will be first presented and
then the performances of the evolutionary algorithms will be analysed. The statistical
comparison of the results of the studied algorithms can be found in the Annexes. Finally,
in the last section we will summarise the result and we will present the future perspectives
opened by this investigation.

6.1 Experiments in Discrete Optimization

The chosen discrete optimisation problems is the set of problems studied in (Alba and
Troya, 2000), which includes the massively multimodal deceptive problem (MMDP), the
frequency modulation sounds (FMS), and the multimodal problem generator P-PEAKS;
we then extend this basic three-problem benchmark with error correcting code design
(ECC), maximum cut of a graph (MAXCUT), the minimum tardy task problem (MTTP),
and the satisfiability problem (SAT).

The problems selected for this benchmark are explained in subsections 6.1.1 to 6.1.7.
The choice of this set of problems is justified by both their difficulty and their applica-
tion domains (parameter identification, telecommunications, combinatorial optimization,
scheduling, etc.). This gives us a high level of confidence in the results, although the eval-
uation of conclusions is consequently more laborious than with a small test suite. Finally,
in subsection 6.1.8 We present and analyze the results.

6.1.1 Massively Multimodal Deceptive Problem

The MMDP is a problem that has been specifically designed to be difficult for an EA
(Goldberg et al., 1992). It is made up of \( k \) deceptive subproblems \( s_i \) of 6 bits each one,
whose value depends on the number of ones \( \text{unitation} \) a binary string has (see Fig. 6.1).
It is easy to see (graphic of Fig. 6.1) that these subfunctions have two global maxima and
a deceptive attractor in the middle point.

In MMDP each subproblem \( s_i \) contributes to the fitness value according to its \text{unitation}
6.1. EXPERIMENTS IN DISCRETE OPTIMIZATION

![Figure 6.1: Basic deceptive bipolar function ($s_i$) for MMDP.](image)

(Fig. 6.1). The global optimum has a value of $k$ and it is attained when every subproblem is composed of zero or six ones. The number of local optima is quite large ($22^k$), while there are only $2^k$ global solutions. Therefore, the degree of multimodality is regulated by the $k$ parameter. We use here a considerably large instance of $k = 20$ subproblems. The instance we try to maximize for solving the problem is shown in the following equation, and its maximum value is equal to $k$.

$$f_{\text{MMDP}}(\vec{s}) = \sum_{i=1}^{k} \text{fitness}_{s_i}.$$  

### 6.1.2 Frequency Modulation Sounds

The FMS problem (Tsutsui and Fujimoto, 1993) is defined as determining the 6 real parameters $\vec{x} = (a_1, w_1, a_2, w_2, a_3, w_3)$ of the frequency modulated sound model given in Eq. 6.1 for approximating it to the sound wave given in Eq. 6.2 (where $\theta = 2 \cdot \pi / 100$). The parameters are defined in the range $[-6.4, +6.35]$, and we encode each parameter into a 32 bit substring in the individual.

$$y(t) = a_1 \cdot \sin(\omega_1 \cdot t \cdot \theta + a_2 \cdot \sin(\omega_2 \cdot t \cdot \theta + a_3 \cdot \sin(\omega_3 \cdot t \cdot \theta))), \quad (6.1)$$

$$y_0(t) = 1.0 \cdot \sin(5.0 \cdot t \cdot \theta - 1.5 \cdot \sin(4.8 \cdot t \cdot \theta + 2.0 \cdot \sin(4.9 \cdot t \cdot \theta))). \quad (6.2)$$

The goal is to minimize the sum of square errors given by Eq. 6.3. This problem is a highly complex multimodal function having strong epistasis, with optimum value 0.0.
Due to the high difficulty of solving this problem with high accuracy without applying local search or specific operators for continuous optimization (like gradual GAs (Herrera and Lozano, 2000)), we stop the algorithm when the error falls below $10^{-2}$. Hence, the objective for this problem will be to minimize the following equation:

$$f_{FMS}(\vec{x}) = \sum_{t=0}^{100} \left( y(t) - y_0(t) \right)^2.$$  \hfill (6.3)

### 6.1.3 Multimodal Problem Generator P-PEAKS

The P-PEAKS problem (DeJong et al., 1997) is a multimodal problem generator. A problem generator is an easily parameterizable task which has a tunable degree of epistasis, thus admitting to derive instances with growing difficulty at will. Also, using a problem generator removes the opportunity to hand-tune algorithms to a particular problem, therefore allowing a larger fairness when comparing algorithms. With a problem generator we evaluate our algorithms on a high number of random problem instances, since a different instance is solved each time the algorithm runs, then the predictive power of the results for the problem class as a whole is increased.

The idea of P-PEAKS is to generate $P$ random $N$-bit strings that represent the location of $P$ peaks in the search space. The fitness value of a string is the number of bits the string has in common with the nearest peak in that space, divided by $N$:

$$f_{P-PEAKS}(\vec{x}) = \frac{1}{N} \max_{1 \leq i \leq P} \left\{ N - \text{HammingD}(\vec{x}, \text{Peak}_i) \right\}.$$  

By using a small/large number of peaks we can get weakly/strongly epistatic problems. In this paper we have used an instance of $P = 100$ peaks of length $N = 100$ bits each, which represents a medium/high epistasis level (Alba and Troya, 2000). The maximum fitness value for this problem is 1.0.

### 6.1.4 Error Correcting Code Design Problem

The ECC problem was presented in MacWilliams and Sloane (1977). We will consider a three-tuple $(n, M, d)$, where $n$ is the length of each codeword (number of bits), $M$ is
6.1. EXPERIMENTS IN DISCRETE OPTIMIZATION

the number of codewords, and \( d \) is the minimum Hamming distance between any pair of codewords. The objective will be to find a code which has a value for \( d \) as large as possible (reflecting greater tolerance to noise and errors), given previously fixed values for \( n \) and \( M \). The problem we have studied is a simplified version of that in (MacWilliams and Sloane, 1977). In this case the algorithm searches half of the codewords \((M/2)\) that will compose the code, and the other half is made up of the complement of the codewords computed by the algorithm.

The fitness function to be maximized is:

\[
f_{\text{ECC}} = \frac{1}{M} \sum_{i=1}^{M} \sum_{j=1, i \neq j}^{M} d_{ij}^{-2},
\]

where \( d_{ij} \) represents the Hamming distance between codewords \( i \) and \( j \) in the code \( C \) (made up of \( M \) codewords, each of length \( n \)). We consider an instance where \( M = 24 \) and \( n = 12 \). The search space is of size \( \left( \begin{array}{c} 4096 \\ 24 \end{array} \right) \), which is approximately \( 10^{87} \). The optimum solution for \( M = 24 \) and \( n = 12 \) has a fitness value of 0.0674 (Chen et al., 1998).

6.1.5 Maximum Cut of a Graph

The MAXCUT problem looks for a partition of the set of vertices \( V \) of a weighted graph \( G = (V, E) \) into two disjoint subsets \( V_0 \) and \( V_1 \) so that the sum of the weights of the edges with one endpoint in \( V_0 \) and the other one in \( V_1 \) is maximized. For encoding the problem we use a binary string \( (x_1, x_2, \ldots, x_n) \) of length \( n \) where each digit corresponds to a vertex. If a digit is 1 then the corresponding vertex is in set \( V_1 \); if it is 0 then the corresponding vertex is in set \( V_0 \). The function to be maximized (Khuri et al., 1994) is:

\[
f_{\text{MAXCUT}}(\vec{x}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} \cdot [x_i \cdot (1 - x_j) + x_j \cdot (1 - x_i)].
\]

Note that \( w_{ij} \) contributes to the sum only if nodes \( i \) and \( j \) are in different partitions. While one can generate different random graph instances to test the algorithm, here we
have used the case “cut20.09”, with 20 vertices and a probability of 0.9 of having an edge between any two randomly chosen vertices. The maximum fitness value for this instance is 56.740064.

6.1.6 Minimum Tardy Task Problem

MTTP (Stinson, 1985 (second edition, 1987)) is a task-scheduling problem wherein each task $i$ from the set of tasks $T = \{1, 2, \ldots, n\}$ has a length $l_i$ (the time it takes for its execution), a deadline $d_i$ (before which a task must be scheduled, and its execution completed), and a weight $w_i$. The weight is a penalty that has to be added to the objective function in the event that the task remains unscheduled. The lengths, weights and deadlines of tasks are all positive integers. Scheduling the tasks of a subset $S$ of $T$ is to find the starting time of each task in $S$, such as at most one task at time is performed and such that each task finishes before its deadline.

We characterize a one-to-one scheduling function $g$ defined on a subset of tasks $S \subseteq T : S \mapsto \mathbb{Z}^+ \cup \{0\}$, so that for all tasks $i, j \in S$ has the following properties:

1. A task can not be scheduled before any previous one has finished: $g(i) < g(j) \Rightarrow g(i) + l_i \leq g(j)$.

2. Every task finishes before its deadline: $g(i) + l_i \leq d_i$.

The objective function for this problem is to minimize the sum of the weights of the unscheduled tasks. Therefore, the optimum scheduling minimizes the following equation:

$$f_{\text{MTTP}}(\vec{x}) = \sum_{i \in T - S} w_i$$

The schedule of tasks $S$ can be represented by a vector $\vec{x} = (x_1, x_2, \ldots, x_n)$ containing all the tasks ordered by its deadline. Each $x_i \in \{0, 1\}$, where if $x_i = 1$ then task $i$ is scheduled in $S$, while if $x_i = 0$ means that task $i$ is not included in $S$. The fitness function is the inverse of the previous equation, as described in Khuri et al. (1994). In this study we have used an instance called “mttp20”, with size 20, and maximum fitness value of 0.02439.
6.1.7 Satisfiability Problem

The satisfiability problem (SAT) has received much attention from the scientific community since it plays a central role in NP-completeness (Garey and Johnson, 1979). The SAT problem was the first which was demonstrated to belong to the NP class of problems.

SAT consists in assigning values to a set of \( n \) boolean variables \( x = (x_1, x_2, \ldots, x_n) \) such that they satisfy a given set of clauses \( c_1(\vec{x}), \ldots, c_m(\vec{x}) \), where \( c_i(\vec{x}) \) is a disjunction of literals, and a literal is a variable or its negation. Hence, we can define SAT as a function \( f : \mathbb{B}^n \to \mathbb{B}, \mathbb{B} = \{0, 1\} \) like:

\[
f_{\text{SAT}}(\vec{x}) = c_1(\vec{x}) \land c_2(\vec{x}) \land \ldots \land c_m(\vec{x}).
\]

An instance of SAT, \( \vec{x} \), is called satisfiable if \( f_{\text{SAT}}(\vec{x}) = 1 \), and unsatisfiable otherwise. A \( k \)-SAT instance consists of clauses with length \( k \). When \( k \geq 3 \) the problem is NP-hard (Garey and Johnson, 1979). In this chapter we will consider an instance of 3-SAT made up of 430 clauses and 100 variables. This instance belongs to the well-known phase transition of hard SAT instances. The fitness function is a linear function of the number of satisfied clauses. In it, we use the so called stepwise adaptation of weights (SAW) (Eiben and van der Hauw, 1997):

\[
f_{\text{SAW}}(\vec{x}) = w_1 \cdot c_1(\vec{x}) + \ldots + w_m \cdot c_m(\vec{x}).
\]

This function weights the values of the clauses with \( w_i \in \mathbb{B} \) in order to give more importance to those clauses which are not satisfied yet by the current best solution. These weights are adjusted dynamically according to \( w_i = w_i + 1 - c_i(\vec{x}^*) \), being \( \vec{x}^* \) the current fittest individual.

6.1.8 Experimental Analysis in Discrete Optimization

Although a full-length study of the problems presented in the previous section is beyond the scope of this work, we present results comparing synchronous and asynchronous square torus cEAs, and also synchronous cEAs having different rectangular torus topologies, al-
ways with von Neumann neighborhood. Note that it is not the aim here to compare cEAs performance with state-of-the-art algorithms and heuristics for combinatorial and numerical optimization. To this end, we should at least tune the parameters and include local search capabilities in the algorithm, which is not the case. Thus, the results only pertain to the relative performance of the different cEA update methods and shapes among themselves. The configuration of the algorithm for the binary encoded problems is shown in Table 6.1, and the three different shapes used are in Table 6.2.

<table>
<thead>
<tr>
<th>Population Size</th>
<th>400 individuals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection of Parents</td>
<td>binary tournament + binary tournament</td>
</tr>
<tr>
<td>Recombination</td>
<td>DPX, $p_c = 1.0$</td>
</tr>
<tr>
<td>Bit Mutation</td>
<td>Bit-flip, $p_m = 1/L$ (10/L for FMS)</td>
</tr>
<tr>
<td>Individual Length</td>
<td>L</td>
</tr>
<tr>
<td>Replacement</td>
<td>Rep_if_Better</td>
</tr>
</tbody>
</table>

Table 6.1: Parameterization used in the algorithm for the binary encoded problems. DPX indicates standard double point crossover, and Rep_if_Better indicates the replacement operator that replaces the considered individual by the offspring if this last has a better fitness.

<table>
<thead>
<tr>
<th>Name</th>
<th>(shape of population)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>(20 × 20 individuals)</td>
</tr>
<tr>
<td>Rectangular</td>
<td>(10 × 40 individuals)</td>
</tr>
<tr>
<td>Narrow</td>
<td>(4 × 100 individuals)</td>
</tr>
</tbody>
</table>

Table 6.2: Torus shapes used in the experiments.

We show in the following tables the results for the problems mentioned before: MMDP (Table 6.3), FMS (Table 6.4), P-PEAKS (Table 6.5), ECC (Table 6.6), MAXCUT (Table 6.7), MTTP (Table 6.8), and SAT (Table 6.9). In these tables we report the average of the final best fitness of the algorithm, the average number of evaluations to obtain the optimum value (if obtained), and the hit rate (percentage of successful runs). Therefore, we are analyzing the final distance to the optimum (especially interesting when the optimum is not found), the effort of the algorithm, and the expected efficacy of the algorithm, respectively. In order to get reliable results, we have performed 100 independent runs for any algorithm and for every problem in the test-suite.
### 6.1. EXPERIMENTS IN DISCRETE OPTIMIZATION

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Avg. Solution</th>
<th>Avg. Generations</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>19.813</td>
<td>214.2</td>
<td>57%</td>
</tr>
<tr>
<td>Rectangular</td>
<td>19.824</td>
<td>236.1</td>
<td>58%</td>
</tr>
<tr>
<td>Narrow</td>
<td>19.842</td>
<td>299.7</td>
<td>61%</td>
</tr>
<tr>
<td>LS</td>
<td>19.518</td>
<td>343.5</td>
<td>23%</td>
</tr>
<tr>
<td>FRS</td>
<td>19.601</td>
<td>209.9</td>
<td>31%</td>
</tr>
<tr>
<td>NRS</td>
<td>19.536</td>
<td>152.9</td>
<td>28%</td>
</tr>
<tr>
<td>UC</td>
<td>19.615</td>
<td>295.7</td>
<td>36%</td>
</tr>
</tbody>
</table>

Table 6.3: MMDP problem with a maximum of 1000 generations. A run is considered successful when a fitness of 20 is found.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Avg. Solution</th>
<th>Avg. Generations</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>90.46</td>
<td>437.4</td>
<td>57%</td>
</tr>
<tr>
<td>Rectangular</td>
<td>85.78</td>
<td>404.3</td>
<td>61%</td>
</tr>
<tr>
<td>Narrow</td>
<td>80.76</td>
<td>610.9</td>
<td>63%</td>
</tr>
<tr>
<td>LS</td>
<td>81.44</td>
<td>353.4</td>
<td>58%</td>
</tr>
<tr>
<td>FRS</td>
<td>73.11</td>
<td>386.2</td>
<td>55%</td>
</tr>
<tr>
<td>NRS</td>
<td>76.21</td>
<td>401.5</td>
<td>56%</td>
</tr>
<tr>
<td>UC</td>
<td>83.56</td>
<td>405.2</td>
<td>57%</td>
</tr>
</tbody>
</table>

Table 6.4: FMS problem with a maximum of 3000 generations. A run is considered successful when a fitness greater or equal 100 is found.

From the inspection of these tables some conclusions can be clearly drawn. First, the studied asynchronous algorithms tend to need a smaller number of generations than the synchronous ones to locate an optimum, in general. Moreover, referring to tables in the Appendix where \( t \)-tests are reported (character ‘+’ stands for significant values, while ‘-’ means no significance), the reader will confirm that the differences among asynchronous and synchronous algorithms are statistically significant (with two exceptions), thus indicating that the asynchronous versions perform more efficiently with respect to cEAs with a changing ratio. However, there are punctual exceptions, like in MMDP and SAT, which are probably the hardest discrete problems of the benchmark.

Conversely, synchronous algorithms perform like asynchronous or even better in terms of the percentage of solutions found (hit rate), while the quality of solutions found by the algorithms does not always have significant differences (the exceptions are probably due to the difference on the hit rate). Conversely, if we pay attention to the success (hit) rate, it can be concluded that the synchronous policies with rectangular shapes outperform the
Table 6.5: P-PEAKS problem with a maximum of 100 generations. A run is considered successful when a fitness of 1 is found.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Avg. Solution</th>
<th>Avg. Generations</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>1.0</td>
<td>51.8</td>
<td>100%</td>
</tr>
<tr>
<td>Rectangular</td>
<td>1.0</td>
<td>50.4</td>
<td>100%</td>
</tr>
<tr>
<td>Narrow</td>
<td>1.0</td>
<td>53.9</td>
<td>100%</td>
</tr>
<tr>
<td>LS</td>
<td>1.0</td>
<td>34.8</td>
<td>100%</td>
</tr>
<tr>
<td>FRS</td>
<td>1.0</td>
<td>38.4</td>
<td>100%</td>
</tr>
<tr>
<td>NRS</td>
<td>1.0</td>
<td>38.8</td>
<td>100%</td>
</tr>
<tr>
<td>UC</td>
<td>1.0</td>
<td>40.1</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 6.6: ECC problem with a maximum of 500 generations. A run is considered successful when a fitness of 0.0674 is found.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Avg. Solution</th>
<th>Avg. Generations</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>0.0670</td>
<td>93.9</td>
<td>85%</td>
</tr>
<tr>
<td>Rectangular</td>
<td>0.0671</td>
<td>93.4</td>
<td>88%</td>
</tr>
<tr>
<td>Narrow</td>
<td>0.0673</td>
<td>104.2</td>
<td>94%</td>
</tr>
<tr>
<td>LS</td>
<td>0.0672</td>
<td>79.7</td>
<td>89%</td>
</tr>
<tr>
<td>FRS</td>
<td>0.0672</td>
<td>82.4</td>
<td>90%</td>
</tr>
<tr>
<td>NRS</td>
<td>0.0672</td>
<td>79.5</td>
<td>89%</td>
</tr>
<tr>
<td>UC</td>
<td>0.0671</td>
<td>87.3</td>
<td>86%</td>
</tr>
</tbody>
</table>

asynchronous algorithms: slightly in terms of the average final fitness, and clearly in terms of probability of finding a solution (i.e., frequency of optimum location).

Another interesting result is that we can define two classes of problems: those solved by any method to optimality (100% hit rate) and those in which no 100% rate is achieved at all. Problems seem to be amenable for cEAs directly, or to need some (yet unstudied) help, e.g., by including local search.

In order to summarize the large set of results and get some useful conclusion we present a ranking with the best algorithms by following three different metrics: average best final solution, average number of generations on success, and hit rate. Table 6.10 shows the three mentioned rankings. These rankings have been computed by adding the position (from better to worse: 1, 2, 3 ...) that algorithms are allocated for the previous results presented from Table 6.3 to Table 6.9, according to the three criteria.

As we would expect after the previous comments, according to the average final best fitness and hit rate criteria, synchronous algorithms with any of the three studied shapes
are in general more accurate than all the asynchronous ones for our test problems, with a special leading position for narrow population grids. On the other hand, asynchronous versions clearly outperform any of the synchronous algorithms in terms of the average number of generations (efficiency), with a trend towards LS as being the best ranked flavor of cEA for our discrete test suite.

### 6.2 Experiments in Continuous Optimization

We will extend in this section the work of the previous one by testing all the algorithms with some continuous functions in order to get a more extensive study. This study may be interesting for analyzing the behavior of the algorithms in continuous optimization in contrast to the performed study on discrete optimization. The functions we have selected for the study are three typical multimodal numerical benchmarks: Rastrigin’s (RASTR) and Ackley’s (ACKL) functions, and a fractal (FRAC) function. These three problems are

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Avg. Solution</th>
<th>Avg. Generations</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>56.74</td>
<td>11.3</td>
<td>100%</td>
</tr>
<tr>
<td>Rectangular</td>
<td>56.74</td>
<td>11.0</td>
<td>100%</td>
</tr>
<tr>
<td>Narrow</td>
<td>56.74</td>
<td>11.9</td>
<td>100%</td>
</tr>
<tr>
<td>LS</td>
<td>56.74</td>
<td>9.5</td>
<td>100%</td>
</tr>
<tr>
<td>FRS</td>
<td>56.74</td>
<td>9.7</td>
<td>100%</td>
</tr>
<tr>
<td>NRS</td>
<td>56.74</td>
<td>9.6</td>
<td>100%</td>
</tr>
<tr>
<td>UC</td>
<td>56.74</td>
<td>9.6</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 6.7: MAXCUT problem with a maximum of 100 generations. A run is considered successful when a fitness of 56.74 is found.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Avg. Solution</th>
<th>Avg. Generations</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>0.02439</td>
<td>8.4</td>
<td>100%</td>
</tr>
<tr>
<td>Rectangular</td>
<td>0.02439</td>
<td>8.3</td>
<td>100%</td>
</tr>
<tr>
<td>Narrow</td>
<td>0.02439</td>
<td>8.9</td>
<td>100%</td>
</tr>
<tr>
<td>LS</td>
<td>0.02439</td>
<td>5.9</td>
<td>100%</td>
</tr>
<tr>
<td>FRS</td>
<td>0.02439</td>
<td>6.2</td>
<td>100%</td>
</tr>
<tr>
<td>NRS</td>
<td>0.02439</td>
<td>6.3</td>
<td>100%</td>
</tr>
<tr>
<td>UC</td>
<td>0.02439</td>
<td>6.3</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 6.8: MTTP problem with a maximum of 50 generations. A run is considered successful when a fitness of 0.02439 is found.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Avg. Solution</th>
<th>Avg. Generations</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>429.54</td>
<td>703.1</td>
<td>79%</td>
</tr>
<tr>
<td>Rectangular</td>
<td>429.67</td>
<td>706.3</td>
<td>84%</td>
</tr>
<tr>
<td>Narrow</td>
<td>429.61</td>
<td>763.7</td>
<td>81%</td>
</tr>
<tr>
<td>LS</td>
<td>429.52</td>
<td>463.2</td>
<td>78%</td>
</tr>
<tr>
<td>FRS</td>
<td>429.67</td>
<td>497.7</td>
<td>85%</td>
</tr>
<tr>
<td>NRS</td>
<td>429.49</td>
<td>610.5</td>
<td>75%</td>
</tr>
<tr>
<td>UC</td>
<td>429.50</td>
<td>725.5</td>
<td>76%</td>
</tr>
</tbody>
</table>

Table 6.9: SAT problem with a maximum of 3000 generations. A run is considered successful when a fitness of 430 is found.

<table>
<thead>
<tr>
<th>Avg. Solution</th>
<th>Avg. Generations</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Narrow</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>1 Rectangular</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>3 Square</td>
<td>14</td>
<td>3</td>
</tr>
<tr>
<td>4 FRS</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>5 LS</td>
<td>18</td>
<td>5</td>
</tr>
<tr>
<td>5 UC</td>
<td>18</td>
<td>6</td>
</tr>
<tr>
<td>7 NRS</td>
<td>21</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 6.10: Ranking of the algorithms with discrete problems.

real-coded in the algorithms, while we previously used standard binary-coded individuals (Section 6.1). That is the cause for our special interest on experimenting with a more traditional global optimization. The codification we have employed for these three problems has been made by following Michalewicz’s implementation (Michalewicz, 1996).

6.2.1 Rastrigin’s Function

The Generalized Rastrigin function is a sinusoidally modulated function with a global minimum of zero at the origin. It is a typical example of non-linear multimodal function. It was first proposed by Rastrigin as a 2-dimensional function and has been generalized by Mühlenbein et al. in (Mühlenbein and Schlierkamp-Voosen, 1993). This function is a fairly difficult problem due to its large search space and its large number of local minima, although the function is separable and the local minima are symmetrical:

\[ f_{RASTR}(\mathbf{x}) = nA + \sum_{i=1}^{n} x_i^2 - A \cos(\omega x_i). \]
The constants are given by $A = 10$, and $\omega = 2\pi$. The domain of variables $x_i$, $i = 1, \ldots, n$, is $-5.12 \leq x_i \leq 5.12$, and $n = 10$. The function has a global minimum at the point $f(0) = 0$.

6.2.2 Ackley’s Function

Ackley’s function is a multimodal test function obtained by cosine modulation of an exponential function. Originally proposed by Ackley (Ackley, 1987) as a two-dimensional function, it has been generalized by Bäck et al. (Bäck et al., 1993):

$$f_{\text{ACKL}}(\mathbf{x}) = -a \exp \left[ -b \left( \frac{1}{n} \sum_{i=1}^{n} x_i^2 \right)^{1/2} \right] - \exp \left( \frac{1}{n} \sum_{i=1}^{n} \cos(cx_i) \right) + a + e.$$  

Contrary to Rastrigin’s, Ackley’s function is not separable, even though it also shows a regular arrangement of the local optima. The constants are $a = 20$, $b = 0.2$, $c = 2\pi$. The variables $x_i$, $i = 1, \ldots, n$ are in the domain $-32.768 \leq x_i \leq 32.768$. This function has a global minimum at the point $f(0) = 0$.

6.2.3 Fractal Function (FRAC)

This function has been taken from Bäck (1996), were its construction, as well as motivations for introducing it as a test problem, are given. Indeed, the function allows the degree of ruggedness to be controlled, and it is likely to capture features of real-world noisy objective functions:

$$f_{\text{FRAC}}(\mathbf{x}) = \sum_{i=1}^{n} (C'(x_i) + x_i^2 - 1),$$

where

$$C'(z) = \begin{cases} 
\frac{C(z)}{C'(1)|z|^{2-D}} & \text{if } z \neq 0 \\
1 & \text{if } z = 0
\end{cases}.$$
and

\[ C(z) = \sum_{j=-\infty}^{\infty} \frac{1 - \cos(b^j z)}{b^{(2-D)j}}. \]

For the runs we have chosen the constants \( D = 1.85 \) and \( b = 1.5 \). The 20 variables \( x_i \) \((x_i = 1 \ldots 20)\) vary in the range \([-5, 5]\). The infinite sum in the function \( C(z) \) is practically calculated starting with \( j = 0 \) and alternating the signs of the \( j \) values. The sum is stopped when the relative difference between its previous value and the present one is lower than \( 10^{-8} \), or when \( j = 100 \) is reached.

### 6.2.4 Experimental Analysis in Continuous Optimization

In this section we will study the results of our experiments with the proposed continuous problems, as we did in Section 6.1.8 for the discrete case. We maintain, in this case, the shapes used for the synchronous algorithms with respect to those studied in Section 6.1, as well as the asynchronous update criteria. On the other hand, we have needed a specific configuration for the genetic operators and theirs probabilities in the case of real encoded problems. This codification is detailed in Table 6.11.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population Size</td>
<td>400 individuals</td>
</tr>
<tr>
<td>Selection of Parents</td>
<td>binary tournament + binary tournament</td>
</tr>
<tr>
<td>Recombination</td>
<td>AX, ( p_c = 1.0 )</td>
</tr>
<tr>
<td>Bit Mutation</td>
<td>Uniform, ( p_m = 1/2L )</td>
</tr>
<tr>
<td>Individual Length</td>
<td>( L )</td>
</tr>
<tr>
<td>Replacement</td>
<td>Rep_if_Better</td>
</tr>
</tbody>
</table>

Table 6.11: Parameterization used in the algorithm for the real encoded problems. AX stands for standard arithmetic crossover, and Rep_if_Better indicates the replacement operator that replaces the considered individual by the offspring if this last has a better fitness.

We present in the following tables the results of our experiments with RASTR (Table 6.12), ACKL (Table 6.13), and FRAC (Table 6.14) problems. Like in the case of discrete problems, these tables contain values for the average of the final best fitness, the average generations needed for finding it, and the hit rate. These three values are calculated over 100 independent runs. For the three real-coded problems a run is stopped successfully as soon as an individual is found with fitness within 0.1 from the optimum.
Table 6.12: RASTR problem with a maximum of 700 generations. A run is considered successful when a fitness within 0.1 from the optimum is found.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Avg. Solution</th>
<th>Avg. Generations</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>0.0900</td>
<td>323.8</td>
<td>100%</td>
</tr>
<tr>
<td>Rectangular</td>
<td>0.0883</td>
<td>309.8</td>
<td>100%</td>
</tr>
<tr>
<td>Narrow</td>
<td>0.0855</td>
<td>354.2</td>
<td>100%</td>
</tr>
<tr>
<td>LS</td>
<td>0.0899</td>
<td>280.9</td>
<td>100%</td>
</tr>
<tr>
<td>FRS</td>
<td>0.0900</td>
<td>289.6</td>
<td>100%</td>
</tr>
<tr>
<td>NRS</td>
<td>0.0906</td>
<td>292.2</td>
<td>100%</td>
</tr>
<tr>
<td>UC</td>
<td>0.0892</td>
<td>292.4</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 6.13: ACKL problem with a maximum of 500 generations. A run is considered successful when a fitness within 0.1 from the optimum is found.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Avg. Solution</th>
<th>Avg. Generations</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>0.0999</td>
<td>321.7</td>
<td>78%</td>
</tr>
<tr>
<td>Rectangular</td>
<td>0.0994</td>
<td>293.1</td>
<td>73%</td>
</tr>
<tr>
<td>Narrow</td>
<td>0.1037</td>
<td>271.9</td>
<td>65%</td>
</tr>
<tr>
<td>LS</td>
<td>0.0932</td>
<td>302.0</td>
<td>84%</td>
</tr>
<tr>
<td>FRS</td>
<td>0.0935</td>
<td>350.6</td>
<td>92%</td>
</tr>
<tr>
<td>NRS</td>
<td>0.0956</td>
<td>335.5</td>
<td>87%</td>
</tr>
<tr>
<td>UC</td>
<td>0.0968</td>
<td>335.0</td>
<td>85%</td>
</tr>
</tbody>
</table>

The results we have obtained with continuous problems are not as clear as in the discrete case. Regarding to the average number of generations needed to find an optimal solution, asynchronous algorithms does not always perform a lesser number of generations with respect to synchronous ones; examples are ACKL and FRAC problems. Differences among synchronous and asynchronous algorithms are usually significant. This result tells us about a larger efficiency of changing shapes cGAs, which contrast with the findings for discrete problems. On the other hand, contrary to that observed in the case of discrete problems, the success rates of asynchronous algorithms are greater than those of synchronous ones in general. Contrary to the results of Section 6.1.8, where either all the algorithms get a 100% hit rate or none finds the solution in every run for any problem, the FRS cEA is the unique algorithm which is able to find the solution in all the executions for the FRAC problem.

In order to summarize these results, and following the organization of Section 6.1.8, we present in Table 6.15 a ranking with the best algorithms in all the problems by means of
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Avg. Solution</th>
<th>Avg. Generations</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>0.0224</td>
<td>75.2</td>
<td>94%</td>
</tr>
<tr>
<td>Rectangular</td>
<td>0.0359</td>
<td>62.8</td>
<td>78%</td>
</tr>
<tr>
<td>Narrow</td>
<td>0.1648</td>
<td>14.6</td>
<td>16%</td>
</tr>
<tr>
<td>LS</td>
<td>0.0168</td>
<td>69.7</td>
<td>98%</td>
</tr>
<tr>
<td>FRS</td>
<td>0.0151</td>
<td>71.5</td>
<td>100%</td>
</tr>
<tr>
<td>NRS</td>
<td>0.0163</td>
<td>73.6</td>
<td>98%</td>
</tr>
<tr>
<td>UC</td>
<td>0.0138</td>
<td>72.8</td>
<td>96%</td>
</tr>
</tbody>
</table>

Table 6.14: FRAC problem with a maximum of 100 generations. A run is considered successful when a fitness within 0.1 from the optimum is found.

Table 6.15: Ranking of the algorithms with continuous problems.

6.3 Summary

The main conclusion of this chapter is that cEAs can be easily induced to promote exploration or exploitation by simply changing the update policy or the shape of the population. This opens new research lines to decide efficient ways of shifting from a given policy/ratio to another one, in order for the optimum to be reached with a smaller effort when compared to the basic cEA or other types of EAs.

In a later part of this chapter we have applied our extended cEAs to a set of both discrete and continuous test problems. Although our goal has not been that of obtaining
solvers able to compete with state-of-the-art specialized heuristics, the results clearly show that cEAs are very efficient optimization techniques, that could be further improved by being hybridized with local search techniques. The results on the test problems largely confirm, with some small exceptions, that the solving abilities using the various update/shapes are directly linked to their induced selection pressures, showing that exploitation plays an important role. It is clear that the role of exploration might be more important on even harder problem instances, but this aspect can be addressed in our algorithms by using more explorative settings, as well as by using different cEA strategies at different times during the search dynamically.

Our results are clear: with respect to discrete problems, asynchronous algorithms are more efficient than synchronous ones; with statistically significant values for most problems. On the other hand, if we pay attention to the success (hit) rate, it can be concluded that the synchronous policies for the different evaluated ratios outperform the asynchronous algorithms: slightly in terms of the average final fitness, and clearly in terms of probability of finding a solution (i.e., frequency of optimum location).

On the contrary, if we pay attention to the experiments on continuous problems we can get different (somewhat complementary) conclusions. Asynchronous algorithms outperform synchronous ones in the cases of average solutions found and hit rate (significant differences in many cases), while in the average number of generations synchronous algorithms are in general more efficient than asynchronous ones.
Chapter 7

Evolution on Irregular Structures

This is the great paradox of mathematics. It doesn’t matter with how much determination their researchers ignore the world: they always end up with better instruments to understand it.
– John Tierney

In this chapter we study irregular topologies for the cellular model, in order to understand the behavior of the global selection intensity with respect to the known bounding panmictic and regular structure cases. The most general irregular topology is the random graph, which is why we concentrate on this kind of structure in the first part of this chapter. While the random graph is a well-studied structure in mathematics, biology, and the social sciences, it has not, to my knowledge, been investigated in cEAs. For example, models of infection transmission over a population of individuals with random links among them have been known for years. Likewise, information transmission in society has been sometimes modeled using a random graph structure (Newman, 2003).

Recently, families of graphs, collectively called small world that are neither regular nor completely random have attracted a lot of interest (Newman, 2003) because of their striking topological and dynamical properties. In the second part of this chapter we study selection intensity on populations structured according to small-world topologies for the first time, as far as we know.

In the next section, the general form of the recurrences that will be used in this chapter’s models will be introduced. Then, in Section 9.3, the random graph structure
will be defined, and we will present the models for the selection pressure curve for the synchronous and the asynchronous new random sweep and uniform choice updates. The models will be validated in Section 7.3 when using a particular selection operator, the uniform selection. In Section 7.4 we will define two classes of small-world structures, the Watts-Strogats model (Section 7.4.1) and the Barabási-Alabert model (Section 7.4.2). The experimental selection pressure curves for these two families of topologies will be presented in Section 7.5. Finally, in the last section we will summarise the result and we will present the future perspectives opened by this investigation.

7.1 The General Model

In the following sections we give the recurrences describing the growth of the random variable $N(t)$ in cEA evolving populations structured on graphs for the synchronous and the two asynchronous updates New Random Sweep and Uniform Choice. In general, such recurrences take the common form

$$E[N(t)] = \sum_{i=1}^{n} P[N(t-1) = i] (i + \Delta_i(t-1)),$$

(7.1)

where $\Delta_i(t-1) = N(t) - N(t-1)$, given $N(t-1) = i$, is a random variable as well. This random variable will depend on the update method. Let’s suppose that at time step $t-1$ there are $i$ copies of the best individuals. At the next time step each of the other $n-i$ individuals will contain a copy of the best with a probability depending on its number of neighbors, the number of its neighbors containing a copy of the best and the selection operator. The first two conditions can be seen as random variables, both depending on the topology of the population.

Therefore, in a synchronous cEA, in Equation 7.1 we have

$$\Delta_i(t-1) = \sum_{r=1}^{n-i} \sum_{j=1}^{n-1} P[K = j] \sum_{l=0}^{j} P[B_j = l] p_{sel}(j, l),$$

(7.2)

where $K$ denotes the number of neighbors of an individual, $B_j$ the number of copies of
the best individual in a neighborhood of \( j \) individuals, and \( p_{sel}(j, l) \) is the probability of selecting a copy of the best individual among the \( l \) best of \( j \) neighbors.

In an asynchronous New Random Sweep cEA,

\[
\Delta_i(t) = \sum_{r=1}^{n-i} \sum_{j=1}^{n-1} P[K = j] \sum_{l=0}^{j} P[B_{jr} = l] p_{sel}(j, l),
\]

where \( P[B_{jr} = l] \) is such that

\[
P[B_{j1} = l] = P[B_j = l] = \begin{cases} 
1 & \text{if } l = L, \\
0 & \text{otherwise}; 
\end{cases}
\]

\[
P[B_{jr} = l] = \begin{cases} 
1 & \text{if } l = L + \sum_{s=1}^{r-1} P[K = j] \sum_{m=1}^{j} P[B_{js} = m] p_{sel}(j, m), \\
0 & \text{otherwise}.
\end{cases}
\]

with the value \( L \) determined by the structure given to the population.

To model the growth of the number of copies of a best individual in an asynchronous Uniform Choice cEA it is preferable to use cell update steps \( u \) instead of time steps \( t \) in Equation 7.1, which otherwise retains its general form. Thus, the growth of \( N(u) \) is determined by

\[
\Delta_i(u) = \frac{n-i}{n} \sum_{j=1}^{n-1} P[K = j] \sum_{l=0}^{j} P[B_{jr} = l] p_{sel}(j, l),
\]

where the symbols have the same meaning as in Equation 7.2.

### 7.2 The Random Graph Structure

A random graph with \( n \) vertices can be constructed by taking all possible pairs of vertices and connecting each pair with probability \( q \) (and thus not connecting it with probability \( 1 - q \)) (Newman, 2003). In the general case of a cEA evolving a population structured as a random graph with probability \( 0 < q < 1 \) of having an edge between any pair of vertices, the random variables \( K \) and \( B_j \) of equation 7.2 have the following probability functions:

\[
P[K = j] = q^j (1 - q)^{n-1-j};
\]
CHAPTER 7. EVOLUTION ON IRREGULAR STRUCTURES

\[ P[B_j = l] = \begin{cases} 
1 & \text{if } l = \frac{ij}{n-1}, \\
0 & \text{otherwise}, 
\end{cases} \]

since any of the \( j \) neighbors of an individual has a probability \( i/(n-1) \) of containing a copy of the best individual.

Cellular evolutionary algorithms are good candidates for using selection methods that are easily extensible to small local pools such as ranking and tournament. The equations for individual growth in the case of those customary local selection policies are mathematically rather complicated, involving higher moments of the distribution. We therefore use a simplified selection policy, called Uniform Selection, which gives rise to a useful and interesting model.

This selection mechanism randomly selects an individual in the selection pool (i.e. the neighborhood of a given individual). The selected individual then replaces the considered individual if it has a better fitness. Such operator is similar to the local parent selection introduced by Gorges-Schleuter (1999), except that a \((\mu + \lambda/\mu, \nu)\)-LES is used instead of a \((\mu, \lambda/\mu, \nu)\)-LES, with \( \kappa = \infty \) (\( \kappa \) being the upper limit for the life span) and \( \rho = 1 \) (\( \rho \) being the number of selected ancestors).

In a cEA whose population is structured as a random graph, since the number of edges incident on a given edge is a binomial random variable, we can use the mean field hypothesis, which consists in taking for all vertices the average number of neighbors \( q(n-1) \). In this way, any vertex “sees” the same isotropic average environment. Under this hypothesis, the expected number of copies of the best individual in a neighborhood not containing a copy of the best at time step \( t \) is \( E[B_{q(n-1)}] = qN(t) \). We will see that this approximation is good unless the probability \( q \) is very low. In the case of uniform selection, the probability that a copy of the best is selected at time step \( t \) is

\[ p_{\text{sel}}^{\text{rnd}}(q(n-1), qN(t)) = \frac{qN(t)}{q(n-1)} = \frac{N(t)}{n-1}. \]  

This approximation, valid when the mean field hypothesis can be used, gives a probability equal to the panmictic case, where the graph describing the topology of a population is a complete graph. In fact, for this structure, each individual has exactly \( K = n-1 \) neighbors,
and the number of copies of the best individual in a neighborhood not containing a copy of the best at time step $t$ is $B_{n-1} = N(t)$. In the case of uniform selection, the probability that a copy of the best is selected at time step $t$ is

$$p_{\text{pan}}(n-1, N(t)) = \frac{N(t)}{n-1}. \quad (7.6)$$

Therefore, the selection pressure for a randomly structured population is similar to that of a panmictic one, when the mean field hypothesis can be applied.

For synchronous update, using equations 7.1 and 7.6, the recurrence for $N(t)$ can be written as

$$N(0) = 1 \quad E[N(t)] = E[N(t-1)] + (n - E[N(t-1)]) \frac{E[N(t-1)]}{n}, \quad (7.7)$$

which is a typical form of a discrete logistic recurrence.

In the asynchronous new random sweep case, the $n - E[N(t-1)]$ individuals not containing a copy of the best in the previous time step will contain a copy of the best following the recurrence

$$E[M(1)] = E[N(t-1)] (1 + \frac{1}{n})$$

$$E[M(\tau)] = E[N(t-1)] (1 + \frac{1}{n})^\tau,$$

where $M(\tau)$ is the probability that individual at position $\tau$ in the sweep, among those not containing the best, will be taken over. Therefore, the growth of $N(t)$ can be described by the following recurrence:

$$N(0) = 1 \quad E[N(t)] = E[N(t-1)] + (1 + \frac{1}{n})^{n-E[N(t-1)]}. \quad (7.8)$$

When employing an asynchronous uniform choice update policy, the growth of random
variable $N(u)$ in terms of single update step $u$ can be described by the recurrence:

$$
\begin{align*}
N(0) &= 1 \\
E[N(u)] &= E[N(u - 1)] + \frac{n-E[N(u-1)]}{n} E[N(u-1)].
\end{align*}
$$

(7.9)

### 7.3 Experimental Results on Random Graph Structures

We report experimental data averaged over 100 independent runs for the three update policies and for panmictic and randomly structured populations, using uniform selection described and modeled in Section 9.3. Although, as stated in that section, we do not provide mathematical models for linear ranking and binary tournament selection, experiments using those selection methods, not shown here for lack of space, give qualitatively similar results. Note also that, although the curves are represented as being continuous for the sake of clarity, they are obviously discrete. In all the curves of Figure 7.1 the population grows exponentially at first and then saturates, giving the usual sigmoidal shape for the growth curves. However, one can clearly distinguish the three update policies with NRS being faster than synchronous. The UC policy starts similar to NRS and then joins the synchronous case when saturation sets in.

Figure 7.1 (a) depicts the theoretical curves corresponding to the three update modes given in the previous section. Of course, according to the mean field approximation, the panmictic and the random graph cases are actually the same. This is clearly confirmed by Figures 7.1 (b) and (c), which show, respectively, the panmictic experimental curves and the random graph case with probability $q = 0.1$. Figure 7.1 (d) shows the experimental random graph case with $q = 0.01$: we observe that for low probabilities the mean field hypothesis gives a worse approximation of the experimental results. It should be noted that for low $q$ values there is a non-negligible probability that the generated random graph is disconnected. To avoid these cases, we only consider connected graphs in our experiments, randomly sampled among the family $G_{n,q}$ of all possible random graphs with $n$ vertices and edge probability $q$.

Table 7.1 gives the predicted and the experimental average takeover times for the three update modes for panmictic and randomly structured populations.
In Figures 7.2 and 7.3 we provide a direct comparison by superposing on the same graph the theoretical curves for the synchronous (a), asynchronous NRS (b), and asynchronous UC (c) updates, and ten randomly chosen corresponding experimental curves. This is more informative than a comparison with the average experimental curves since the theoretical result is an expectation curve.

Figure 7.2 reports results for the random graph with $q = 0.1$; it is clear that there is a very good agreement between the prediction of the models and experiments. As stated above, we cannot hope that such an agreement also holds for random graphs with very
Table 7.1: Predicted takeover times and experimental mean takeover times (with corresponding standard deviations) for the three update methods. The experimental results are obtained over 100 independent runs. Population size is 1024 in all cases.

<table>
<thead>
<tr>
<th>Method</th>
<th>Predicted Takeover Time</th>
<th>Synchronous</th>
<th>Asynch NRS</th>
<th>Asynch UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Takeover Time</td>
<td></td>
<td>14</td>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td>Panmictic Mean Takeover Time</td>
<td></td>
<td>14.65 (1.43)</td>
<td>11.47 (1.28)</td>
<td>16.6 (1.73)</td>
</tr>
<tr>
<td>Random (q=0.05) Mean Takeover Time</td>
<td></td>
<td>15.49 (1.46)</td>
<td>12.2 (1.2)</td>
<td>17.07 (1.9)</td>
</tr>
<tr>
<td>Random (q=0.01) Mean Takeover Time</td>
<td></td>
<td>19.5 (2.55)</td>
<td>15.56 (2.01)</td>
<td>20.2 (2.09)</td>
</tr>
</tbody>
</table>

Figure 7.2: Theoretical curve (black) and ten randomly chosen experimental curves (gray) for a random graph population with $q = 0.05$ using synchronous update (a), asynchronous NRS (b), asynchronous UC (c).

low probability $q$. In fact, Figure 7.3 shows that the approximation is much worse. This is understandable qualitatively on the following grounds. In a random graph the node degree is binomially distributed by construction. Therefore, for 1024 vertices, the average number of neighbors of an individual is about 50 for $q = 0.05$ and about 10 for $q = 0.01$. The standard deviation in the latter case is around 3, while it is around 7 in the former in absolute terms; however, in terms of relative deviation (i.e., $\sigma/\mu$), we have around $1/3$ for $q = 0.01$, ca. 228% more than around $1/7$ for $q = 0.05$. This means that a significant number of nodes will have much fewer edges than the average in the $q = 0.01$ case, which will slow down the propagation rate of the best individual.


7.4. SMALL-WORLD GRAPH TOPOLOGIES

It has been shown in recent years that graphs that occur in many social, biological, and man-made systems are often neither completely regular, such as lattices, nor completely random (Watts and Strogatz, 1998; Watts, 1999). They have instead what has been called a small-world topology, in which nodes are highly clustered yet the path length between them is small. This behavior is due to the presence of shortcuts i.e., a few direct links between nodes that would otherwise be far removed. Following Watts’ and Strogatz’s discovery, Barabasi et al. (Barabasi and Albert, 1999) found that several important networks such as the World Wide Web, Internet, author citation networks, and metabolic networks among others, also have the small world property but their degree distribution function differs: they have more nodes of high degree that are likely in a random graph of the same size and edge density. These graphs have been called scale-free because the degree probability distribution function follows a power law. In the next sections we briefly describe how small-world and scale-free graphs can be constructed, more details can be found in (Albert and Barabasi, 2002; Newman, 2003).

7.4.1 The Watts-Strogatz Model

Although this model has been a real breakthrough in the technical sense when it appeared, today it is clear that it is not a good representation of real networks as it retains many features of the random graph model. In fact, scale-free and other types of graphs have been

![Figure 7.3: Theoretical curve (black) and ten randomly chosen experimental curves (gray) for a random graph population with $q = 0.01$ using synchronous update (a), asynchronous NRS (b), asynchronous UC (c).](image)
successively proposed as more faithful description of the kind of big technological, human, and biological networks we observe. In spite of this, the Watts-Strogatz model, because of its simplicity of construction and the richness of behavior, is still an interesting topology in artificial systems where there is no “natural” constraint on the type of connectivity.

According to Watts and Strogatz (Watts and Strogatz, 1998), 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 $\beta$. When the edge is deleted, it is replaced with an edge to a randomly chosen node. If rewiring an edge would lead to a duplicate edge, it is left unchanged. This procedure will create a number of *shortcuts* that join distant parts of the lattice. Figure 7.4 schematically depicts this process for a small ring with $k = 4$.

![Figure 7.4](image-url)

Figure 7.4: (a) regular one-dimensional lattice with $k = 4$. (b) a small-world graph obtained by randomly rewiring some of the nearest neighbor links.

Shortcuts are the hallmark of small worlds. While the average path length\(^1\) between nodes scales logarithmically in the number of nodes for a random graph, in Watts-Strogatz graphs it scales approximately linearly for low rewiring probability but goes down very quickly and tends to the random graph limit as $\beta$ 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

---

\(^1\)The average path length $L$ of a graph is the average value of all pair shortest paths.
Small-world networks have a degree distribution $P(k)$ i.e., the probability that a randomly selected vertex has degree $k$, that is close to binomial for intermediate and large values of the rewiring probability $\beta$ as for a random graph. $P(k)$ tends to a delta function for $\beta \to 0$ since in this case we recover the regular lattice.

### 7.4.2 The Barabási-Albert Model

Albert and Barabási were the first to realize that 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 that is still useful based on these ideas (Barabasi and Albert, 1999). At the beginning one starts with a small clique (a completely connected graph) of $m_0$ nodes. 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 connects, it is assumed that the probability $\pi$ that a new node will be connected to node $i$ depends on the degree $k_i$ of $i$ such that nodes that have already many links are more likely to be chosen over those that have few. This is called preferential attachment and is an effect that can be observed in real networks. The probability $\pi$ is given by:

$$\pi(k_i) = \frac{k_i}{\sum_j k_j},$$

where the sum is over all nodes already in the graph. Barabási and Albert have shown that the model evolves into a stationary scale-free network with a power-law probability distribution for the vertex degree $P(k) \sim k^{-\gamma}$, with $\gamma \sim 3$.

There exist other more general and more refined models that are capable of producing graphs with a power law degree distribution (see, e.g. (Dorogovtsev and Mendes, 2003)). However, the basic Barabási-Albert model is enough for our initial investigation.

---

2The 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.
7.5 Experimental Results on Small Worlds

In all the experiments described below we have used a population size of 1024, and a total number of edges of the same order of that of a random graph with $q = 0.01$. The selection mechanism employed has been uniform selection in all cases. All the curves are averages of 100 independent runs.

7.5.1 Watts-Strogatz Model

We have used Watts’ small world construction with 1024 individuals starting from a ring with $k = 10$ neighbors i.e., a regular radius-five one-dimensional lattice. In this way the mean number of neighbors (10.24) is equal to that of random graphs with $q = 0.01$.

![Figure 7.5: Growth curves for synchronous update for different values of the rewiring probability $\beta$. Rightmost curve is for a ring ($\beta = 0$). Leftmost curve corresponds to $\beta = 0.8$, which is almost in the random graph region.](image)

Figure 7.5 shows the growth curves with synchronous update for different values of the rewiring probability $\beta$ and for the ring. The trend is clear: increasing $\beta$ from 0 (the ring case), to 0.8 (topologies approaching that of a random graph), the selection pressure increases slowly at first and then very quickly around $\beta = 0.005$. This can be easily understood if one takes into account how the mean path length and the clustering coefficient vary in a small-world graph. From Figure 7.6 (redrawn from (Watts, 1999)) one can see that for $\beta$ around 0.005 there is a sudden drop in the average path length from values that pertain to the lattice to values that are close to those of a random graph. This means that, suddenly, many short paths become available through the network between
most nodes, which explains the higher growth rate.

Figure 7.6: Average path length and clustering coefficient for small-world graphs as a function of the rewiring probability $\beta$. Note the log scale on the abscissae.

Figure 7.7 depicts the growth curves for synchronous update, and for the two asynchronous policies in small worlds with $\beta = 0.005$. As for panmictic populations and random graphs, new random sweep is faster than uniform choice, which is in turn faster than synchronous. The experimental takeover time values are to be compared with those of random graphs with the same average number of edges i.e., figure 7.1 (d). Clearly, the corresponding small-world graphs induce a lower global selection pressure.

Figure 7.7: Growth curves for synchronous, new random sweep, and uniform choice asynchronous update. The rewiring probability is $\beta = 0.05$. Please note the change of horizontal axis scale with respect to Figure 7.5.
7.5.2 Barabási-Albert Model

Scale-free graphs have been generated according to the Barabási-Albert model of section 7.4.2. The algorithm starts from a clique of \( m_0 = 14 \) nodes, and we add \( 1024 - 14 = 1010 \) individuals, each creating \( m = 10 \) edges with preferential attachment, following the algorithm.

Figure 7.8 shows the behavior of the growth curves for the three update policies used here. For the position of the initial best individual any vertex is equally likely. The order of the curves is the same as that observed for random graphs and Watts-Strogatz small-world networks. The inversion in the last part of the uniform choice curve is due to the fact that cells are chosen with replacement, and thus the last few non-conquered individuals are increasingly unlikely to be chosen. Apart from this effect, the takeover times are very close to those observed in the corresponding random graphs. This confirms that scale-free graphs are a topology in which propagation is at least as fast as for random graphs which, for example, has important consequences in infection rates (Newman, 2003).

![Figure 7.8: Growth curves for synchronous, new random sweep, and uniform choice asynchronous update in scale-free graphs. Initial best individual uniformly distributed at random among the nodes.](image)

But scale-free graphs have other surprising properties. In particular, those networks are extremely tolerant to attacks on randomly chosen target nodes, which is due to the fact that there are few important (highly connected) nodes and many unimportant (sparsely connected) ones. On the other hand, deliberate suppression of highly connected nodes is likely to produce a lot of damage (Albert and Barabasi, 2002). The different status of
Table 7.2: Experimental mean takeover times (with corresponding standard deviations) for the three update methods and for the small-world topologies discussed in the text. WS stands for Watts-Strogatz and BA stands for Barabási-Albert. The experimental results are obtained over 100 independent runs. Population size is 1024 in all cases.

<table>
<thead>
<tr>
<th>Topology</th>
<th>Synchronous</th>
<th>Asynch NRS</th>
<th>Asynch UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring ($\beta = 0$)</td>
<td>279.64 (9.35)</td>
<td>201.51 (9.03)</td>
<td>229.30 (10.54)</td>
</tr>
<tr>
<td>WS ($\beta = 0.001$)</td>
<td>168.89 (45.93)</td>
<td>116.50 (27.12)</td>
<td>135.15 (35.10)</td>
</tr>
<tr>
<td>WS ($\beta = 0.005$)</td>
<td>80.16 (14.47)</td>
<td>60.18 (8.75)</td>
<td>70.43 (12.08)</td>
</tr>
<tr>
<td>WS ($\beta = 0.02$)</td>
<td>45.96 (4.32)</td>
<td>36.58 (4.15)</td>
<td>41.31 (4.27)</td>
</tr>
<tr>
<td>WS ($\beta = 0.8$)</td>
<td>19.16 (1.69)</td>
<td>15.43 (1.76)</td>
<td>20.50 (2.46)</td>
</tr>
<tr>
<td>BA</td>
<td>17.94 (2.59)</td>
<td>14.60 (2.66)</td>
<td>19.77 (3.55)</td>
</tr>
</tbody>
</table>

highly connected nodes is demonstrated in the following experiment, where the initial best individual has always been placed in a “hub” node (see Figure 7.9).

The takeover time is very short; shorter than the random graph case (see Figure 7.1 and Table 7.1). This is also known to happen in infectious processes, where scale-free communication patterns have the effect of eliminating the so-called infection threshold (Newman, 2003).

Table 7.2 summarizes numerical results of the takeover times in Watts-Strogatz and scale-free topologies for the synchronous and the two asynchronous updates.

One could probably exploit the effect of small-world topologies on the dynamical properties of evolutionary computation processes by letting the topology dynamically adapt
or self-organize in order to control the selection pressure, and thus the explorative or exploitative characteristics of the algorithm.

7.6 Summary

In the first part of this chapter we presented general stochastic models for individual growth that are valid for any graph-structured population, including the classical panmictic EA as a limiting case. We have given discrete difference equations for the synchronous and two typical asynchronous cell update mechanisms, which are new, as far as we can tell, at least for the asynchronous cases. These discrete logistic recurrence equations seem to us more adequate for finite populations than the usual continuous logistic fitting.

Although finding closed forms for the recurrences appears to be mathematically very difficult, the comparison of predicted growth curves with experimental ones confirms that the models are a good description of the propagation phenomena. We have used a selection method that lends itself to an approximate mathematical treatment in terms of the mean-field hypothesis. Under this hypothesis, we found that the panmictic and random graph cases have the same behavior, unless the edge probability of the random graph is very small. A practical consequence of our results is that it appears unnecessary to use the whole population as a selection pool, given that the random graph case using a fraction of the population shows the same behavior as the panmictic one. The models and the experiments also confirm previous results for regular lattices, i.e., that the selection pressure can be varied using different updating schemes. In the future we intend to pursue the study of propagation under classical selection schemes such as linear ranking and tournament.

In the second part of this chapter we presented an experimental investigation of individual growth in a couple of families of networks that are neither regular nor random: the Watts-Strogatz model and the scale-free Barabási-Albert model. We have empirically shown that the individual propagation properties and the global induced selection pressure are qualitatively similar to those found in panmictic populations and random graphs. However, the inhomogeneity of these small-world networks opens up new possibilities for evolutionary computation when the nature of a given node is taken into account. Thus, for
example, hubs in scale-free networks largely determine the dynamical properties of the population. By controlling these features, or allowing the population network to self-organize, it is possible to change the selection pressure and thus the algorithm characteristics within a wide range.
Part III

Artificial Evolution of Networks
Chapter 8

Cellular Automata

You can fool all the people some of the time. You can even fool some of the people all the time. But you can’t fool all the people all the time.

– Abraham Lincoln

Cellular automata are dynamical systems where locally interacting elementary elements evolve in time producing interesting global behaviors. They provide simple discrete mathematical models for physical, biological and computational systems. Despite their simple construction, cellular automata are capable of complicated behavior, and can be used to generate complex patterns with universal features. They can be compared to insect colonies where a single individual has a very limited knowledge and strength but achieves remarkable tasks when a colony unites its individual forces.

A brief introduction to these dynamical systems will be given in the next section, followed by a more formal mathematical definition in Section 8.2. Then, in Section 8.3, we will present the four principal variants of the original model: non-uniform cellular automata (Section 8.3.1), non-standard architectures (Section 8.3.2), asynchronous cellular automata (Section 8.3.3), and probabilistic cellular automata (Section 8.3.4). In the next chapter the non-standard architecture variant will be evolved in order to solve the two cellular automata problems that will be presented in the last section of this chapter.
8.1 Introduction

Cellular automata (CAs) were originally conceived by Ulam and von Neumann in the 1940s to provide a formal framework for investigating the behavior of complex, extended systems (von Neumann, 1966). CAs are dynamical systems in which space and time are discrete. A cellular automaton consists of an array of cells, each of which can be in one of a finite number of possible states, updated synchronously in discrete time steps, according to a local, identical interaction rule. The state of a cell at the next time step is determined by the current states of a surrounding neighborhood of cells (Sipper, 1997; Toffoli and Margolus, 1987; Wolfram, 1994).

The cellular array (grid) is usually an $d$-dimensional regular lattice, where $d = 1, 2, 3$ is used in practice; in this part we will concentrate on $d = 1, 2$, i.e., one- and two-dimensional grids. The identical rule contained in each cell is essentially a finite state machine, usually specified in the form of a rule table (also known as the transition function), with an entry for every possible neighborhood configuration of states. The cellular neighborhood of a cell consists of the surrounding (adjacent) cells. For one-dimensional CAs, a cell is connected to $r$ local neighbors (cells) on either side, as well as to itself, where $r$ is a parameter referred to as the radius (thus, each cell has $2r + 1$ neighbors). For two-dimensional CAs, two types of cellular neighborhoods are usually considered: 5 cells, consisting of the cell along with its four immediate nondiagonal neighbors (usually called von Neumann neighborhood), and 9 cells, consisting of the cell along with its eight surrounding neighbors (usually called Moore neighborhood). When considering a finite-size grid, spatially periodic boundary conditions are frequently applied, resulting in a circular grid for the one-dimensional case, and a toroidal one for the two-dimensional case.

As an example, let us consider the parity rule (also known as the XOR rule) for a 2-state, 5-neighbor, two-dimensional CA. Each cell is assigned a state of 1 at the next time step if the parity of its current state and the states of its four neighbors is odd, and is assigned a state of 0 if the parity is even (alternatively, this may be considered a modulo-2 addition). The rule table consists of entries of the form:
8.2. FORMAL DEFINITIONS

<table>
<thead>
<tr>
<th>CNESW</th>
<th>$S_{\text{next}}$</th>
<th>CNESW</th>
<th>$S_{\text{next}}$</th>
<th>CNESW</th>
<th>$S_{\text{next}}$</th>
<th>CNESW</th>
<th>$S_{\text{next}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>0</td>
<td>00010</td>
<td>1</td>
<td>01000</td>
<td>0</td>
<td>10001</td>
<td>1</td>
</tr>
<tr>
<td>00001</td>
<td>1</td>
<td>00100</td>
<td>0</td>
<td>01001</td>
<td>0</td>
<td>10010</td>
<td>0</td>
</tr>
<tr>
<td>00011</td>
<td>0</td>
<td>01010</td>
<td>1</td>
<td>10011</td>
<td>1</td>
<td>11011</td>
<td>0</td>
</tr>
<tr>
<td>00100</td>
<td>1</td>
<td>01100</td>
<td>0</td>
<td>10100</td>
<td>0</td>
<td>11100</td>
<td>1</td>
</tr>
<tr>
<td>00101</td>
<td>0</td>
<td>01101</td>
<td>1</td>
<td>10101</td>
<td>1</td>
<td>11101</td>
<td>0</td>
</tr>
<tr>
<td>00110</td>
<td>0</td>
<td>01110</td>
<td>1</td>
<td>10110</td>
<td>1</td>
<td>11110</td>
<td>0</td>
</tr>
<tr>
<td>00111</td>
<td>1</td>
<td>01111</td>
<td>0</td>
<td>10111</td>
<td>0</td>
<td>11111</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 8.1: Parity rule table. CNESW denotes the current states of the center, north, east, south, and west cells, respectively. $S_{\text{next}}$ is the cell’s state at the next time step.

\[
\begin{array}{c|c|c}
0 & 1 & 0 \\
1 & 1 & \rightarrow 1 \\
1 & 0 & \\
\end{array}
\]

This means that if the current state of the cell is 1 and the states of the north, east, south, and west cells are 0, 0, 1, 1, respectively, then the state of the cell at the next time step will be 1 (odd parity). The rule is completely specified by the rule table given in Table 8.1. Figure 8.1 demonstrates patterns that are produced by the parity CA.

8.2 Formal Definitions

A $d$-dimensional CA consists of a finite or infinite $d$-dimensional grid of cells, each of which can take on a value from a finite, typically small, set of integers. 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.

Formally, a cellular automaton $A$ is a quadruple

$$A = (S, G, d, f),$$

where $S$ 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, $i \in \mathbb{Z}^d$, in a regular $d$-dimensional uniform lattice, or grid (i.e., $i$ is an integer vector in a $d$-dimensional space), its neighborhood $G$ is defined by:

$$G_i = \{i, i + r_1, i + r_2, \ldots, i + r_k\},$$

where $k$ is a fixed parameter that determines the neighborhood size, and $r_j$ is a fixed vector in the $d$-dimensional space.
The local transition rule $f$

$$f : S^{k+1} \rightarrow S$$

maps the state $s_i \in S$ of a given cell $i$ into another state from the set $S$, as a function of the states of the cells in the neighborhood $G_i$. 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 $i$ (see Section 8.3.1).

For a finite-size CA of size $N$ a configuration of the grid at time $t$ is defined as

$$C(t) = (s_0(t), s_1(t), \ldots, s_{k-1}(t)),$$

where $s_i(t) \in S$ is the state of cell $i$ at time $t$. The progression of the CA in time is then given by the iteration of the global mapping $F$

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

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 (Wolfram, 1994).

A well explored system is that of one-dimensional CAs with two possible states per cell, i.e., $S = \{0, 1\}$. In this case $f$ is a function $f : \{0, 1\}^k \rightarrow \{0, 1\}$ and the neighborhood size $k$ is usually taken to be $k = 2r + 1$ such that:

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

where $r \in \mathbb{Z}^+$ is a parameter, known as the radius, representing the standard one-dimensional cellular neighborhood. Considering the $r = 1$ case one obtains so-called elementary CAs, for which the neighborhood size is $k = 3$: $f : \{0, 1\}^3 \rightarrow \{0, 1\}$, $s_i(t+1) = f(s_{i-1}(t), s_i(t), s_{i+1}(t))$.

The domain of $f$ is the set of all $2^3$ 3-tuples, which gives rise to $2^8 = 256$ distinct elementary
rules. It is common to use Wolfram’s decimal numbering convention for describing these rules (Wolfram, 1994). For two-state CAs, a configuration of a size $N$ grid at time $t$ is a binary sequence $C(t)$, $C(t) \in \{0, 1\}^N$. For finite-size grids, spatially periodic boundary conditions are frequently assumed, resulting in a circular grid; formally, this implies that cellular indices are computed modulo $N$.

8.3 Variations on the Original Model

In this section we briefly outline a number of variations of the original, classic CA model, presented above. These variations concern the cellular rules, the connectivity architectures, temporal considerations, and determinism.

8.3.1 Non-Uniform Cellular Automata

Non-uniform cellular automata function in the same way as uniform ones, the only difference being in the cellular rules that need not be identical for all cells. Note that non-uniform CAs share the basic “attractive” properties of uniform ones (simplicity, parallelism, locality). They have been investigated by Vichniac et al. (1986) who discuss a 1-D CA in which a cell probabilistically selects one of two rules at each time step, and by Sipper (1994). They showed that complex patterns appear characteristic of class IV behavior. Garzon (1995) presented two generalizations of cellular automata, namely, discrete neural networks and automata networks. These were compared to the original model from a computational point of view which considers the classes of problems such models can solve.

8.3.2 Non-Standard Architectures

Another possible variation concerns the connectivity pattern of the cells, the architecture, which is regular and homogeneous in the original CA. One can consider so-called non-standard connectivity architectures, where each cell has a small, identical number

\[ f(111) = 1, \quad f(110) = 0, \quad f(101) = 1, \quad f(100) = 1, \quad f(011) = 1, \quad f(010) = 0, \quad f(001) = 0, \quad f(000) = 0, \] is denoted rule 184 (the decimal equivalent of 10111000).
of connections, yet not necessarily from its most immediate neighboring cells. It can be shown that such architectures are computationally more efficient than standard architectures in solving certain computational tasks. Watts (1999) has recently shown that cellular automata having small-world topologies have interesting computational properties.

8.3.3 Asynchronous Cellular Automata

One of the prominent features of the CA model is its synchronous mode of operation, meaning that all cells are updated simultaneously. A preliminary study of asynchronous CAs, where one cell is updated at each time step, was carried out by (Ingerson and Buvel, 1984), where the different dynamical behavior of synchronous and asynchronous CAs was compared; the authors argued that some of the apparent self-organization of CAs is an artifact of the synchronization of the clocks. Wolfram (Wolfram, 1994) noted that asynchronous updating makes it more difficult for information to propagate through the CA and that, furthermore, such CAs may be harder to analyze. Asynchronous CAs have also been discussed in Nowak et al. (1994) and Bersini and Detour (1994).

8.3.4 Probabilistic Cellular Automata

In a deterministic cellular automaton, for any given input, the system always goes through the same trajectory of states, ending with the same output. For a nondeterministic, or probabilistic CA the same input may result in different trajectories, and possibly different outputs. Nondeterminism may be inherent to the system’s functional definition or it may result due to faults. As an example, consider a two-state CA, where a cell updates its state in a non-deterministic manner, setting it at the next time step to that specified in the rule table, with probability $1 - p_f$, or the complementary state, with probability $p_f$. The value $p_f$ can be regarded as the probability that a cell will malfunction: this type of fault was studied, e.g., by Sipper et al. (1996).
8.4 Cellular automata as computational systems

As noted above, the CA model was originally introduced in the late 1940s by Ulam and von Neumann and used extensively by the latter to study issues related with the logic of life (von Neumann, 1966). In particular, von Neumann asked whether we can use purely mathematical-logical considerations to discover the specific features of biological automata that make them self-replicating.

Von Neumann used two-dimensional CAs with 29 states per cell and a 5-cell neighborhood. He showed that a universal computer can be embedded in such cellular space, namely, a device whose computational power is equivalent to that of a universal Turing machine (Hopcroft and Ullman, 1979). He also described how a universal constructor may be built, namely, a machine capable of constructing, through the use of a “constructing arm,” any configuration whose description can be stored on its input tape. This universal constructor is therefore capable, given its own description, of constructing a copy of itself, i.e., of self replicating. The terms ‘machine’ and ‘tape’ refer here to configurations, i.e., patterns of states (as defined in Section 8.2). The mechanisms von Neumann proposed for achieving self-replicating structures within a cellular automaton bear strong resemblance to those employed by biological life, discovered during the following decade. Von Neumann’s universal computer-constructor was simplified by Codd (1968) who used an 8-state, 5-neighbor cellular space.

Over the years CAs have been applied to the study of general phenomenological aspects of the world, including communication, computation, construction, growth, reproduction, competition, chemical systems, and evolution (see, e.g., Burks (1970); Sipper (1997); Toffoli and Margolus (1987); Kier et al. (2005)). One of the most well-known CA rules, the “game of life,” was conceived by Conway in the late 1960s and was shown by him to be computation-universal (Berlekamp et al., 1982). For a review of computation-theoretic CA results refer to Culik II et al. (1990).

The question of whether cellular automata can model not only general phenomenological aspects of our world, but also directly model the laws of physics themselves was raised by Fredkin and Toffoli (1982); Toffoli (1977). A primary theme of this research is the
formulation of computational models of physics that are information-preserving, and thus retain one of the most fundamental features of microscopic physics, namely, reversibility (Fredkin and Toffoli, 1982; Margolus, 1984; Toffoli, 1980). This approach has been used to provide extremely simple models of common differential equations of physics, such as the heat and wave equations (Toffoli, 1984) and the Navier-Stokes equation (Frisch et al., 1986). CAs also provide a useful model for a branch of dynamical systems theory which studies the emergence of well-characterized collective phenomena, such as ordering, turbulence, chaos, symmetry-breaking, and fractality, in discrete systems (Chopard and Droz, 1997; Vichniac, 1984).

In recent years there is a growing interest in the utilization of CAs as actual computing devices. CAs exhibit three notable features: massive parallelism, locality of cellular interactions, and simplicity of basic components (cells). They perform computations in a distributed fashion on a spatially extended grid. As such they differ from the standard approach to parallel computation in which a problem is split into independent sub-problems, each solved by a different processor, later to be combined in order to yield the final solution. CAs suggest a new approach in which complex behavior arises in a bottom-up manner from non-linear, spatially extended, local interactions. This is often referred to as emergent computation, meaning the appearance of global information processing capabilities that are not explicitly represented in the system’s elementary components or in their local interconnections (Forrest, 1991). The CA’s properties greatly facilitate its implementation as electronic hardware (Sipper et al., 1997). CAs also suggest a possible approach to attaining novel computational architectures at the nanometer scale (Benjamin and Johnson, 1997).

When considering CAs that perform computations two possibilities manifest themselves: (1) Embedding a universal Turing machine within the CA, or (2) using the CA in a direct, parallel manner: the input to the computation is encoded as an initial configuration, the output is the configuration after a certain number of time steps, and the intermediate steps that transform the input to the output are considered to be the steps in the computation. In this latter case, the “program” emerges through “execution” of the CA rule in each cell. In the next subsections, I will introduce two classical computa-
tional problems for CAs that will be used in the next chapter, the density task and the synchronization task, which are examples of the second kind of CA computation above.

8.4.1 The majority task

The density (also called majority) task is a prototypical distributed computational task for CAs. For a finite CA of size $N$ it is defined as follows. Let $\rho^0$ be the fraction of 1s in the initial configuration (IC) $s^0$. The task is to determine whether $\rho^0$ 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 (Mitchell et al., 1993).

It has been shown that the density task cannot be solved perfectly by a uniform, two-state CA with finite radius (Land and Belew, 1995), although a slightly modified version of the task can be shown to admit a perfect solution by such an automaton (Capcarrère et al., 1996).

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 computationally
expensive except for elementary \((d = 1, r = 1)\) and other fairly simple automata, one possible solution of the problem consists in using evolutionary algorithms, as first proposed by Mitchell \textit{et al.} (Mitchell et al., 1994, 1993) for uniform CAs, and by Sipper for nonuniform ones (Sipper, 1997). Mitchell \textit{et al.} have shown empirically that homogeneous two-state ring CAs of radius three can be evolved and are capable of reaching a fairly high performance (see below). With \(d = 1\) and \(r = 3\) there are \(2^{128} \sim 10^{36}\) possible rules that the evolutionary algorithm has to search through.

Watts (1999) 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 \textit{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 several networks with performance values \(P > 0.8\), while the best evolved lattices with the same average number of neighbors had \(P\) around 0.77 (Mitchell et al., 1994, 1993) and were difficult to obtain. In fact, according to (Crutchfield et al., 2003), high-performance strategies obtained only nine times in 300 runs.

In a remarkable paper, Sipper and Ruppin (1997) 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.

8.4.2 The synchronization task

The one-dimensional synchronization task was introduced in Das et al. (1995). In this task the CA, given an arbitrary initial configuration, must reach a final configuration, within \( M \approx 2N \) time steps, that oscillates between all 0s and all 1s on successive time steps.

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 (Das et al., 1995). Sipper did the same for quasi-homogeneous CAs, i.e. CAs with a few different rules instead of just one (Sipper, 1997), 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 \( M \approx 2N \) time steps.

Watts (1999) also has a brief section on the synchronization task on small worlds. He 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.
Chapter 9

Automata Network Evolution

The value of these mathematical bridges is enormous. They allow mathematicians who live on separate islands to exchange ideas and to examine each others’ discoveries. Mathematics consist in islands of knowledge in a sea of ignorance.

– Simon Singh

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 (Leighton, 1992), while lattices and random networks of simple automata have also received a great deal of attention (Garzon, 1995; Kauffman, 1993). 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 (Watts, 1999) in which cellular automata (CAs) computation on small-world networks is examined for some representative network structures.

However, there is no hint in these works as to how such automata 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 (see Section 7.4.2) has given good results (Albert and Barabasi, 2002). 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. Indeed, many actual networks seem to have undergone some kind of Darwinian variation and selection process.

Thus, how these automata networks might have come to be selected is an interesting yet unanswered question. In this chapter, 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 density problem and the synchronization task described in Sections 8.4.1 and 8.4.2, which are the same that Watts discusses in (Watts, 1999) 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.

The chapter is organised as follows: first the evolutionary algorithm setting will be described in Section 9.1. The two problems’ evolutions will be separately analysed in Sections 9.2 and 9.3, and in Section 9.4. A new fitness function to limit the number of shortcuts of the evolved networks will be defined and used in Section 9.5. Last, the task flexibility and the fault tolerance of the evolved networks will be analysed in Sections 9.6 and 9.7 respectively. Finally, in the last section we will summarise the result and we will present the future perspectives opened by this investigation.

### 9.1 The Evolutionary Machinery

Evolutionary algorithms have been successfully used for more than ten years to evolve network topologies for artificial neural networks, and several techniques are available (Tettamanzi and Tomassini, 2001). As far as the network topology is concerned, the problem that we describe in this chapter is similar. We use an unsophisticated cEA with the aim of evolving small-world networks for the density and synchronization problems presented in the previous chapter. The choice of a cEA is mainly due to their characteristic of steady diffusion of good solutions in the population due to a less intense selection pressure seen
in previous chapters. A simple panmictic EA would probably be equally effective.

The population is arranged on a $20 \times 20$ square grid for a total of 400 individuals. The individual’s encoding represents an undirected network with identifiable vertices and unweighted edges. Of course, the topology depiction must remain mutation-compatible at all times. We coded each individual as an array of integers denoting its vertices, each one of which has a list of the vertices that it is connected to (see Figure 9.1). As the graph is undirected the information is redundant (e.g., if X is connected to Y, then both have the other vertex in their own connections list).

![Figure 9.1](image)

Figure 9.1: Individual encoding is the array of vertices (a): to each vertex is attached a list of vertices to which it is connected to by an edge. The corresponding network representation is plotted in(b).

The automaton rule is the generalized majority rule in the case of the density task, and the contrarian rule for synchronization. The majority 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. The contrarian rule operates in the same way as the majority rule, except that it gives the opposite state as output.
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 $M = 2^N$ 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 von Neumann neighborhood 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 topologies and works as follows. Each node of an individual is mutated with probability $p_M = 0.5$. 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—see below). If the source vertex has already reached its maximum degree and one edge ought to be added or if it reached its minimum degree and one edge ought to be removed, the mutation will not happen on this vertex. If the same case happens with the target, another one is randomly chosen among the remaining ones. One could argue that the mutation probability of $p_M = 0.5$ is unrealistically high and it clearly does not follow nature’s way very closely. On the other hand, we consider it is a way to push evolution a little faster and to compensate for the fact that this version of the algorithm does not use recombination operators. Ultimately, having no crossover was a decision we made after close consideration and has been motivated by the excellent results obtained without it. In addition to that, recombination among individuals that are networks is a demanding task and making sure the new individuals are feasible requires a large overhead.

During the evolution the network nodes are constrained to have a maximum degree of 50. This is a number of edges we obtained by actually performing several runs of testbeds. But even in letting the nodes’ degrees increase at will (i.e. up to $N - 1$), they never actually grew bigger than 25 for $N = 149$. Therefore we decided that allowing them to reach at most a degree of 50 is a reasonably weak constraint and it actually keeps
the computational performances fair. We were also pleasantly surprised by the natural limitation that was reached by our bound-free networks. In fact we were half expecting their nodes’ degree to grow indefinitely; because the “global knowledge” brought by full connectivity is the easiest way of solving the density task.

The termination condition is reached after computing exactly 100 generations. This was also a matter of discussion. Actually the first stopping criterion was reached when any individual attained the maximum possible fitness value, which is 1. But this was met far too prematurely, sometimes after only a few generations when randomness would allow a succession of “easy-to-solve” IC. Obviously, the performances of such an individual were extremely unsatisfactory. Thus we forced evolution to 100 generations, enabling good individuals to become even fitter.

9.2 Evolution for the density task from regular lattices

In this first series of experiments we started from regular rings, which is the customary way for constructing small-world graphs (Watts, 1999). 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, to start with sufficient diversity in the population, we slightly modified each of them by adding an edge with a probability of 0.1 applied to each vertex.

Figure 9.2(a) shows the genotypical population entropy, $\phi$ (the fraction of edges in a graph that are shortcuts, i.e., edges that join vertices that would be more than two edges apart if they were not connected directly), fitness, and performance of the best individual as a function of the generation number. The curves represent data from a typical run out of 50 independent runs of the EA. Recall that performance is defined off-line as the fraction of correctly classified ICs which are binomially distributed, while fitness computes the same fraction on a uniformly distributed sample over density, which is an easier task. 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. The \( \phi \) curve is particularly interesting as it permits a direct comparison with Watts’ hand-constructed graphs (Watts, 1999). The results fully confirm his measurements, with networks having best performance clustering around \( \phi \) values between 0.6 and 0.8 (see figure 9.2(b)). The mean degree \( \langle k \rangle \) of the evolved networks is around 7, which compares well with the radius-three lattice CA case and Watts’s (see table 9.8).

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. (1994) 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. 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 9.3. It can be noted that the information transfer is much faster thanks to the distant connections, and the problem is thus solved in few steps.

![Space-time diagram of an evolved small-world CA for the density task](image_url)

**Figure 9.3**: The operation of an evolved small-world CA for the density task. The density $\rho^0$ is in 0.470 (a) and 0.523 in (b).

### 9.3 Evolution for the density task 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 was 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 $q$. 
(see Section 9.3), or not connecting it with probability $1 - q$. 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 ensuring that all the networks in the initial population were connected with an average degree $\langle k \rangle = Nq = 4.47$.

Figure 9.4: Density task. A typical evolutionary run starting from a random graph population.

Figure 9.5: Density task. The $\phi$ - performance values of the best individuals found by evolution starting from rings and random graphs. For comparison, Watts’ results are also plotted (redrawn from (Watts, 1999)).

We see again in figure 9.4 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 $\phi$ curve. While in the perturbed ring case $\phi$ starts low ($\phi$ is 0 for a lattice) and then slowly increases toward values around 0.7. In the random graph case the contrary happens: $\phi$ 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 $\phi$ region in both cases. This can be clearly seen in figure 9.5, where the best 50 individuals of all runs for both initial rings and random graphs are plotted together. The figure also reports the results of Watts for comparison and two lattice CAs that have been hand-designed or evolved with a GA (Mitchell et al., 1994). Note that the best evolved ring CA for the task known to date has been obtained by Juillé and Pollack and has a performance of about 0.86 (Juillé and Pollack, 1998). It should be noted that in both figures 9.2(a) and 9.4 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.

Figure 9.6 compares the evolution of $\phi$ over the generations for both individual kinds. In the case of the ring-based individual, we can see it becoming less organized, while the random-based one is becoming more so. Both then join and cluster around a point of high performance.

Moreover, we carried out a few tests trying to increase $N$ to 599 and 999. The results are clearly in line with Watts’ (see Table 9.1). We made 5 testbed runs on individuals with $N = 599$, where the average performance is around 0.89, $\phi = 0.54$, $C = 0.032$, $\langle k \rangle = 18.13$ and $L = 2.53$. For $N = 999$, we achieved performance of around 0.92, $\phi = 0.39$ (very random-like again), $C = 0.031$, $\langle k \rangle = 30.06$ and $L = 2.35$. These values are obtained over only 5 runs and should not be considered as accurate statistical values.

The following figure 9.7 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.

Figures 9.8 and 9.9 summarize the graph-theoretic statistical properties of the five best evolved individuals for the ring case, and for the random graph case. 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 (1999) (remember...
9.3. EVOLUTION FOR THE DENSITY TASK FROM RANDOM GRAPHS

<table>
<thead>
<tr>
<th>Ring-net</th>
<th>$\langle k \rangle$</th>
<th>C</th>
<th>L</th>
<th>$\phi$</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7.906</td>
<td>0.053</td>
<td>2.649</td>
<td>0.654</td>
<td>0.823</td>
</tr>
<tr>
<td>B</td>
<td>7.611</td>
<td>0.053</td>
<td>2.703</td>
<td>0.670</td>
<td>0.820</td>
</tr>
<tr>
<td>C</td>
<td>7.409</td>
<td>0.048</td>
<td>2.750</td>
<td>0.685</td>
<td>0.813</td>
</tr>
<tr>
<td>D</td>
<td>7.342</td>
<td>0.049</td>
<td>2.736</td>
<td>0.669</td>
<td>0.807</td>
</tr>
<tr>
<td>E</td>
<td>7.450</td>
<td>0.057</td>
<td>2.730</td>
<td>0.679</td>
<td>0.807</td>
</tr>
</tbody>
</table>

Figure 9.8: The five best evolved networks for ring-based initial population. $\langle k \rangle$ is the mean node degree. C is the clustering coefficient. L is the characteristic path length. $\phi$ is the percentage of shortcuts, and P is the network performance on the density task.

<table>
<thead>
<tr>
<th>Rand-net</th>
<th>$\langle k \rangle$</th>
<th>C</th>
<th>L</th>
<th>$\phi$</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7.798</td>
<td>–</td>
<td>2.695</td>
<td>0.664</td>
<td>0.821</td>
</tr>
<tr>
<td>B</td>
<td>7.543</td>
<td>–</td>
<td>2.736</td>
<td>0.585</td>
<td>0.812</td>
</tr>
<tr>
<td>C</td>
<td>7.355</td>
<td>–</td>
<td>2.729</td>
<td>0.686</td>
<td>0.800</td>
</tr>
<tr>
<td>D</td>
<td>7.422</td>
<td>0.062</td>
<td>2.736</td>
<td>0.631</td>
<td>0.798</td>
</tr>
<tr>
<td>E</td>
<td>6.778</td>
<td>–</td>
<td>2.858</td>
<td>0.748</td>
<td>0.797</td>
</tr>
</tbody>
</table>

Figure 9.9: The five best evolved networks for random graph-based initial population. $\langle k \rangle$ is the mean node degree. C is the clustering coefficient. L is the characteristic path length. $\phi$ is the percentage of shortcuts, and P is the network performance on the density task (a – means that the clustering coefficient is not computable since those graphs are allowed to have vertices with a degree smaller than 2).

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 $\phi$ values which, without being in the random graph regime, are nevertheless not far from it for both initial rings and initial random graphs. In other words, the networks with good performance 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 of 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 emerged from rings (see figure 9.3).

Finally, though only a few experiments were carried out, we wanted to draw some qualitative parallels between our $N = 599$ and $N = 999$ individual’s performance, Watts’
results, and the performance of both the handmade and CA-evolved. The results, summarized in Table 9.1, are interesting. Networks evolved from rings tend to have lower performance but keep $\langle k \rangle$ low and $\phi$ high, which matches Watts’ results. On the other hand, starting from random graphs, the performances are higher than Watts’ (about 0.92), but also have an average degree that is around 18 for $N = 599$ and 30 for $N = 999$. Interestingly, our random evolved graphs are nevertheless much more small-world like, at least as far as the $\phi$ factor is concerned, which is around 0.53 and 0.39 for $N = 599$ and $N = 999$ respectively.

9.4 Evolution of automata networks for the synchronization task

As for the density task, 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 9.10 (a) below shows one typical run starting from perturbed rings out of the 50 we performed to obtain the final population of solutions represented on (b). In that second figure, we emphasize the relationship between the $\phi$ factor and performances. We observe that, generally speaking, performances are slightly higher than in the case of the density task and that the $\phi$ value of the evolved individuals is lower, thus more small-worldlike. The gain in performance can be explained by the fact that the synchronization task is “easier” to perform than the density classification task. Indeed, the CA only needs to converge towards any stable point, it does not have to converge towards the majority of $\rho^0$. Again, the evolution of $\phi$ starts low as the lattice is almost regular and becomes more random over the generations.

Similar experimental runs are represented in Figure 9.11, but with an initial population constituted of random networks. Part (b) compares results obtained with ring-based and random-based individuals. We see that evolution from random graphs leads to more random-like individuals, with higher $\phi$ values and slightly lower but still appreciable performances. In contrast to Figure 9.10, Figure 9.11 shows the $\phi$ factor getting lower as the
9.4. EVOLUTION OF AUTOMATA NETWORKS FOR THE SYNCHRONIZATION TASK

Figure 9.10: (a) is a typical evolutionary run starting from a perturbed ring population for the synchronization task. (b) shows the 50 best evolved individuals in 50 independent runs.

Figure 9.11: Evolution curves of a typical run(a) and $\phi$ vs performance values (b) of the 50 best individuals found in the 50 evolutionary runs on the synchronization task starting from random graphs.

Finally, we can see in Figure 9.12 that the evolution of the two different graph-types (i.e., ring and random) has very different $\phi$ values which subsequently tend to meet halfway between order and randomness — the small-world regime — and to stabilize and cluster with $\phi$ values roughly between 0.4 and 0.8, and impressive performances between
0.75 and 0.99. Table 9.2 shows all the main properties of the 5 best evolved-individuals both in the case of rings and random ones. Unfortunately, in contrast to the density task, these data are not available in Watts (1999) for the synchronization problem.

9.5 Limiting the number of shortcuts

The network evolutions described in the previous sections 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 is still interesting to study the evolution of the same graphs with the added requirement that \( \phi \) is as low as possible. Notice, however, that this does not necessarily imply a shorter wire length in a two-dimensional physical realization.

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 \( \phi \) value. A similar approach was used by Sipper and Ruppin (1997) with the aim of minimizing the connection lengths 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. Some details of the experiments can be found in Table 9.3. We ran tests of 5 runs for each value of \( w \) between 0.1 and 1.0. As demonstrated in Table 9.3, differences are virtually insignificant. One can actually see that by pushing \( w \) too high, the performance can drop dramatically. Suspicions of this phenomenon motivated our exhaustive testing of the \( w \) factor.

![Figure 9.13](image-url)  
(a) A typical run for the evolution of a topology using the modified \( f' \) density task. (b) The \( \phi \) vs performance values of the 50 best individuals found in the 50 evolutionary runs on the modified \( f' \) density task starting from rings.

As depicted in Figures 9.13 and 9.14 for the density task, we see that the introduction of a selection pressure favoring networks with smaller \( \phi \) values is effective in evolving graph CAs that maintain high performance, equal to or better than those previously found using unconstrained evolution (see Sections 9.2 and 9.3 for comparison). Here also, starting from a population of perturbed rings or random graphs, it does not make a big difference, although as expected, starting from slightly perturbed rings, which have low \( \phi \), it tends
to favor slightly lower $\phi$ values of the evolved networks. The $\phi$ values are about 0.3 (see Figures 9.13 and 9.14), while they are about 0.7 in the previous case (Figures 9.2(b) and 9.5).

Figure 9.14: (a) A typical run for the evolution of a topology using the modified $f'$ density task. (b) The $\phi$ vs performance values of the 50 best individuals found in the 50 evolutionary runs on the modified $f'$ density task starting from random and starting from rings.

Figure 9.15: Evolution of a typical run for both ring and random-based individuals. The curves show performance vs. $\phi$, each angle in the curves is a new generation.

Again we see that during the evolution, the random-based and ring-based individuals, starting from opposite values of $\phi$, join in the small-world range. But this time the final
9.5. LIMITING THE NUMBER OF SHORTCUTS

\( \phi \) values are lower (see Figures 9.6 and 9.12).

![Figure 9.16: 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).](image)

The average degrees are somewhat higher however: 11.76 and 9.72 for ring-based and random-based graphs respectively. This compares favorably with Watts' hand-constructed networks (Figure 7.4, p. 192 in (Watts, 1999)), where one can see high-performance networks with \( \phi \) around 0.3 but with average degree \( \langle k \rangle \) equal to 12. Thus, it is clear that to some extent having more neighbors on average compensates for the reduced number of shortcut links. The best evolved solutions are summarized in Table 9.4. The degree distribution for evolved networks is shown in Figure 9.16 and confirms that the degree distribution \( P(k) \) is approximately Poissonian.

Experiments of the same type on the synchronization task, give similar results, in the sense that high-performance graph-CA's are obtained easily by artificial evolution. So similar, in fact, that we decided not to give any more details about them. The average values of \( \phi \) 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.
9.6 Task flexibility of the evolved networks

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. As noted by Watts (1999), the two tasks are nearly identical and thus this finding is not surprising. Furthermore, this remains true for the whole range of \( \phi \) values for which automata have been evolved or generated by hand.

Here, we depict several combinations of performance evaluations:

- performance on the synchronization task of individuals evolved for the density task (see Figure 9.17).

- performance on the task density of individuals evolved for the synchronization task (see Figure 9.18).

- performance on the synchronization task of individuals evolved for the density task favoring smaller \( \phi \) values (see Figure 9.19).

Figure 9.17: Performance vs \( \phi \) of networks evolved for the density task on both density and synchronization. Ring-based networks (a), random graph-based networks (b).

Figure 9.19 shows how networks evolved for the density task using \( \phi \) as a second objective (see section 9.5) are also well-suited for synchronization (of course, upon changing
Figure 9.18: Performance vs $\phi$ of networks evolved for the synchronization task on both density and synchronization. Ring-based networks (a), random graph-based networks (b).

The opposite is also true: namely, that networks evolved for the synchronization task can be used for solving the density problem.

All the considerations in the chapter so far lead to a more general question of knowing which evolution generated the most successful population for each task. We represented how well each of the evolved populations performed on each task. We have to limit ourselves to represent only the 5 best individuals out of the 50 available for obvious ease of reading.

The results are impressive. Figure 9.20 clearly show that on both the density classification and the synchronization, the best-performing individuals were evolved using the density task that was favoring the low $\phi$ values starting from slightly perturbed rings. We can state with some certainty that individuals with the lowest $\phi$ value (i.e., the most small-worldlike) are also those which perform the best, regardless of the problem they are presented with.

9.7 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.

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 the Hamming distances between unperturbed and faulty configurations, which give information on the spreading of damage (e.g., (Sipper et al., 1996) where the case of synchronous, nonuniform CAs is examined). Figure 9.21 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 (Sipper et al., 1996), 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 9.22 and 9.23. The faulty CA depicted is figure 9.23 is the best one obtained by artificial evolution in (Mitchell et al., 1994, 1993) and it is called EvCA here. It is clear that even small amounts of noise 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 \( \rho^0 = 0.416 \) whereas the one used in the small-world

Figure 9.20: This \( \phi \) vs. performance graph depicts how well all the different kinds of evolved solutions perform at the density task (a) and at the synchronisation task (b). Only the 5 best individuals are represented.
CA has $\rho^0 = 0.490$, and it is thus more difficult to classify. For completeness, we note that in a previous study, Tomassini and Venzi (2002) investigated the behavior of evolved asynchronous lattice CAs for the density task under probabilistic noise. They 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 9.21 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 (Sipper et al., 1996), while it is only about 8 in the small-world CA.

9.8 Summary

Starting from the work of Watts on small-world cellular automata, we have used an evolutionary algorithm to automatically 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-$\phi$ networks, the evolutionary process has been able to find networks with a low-$\phi$ and excellent performance for both tasks.

These results have been easy to find even though the evolutionary algorithm is an unsophisticated 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 CAs 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.
<table>
<thead>
<tr>
<th>Method</th>
<th>N</th>
<th>⟨k⟩</th>
<th>C</th>
<th>L</th>
<th>Φ</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Watts’ CA</td>
<td>599</td>
<td>6</td>
<td>N/A</td>
<td>N/A</td>
<td>~0.94</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>999</td>
<td>6</td>
<td>N/A</td>
<td>N/A</td>
<td>~0.95</td>
<td>0.86</td>
</tr>
<tr>
<td>Handmade CA</td>
<td>599</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>0.766</td>
</tr>
<tr>
<td></td>
<td>999</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>0.757</td>
</tr>
<tr>
<td>GA-made CA</td>
<td>599</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>0.725</td>
</tr>
<tr>
<td></td>
<td>999</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>0.417</td>
</tr>
<tr>
<td>Ring-based</td>
<td>599</td>
<td>6.938</td>
<td>0.020</td>
<td>3.676</td>
<td>0.896</td>
<td>0.717</td>
</tr>
<tr>
<td></td>
<td>999</td>
<td>6.735</td>
<td>0.012</td>
<td>3.756</td>
<td>0.916</td>
<td>0.698</td>
</tr>
<tr>
<td></td>
<td>599</td>
<td>6.778</td>
<td>0.011</td>
<td>3.754</td>
<td>0.924</td>
<td>0.696</td>
</tr>
<tr>
<td></td>
<td>599</td>
<td>6.518</td>
<td>0.011</td>
<td>3.819</td>
<td>0.919</td>
<td>0.679</td>
</tr>
<tr>
<td></td>
<td>599</td>
<td>6.244</td>
<td>0.010</td>
<td>3.943</td>
<td>0.919</td>
<td>0.634</td>
</tr>
<tr>
<td></td>
<td>599</td>
<td>6.244</td>
<td>0.010</td>
<td>3.943</td>
<td>0.919</td>
<td>0.634</td>
</tr>
<tr>
<td></td>
<td>999</td>
<td>6.354</td>
<td>0.009</td>
<td>4.265</td>
<td>0.954</td>
<td>0.607</td>
</tr>
<tr>
<td></td>
<td>999</td>
<td>6.294</td>
<td>0.007</td>
<td>4.288</td>
<td>0.958</td>
<td>0.601</td>
</tr>
<tr>
<td></td>
<td>999</td>
<td>6.294</td>
<td>0.007</td>
<td>4.288</td>
<td>0.958</td>
<td>0.601</td>
</tr>
<tr>
<td></td>
<td>999</td>
<td>6.332</td>
<td>0.006</td>
<td>4.270</td>
<td>0.958</td>
<td>0.586</td>
</tr>
<tr>
<td></td>
<td>999</td>
<td>6.082</td>
<td>0.009</td>
<td>4.409</td>
<td>0.943</td>
<td>0.539</td>
</tr>
<tr>
<td></td>
<td>999</td>
<td>5.940</td>
<td>0.008</td>
<td>4.464</td>
<td>0.953</td>
<td>0.479</td>
</tr>
<tr>
<td>Random-based</td>
<td>599</td>
<td>18.651</td>
<td>0.035</td>
<td>2.515</td>
<td>0.509</td>
<td>0.895</td>
</tr>
<tr>
<td></td>
<td>999</td>
<td>17.836</td>
<td>0.031</td>
<td>2.547</td>
<td>0.568</td>
<td>0.892</td>
</tr>
<tr>
<td></td>
<td>599</td>
<td>18.110</td>
<td>0.031</td>
<td>2.533</td>
<td>0.556</td>
<td>0.891</td>
</tr>
<tr>
<td></td>
<td>599</td>
<td>18.344</td>
<td>0.033</td>
<td>2.523</td>
<td>0.538</td>
<td>0.891</td>
</tr>
<tr>
<td></td>
<td>599</td>
<td>17.923</td>
<td>0.033</td>
<td>2.543</td>
<td>0.537</td>
<td>0.881</td>
</tr>
<tr>
<td></td>
<td>599</td>
<td>17.923</td>
<td>0.033</td>
<td>2.543</td>
<td>0.537</td>
<td>0.881</td>
</tr>
<tr>
<td></td>
<td>999</td>
<td>30.224</td>
<td>0.032</td>
<td>2.353</td>
<td>0.380</td>
<td>0.922</td>
</tr>
<tr>
<td></td>
<td>999</td>
<td>30.250</td>
<td>0.031</td>
<td>2.351</td>
<td>0.389</td>
<td>0.917</td>
</tr>
<tr>
<td></td>
<td>999</td>
<td>30.250</td>
<td>0.031</td>
<td>2.351</td>
<td>0.389</td>
<td>0.917</td>
</tr>
<tr>
<td></td>
<td>999</td>
<td>29.986</td>
<td>0.031</td>
<td>2.359</td>
<td>0.394</td>
<td>0.915</td>
</tr>
<tr>
<td></td>
<td>999</td>
<td>29.612</td>
<td>0.031</td>
<td>2.366</td>
<td>0.402</td>
<td>0.914</td>
</tr>
<tr>
<td></td>
<td>999</td>
<td>30.054</td>
<td>0.032</td>
<td>2.356</td>
<td>0.378</td>
<td>0.911</td>
</tr>
</tbody>
</table>

Table 9.1: Watts’, best handmade (Das et al.) and GA-evolved (Mitchell et al.) results vs. our ten best evolved networks for $N = 599$ and $N = 999$. $⟨k⟩$ is the mean node degree. $C$ is the clustering coefficient. $L$ is the characteristic path length. $\phi$ is the percentage of shortcuts, and $P$ is the network performance on the density task.
<table>
<thead>
<tr>
<th></th>
<th>(\langle k \rangle)</th>
<th>C</th>
<th>L</th>
<th>(\phi)</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring-based</td>
<td>10.456</td>
<td>0.089</td>
<td>2.392</td>
<td>0.417</td>
<td>0.989</td>
</tr>
<tr>
<td></td>
<td>10.349</td>
<td>0.080</td>
<td>2.378</td>
<td>0.446</td>
<td>0.988</td>
</tr>
<tr>
<td></td>
<td>10.188</td>
<td>0.078</td>
<td>2.426</td>
<td>0.449</td>
<td>0.988</td>
</tr>
<tr>
<td></td>
<td>10.215</td>
<td>0.065</td>
<td>2.399</td>
<td>0.490</td>
<td>0.986</td>
</tr>
<tr>
<td></td>
<td>9.396</td>
<td>0.072</td>
<td>2.459</td>
<td>0.500</td>
<td>0.984</td>
</tr>
<tr>
<td>Random-based</td>
<td>8.738</td>
<td>-</td>
<td>2.547</td>
<td>0.591</td>
<td>0.976</td>
</tr>
<tr>
<td></td>
<td>9.329</td>
<td>-</td>
<td>2.486</td>
<td>0.488</td>
<td>0.975</td>
</tr>
<tr>
<td></td>
<td>8.725</td>
<td>-</td>
<td>2.574</td>
<td>0.486</td>
<td>0.969</td>
</tr>
<tr>
<td></td>
<td>8.859</td>
<td>-</td>
<td>2.541</td>
<td>0.533</td>
<td>0.967</td>
</tr>
<tr>
<td></td>
<td>8.215</td>
<td>-</td>
<td>2.649</td>
<td>0.618</td>
<td>0.965</td>
</tr>
</tbody>
</table>

Table 9.2: The ten best evolved networks. \(\langle k \rangle\) is the mean node degree. \(C\) is the clustering coefficient. \(L\) is the characteristic path length. \(\phi\) is the percentage of shortcuts, and \(P\) is the network performance on the density task. (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).
<table>
<thead>
<tr>
<th>w</th>
<th>$\langle k \rangle$</th>
<th>C</th>
<th>L</th>
<th>$\Phi$</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>9.066</td>
<td>0.090</td>
<td>2.518</td>
<td>0.430</td>
<td>0.832</td>
</tr>
<tr>
<td>0.2</td>
<td>10.105</td>
<td>0.093</td>
<td>2.415</td>
<td>0.369</td>
<td>0.848</td>
</tr>
<tr>
<td>0.3</td>
<td>11.305</td>
<td>0.104</td>
<td>2.314</td>
<td>0.300</td>
<td>0.858</td>
</tr>
<tr>
<td>0.4</td>
<td>11.146</td>
<td>0.112</td>
<td>2.318</td>
<td>0.280</td>
<td>0.854</td>
</tr>
<tr>
<td>0.5</td>
<td>11.511</td>
<td>0.112</td>
<td>2.293</td>
<td>0.269</td>
<td>0.856</td>
</tr>
<tr>
<td>0.6</td>
<td>11.517</td>
<td>0.118</td>
<td>2.303</td>
<td>0.263</td>
<td>0.857</td>
</tr>
<tr>
<td>0.7</td>
<td>11.230</td>
<td>0.110</td>
<td>2.328</td>
<td>0.267</td>
<td>0.854</td>
</tr>
<tr>
<td>0.8</td>
<td>10.153</td>
<td>0.103</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0.9</td>
<td>8.067</td>
<td>0.084</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1.0</td>
<td>4.239</td>
<td>0.444</td>
<td>8.885</td>
<td>0.079</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>Random-based</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>7.630</td>
<td>-</td>
<td>2.726</td>
<td>0.549</td>
<td>0.799</td>
</tr>
<tr>
<td>0.2</td>
<td>8.040</td>
<td>-</td>
<td>2.650</td>
<td>0.488</td>
<td>0.808</td>
</tr>
<tr>
<td>0.3</td>
<td>9.160</td>
<td>-</td>
<td>2.530</td>
<td>0.416</td>
<td>0.829</td>
</tr>
<tr>
<td>0.4</td>
<td>9.533</td>
<td>-</td>
<td>2.478</td>
<td>0.380</td>
<td>0.836</td>
</tr>
<tr>
<td>0.5</td>
<td>10.011</td>
<td>0.105</td>
<td>2.420</td>
<td>0.335</td>
<td>0.841</td>
</tr>
<tr>
<td>0.6</td>
<td>9.804</td>
<td>0.103</td>
<td>2.462</td>
<td>0.355</td>
<td>0.838</td>
</tr>
<tr>
<td>0.7</td>
<td>8.991</td>
<td>0.119</td>
<td>2.530</td>
<td>0.394</td>
<td>0.823</td>
</tr>
<tr>
<td>0.8</td>
<td>10.381</td>
<td>0.099</td>
<td>2.393</td>
<td>0.329</td>
<td>0.848</td>
</tr>
<tr>
<td>0.9</td>
<td>10.693</td>
<td>0.108</td>
<td>2.365</td>
<td>0.301</td>
<td>0.848</td>
</tr>
<tr>
<td>1.0</td>
<td>9.906</td>
<td>-</td>
<td>2.449</td>
<td>0.345</td>
<td>0.837</td>
</tr>
</tbody>
</table>

Table 9.3: Average over 5 runs with $w$ varying from 0.1 to 1.0, for both ring-based and random-based individuals. $\langle k \rangle$ is the mean node degree. $C$ is the clustering coefficient. $L$ is the characteristic path length. $\phi$ is the percentage of shortcuts, and $P$ is the network performance on the density task (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).
### Table 9.4: The ten best evolved networks for the density task using $f'$. $\langle k \rangle$ is the mean node degree. $C$ is the clustering coefficient. $L$ is the characteristic path length. $\phi$ is the percentage of shortcuts, and $P$ is the network performance on the density task. (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).

<table>
<thead>
<tr>
<th>Network Type</th>
<th>$\langle k \rangle$</th>
<th>$C$</th>
<th>$L$</th>
<th>$\phi$</th>
<th>$P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring-based</td>
<td>13.315</td>
<td>0.114</td>
<td>2.162</td>
<td>0.211</td>
<td>0.882</td>
</tr>
<tr>
<td></td>
<td>12.523</td>
<td>0.102</td>
<td>2.230</td>
<td>0.251</td>
<td>0.876</td>
</tr>
<tr>
<td></td>
<td>13.020</td>
<td>0.113</td>
<td>2.203</td>
<td>0.221</td>
<td>0.875</td>
</tr>
<tr>
<td></td>
<td>12.886</td>
<td>0.108</td>
<td>2.209</td>
<td>0.235</td>
<td>0.873</td>
</tr>
<tr>
<td></td>
<td>11.732</td>
<td>0.105</td>
<td>2.291</td>
<td>0.275</td>
<td>0.871</td>
</tr>
<tr>
<td>Random-based</td>
<td>10.483</td>
<td>0.098</td>
<td>2.378</td>
<td>0.348</td>
<td>0.859</td>
</tr>
<tr>
<td></td>
<td>10.671</td>
<td>-</td>
<td>2.358</td>
<td>0.323</td>
<td>0.859</td>
</tr>
<tr>
<td></td>
<td>11.275</td>
<td>-</td>
<td>2.324</td>
<td>0.294</td>
<td>0.856</td>
</tr>
<tr>
<td></td>
<td>10.899</td>
<td>-</td>
<td>2.358</td>
<td>0.310</td>
<td>0.855</td>
</tr>
<tr>
<td></td>
<td>11.597</td>
<td>-</td>
<td>2.284</td>
<td>0.258</td>
<td>0.854</td>
</tr>
</tbody>
</table>
Figure 9.22: Typical behavior of a small-world CA for the density task under probabilistic updating. The density $\rho^0$ 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.
Figure 9.23: Typical behavior of EvCA (Mitchell et al., 1993) under probabilistic updating on the density task. The density \( \rho^0 \) 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.
Part IV

Final Considerations
Chapter 10

Discussion of the Results

Finally I will not become any dumber.
– Epitaph on the gravestone of Pál Erdös

The direct influence of the spatial and temporal dimensions in any dynamical process cannot be neglected nowadays. In biology, as well as in many other areas, researchers are focusing their attention on the spatial structure of the processes under investigation. This knowledge is viewed as necessary in order to understand the dynamical properties of any system. Networks of interacting agents (from chemical to sociological, from animal to computational) are studied and classified in families of graphs according to their statistical properties. Some of these networks are clearly the result of an evolutionary process, in which a kind of darwinian selection has operated. How these networks have emerged is an interesting and most of the time still an open question.

10.1 Part II: Artificial Evolution on Structures

In the second part we have investigated the effect of the spatial structure and temporal dynamics on an artificial evolutionary process. We have derived models for the selection pressure curves induced in the system by different topologies and different generational steps. Our intent not being to perfectly model the dynamical process, we have often used spatial and physical approximations in order to derive recurrence equations that could describe the qualitative different dynamics.
For regular two-dimensional structures, the geometrical approximation permitted us to find that the selection pressure curves must follow a quadratic or sub-quadratic growth and saturation, clearly differentiating this dynamic from that of a regular one-dimensional structure, for which the growth is linear, and from that of a panmictic population, whose growth and saturation follow exponential laws. Also in the case of a randomly structured evolving population, the mean field approximation on the number of neighbors for each individual lets us state that, for large enough link probability in the random graph, the selection pressure is equivalent to that induced by a panmictic population. Small-world and scale-free structures have also been proposed, showing how these topologies have dynamical properties that set them apart and situate them between regular and random structures.

While we believe that we have clearly shown the influence of the spatial and temporal dimension in an artificial evolutionary process, we cannot claim that our investigation has been exhaustive. In fact, on one side some points are still open and need further investigation and, on the other side, the results presented here open new perspectives and directions. In particular, for regular structures we have modeled only the radius 1 von Neumann neighborhood, giving only an idea of the generalisation of the models for larger radii. The use of other neighborhood structures such as Moore’s still has to be studied, and we agree with Manderick and Spiessens (1989) that a geometrical approximation of a sphere should be used in this case. The interesting behaviors showed by small-world and scale-free topologies surely need further investigations: models can be derived for these two families of topologies, and their optimisation as well as their emergence capabilities must still be further explored. In the future, it would also be interesting to investigate Markov chain modeling of these systems and the relationships that may exist with probabilistic particle systems such as voter models Mühlenbein and Höns (2002).

10.2 Part III: Artificial Evolution of Networks

In the third part we have approached the different question of how network structures that show interesting computational capabilities could have emerged. In particular, we
have chosen two prototypical problems, the density classification and the synchronisation tasks for binary automata networks.

Inspired by 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 even-higher performance for both tasks.

The power of this artificial evolution is highlighted by 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 premature 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 be of interest to let the rule evolve simultaneously with the network topology. This has been suggested by Watts (1999) and has previously been attempted by Sipper and Ruppin (1997) with good results. Further work along these lines is needed. The collective computational capabilities of other small-world graph structures, especially scale-free networks, must also be investigated, and we think that a well-designed artificial coevolution of automata rules and network topologies could lead to networks of this family.
Part V

Annexes
In this appendix we show a statistical comparison of the algorithms studied in Chapter 6 by performing \( t \)-tests to the results of all the algorithms. Tables 10.1 to 10.9 are the results of our statistical comparison in terms of the solutions found and the number of generations. No tables are provided for those cases in which the solutions of all the algorithms are equal. On those tables, statistical significance is shown by using symbol ‘+’, and not statistical significance is matched with ‘-’.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Square</th>
<th>Rectangular</th>
<th>Narrow</th>
<th>LS</th>
<th>FRS</th>
<th>NRS</th>
<th>UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>•</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Rectangular</td>
<td>–</td>
<td>•</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Narrow</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>LS</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>•</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>FRS</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>•</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>NRS</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>–</td>
</tr>
<tr>
<td>UC</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>•</td>
</tr>
</tbody>
</table>

Table 10.1: p-values of the avg. fitness for MMDP.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Square</th>
<th>Rectangular</th>
<th>Narrow</th>
<th>LS</th>
<th>FRS</th>
<th>NRS</th>
<th>UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>•</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Rectangular</td>
<td>–</td>
<td>•</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Narrow</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>LS</td>
<td>+</td>
<td>–</td>
<td>•</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>–</td>
</tr>
<tr>
<td>FRS</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>NRS</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>+</td>
</tr>
<tr>
<td>UC</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>•</td>
</tr>
</tbody>
</table>

Table 10.2: p-values of the generations for MMDP.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Square</th>
<th>Rectangular</th>
<th>Narrow</th>
<th>LS</th>
<th>FRS</th>
<th>NRS</th>
<th>UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>⋄</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Rectangular</td>
<td>−</td>
<td>⋄</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Narrow</td>
<td>−</td>
<td>−</td>
<td>⋄</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>LS</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>⋄</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>FRS</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>⋄</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>NRS</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>⋄</td>
<td>−</td>
</tr>
<tr>
<td>UC</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>⋄</td>
</tr>
</tbody>
</table>

Table 10.3: p-values of the avg. fitness for FMS.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Square</th>
<th>Rectangular</th>
<th>Narrow</th>
<th>LS</th>
<th>FRS</th>
<th>NRS</th>
<th>UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>⋄</td>
<td>−</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Rectangular</td>
<td>−</td>
<td>⋄</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Narrow</td>
<td>+</td>
<td>+</td>
<td>⋄</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>LS</td>
<td>−</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>⋄</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>FRS</td>
<td>−</td>
<td>−</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>⋄</td>
<td>−</td>
</tr>
<tr>
<td>NRS</td>
<td>−</td>
<td>−</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>⋄</td>
</tr>
<tr>
<td>UC</td>
<td>−</td>
<td>−</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>⋄</td>
</tr>
</tbody>
</table>

Table 10.4: p-values of the generations for FMS.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Square</th>
<th>Rectangular</th>
<th>Narrow</th>
<th>LS</th>
<th>FRS</th>
<th>NRS</th>
<th>UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>⋄</td>
<td>−</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Rectangular</td>
<td>−</td>
<td>⋄</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Narrow</td>
<td>+</td>
<td>+</td>
<td>⋄</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>LS</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>⋄</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>FRS</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>⋄</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>NRS</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>⋄</td>
<td>−</td>
</tr>
<tr>
<td>UC</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>⋄</td>
</tr>
</tbody>
</table>

Table 10.5: p-values of the generations for P-PEAKS.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Square</th>
<th>Rectangular</th>
<th>Narrow</th>
<th>LS</th>
<th>FRS</th>
<th>NRS</th>
<th>UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>⋄</td>
<td>−</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Rectangular</td>
<td>−</td>
<td>⋄</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Narrow</td>
<td>+</td>
<td>−</td>
<td>⋄</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>LS</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>⋄</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>FRS</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>⋄</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>NRS</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>⋄</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>UC</td>
<td>−</td>
<td>−</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>⋄</td>
</tr>
</tbody>
</table>

Table 10.6: p-values of the avg. fitness for ECC.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Square</th>
<th>Rectangular</th>
<th>Narrow</th>
<th>LS</th>
<th>FRS</th>
<th>NRS</th>
<th>UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>⬤</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Rectangular</td>
<td>–</td>
<td>⬤</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Narrow</td>
<td>+</td>
<td>+</td>
<td>⬤</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>LS</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>FRS</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>⬤</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>NRS</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>⬤</td>
<td>–</td>
</tr>
<tr>
<td>UC</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>⬤</td>
</tr>
</tbody>
</table>

Table 10.7: p-values of the generations for ECC.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Square</th>
<th>Rectangular</th>
<th>Narrow</th>
<th>LS</th>
<th>FRS</th>
<th>NRS</th>
<th>UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>⬤</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Rectangular</td>
<td>–</td>
<td>⬤</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Narrow</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>⬤</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>LS</td>
<td>+</td>
<td>+</td>
<td>⬤</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>FRS</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>⬤</td>
<td>–</td>
<td>⬤</td>
<td>–</td>
</tr>
<tr>
<td>NRS</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>⬤</td>
<td>–</td>
<td>⬤</td>
<td>–</td>
</tr>
<tr>
<td>UC</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>⬤</td>
</tr>
</tbody>
</table>

Table 10.8: p-values of the generations for MAXCUT.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Square</th>
<th>Rectangular</th>
<th>Narrow</th>
<th>LS</th>
<th>FRS</th>
<th>NRS</th>
<th>UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>⬤</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Rectangular</td>
<td>–</td>
<td>⬤</td>
<td>–</td>
<td>–</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Narrow</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>LS</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>FRS</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>NRS</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>UC</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
</tbody>
</table>

Table 10.9: p-values of the generations for MTTP.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Square</th>
<th>Rectangular</th>
<th>Narrow</th>
<th>LS</th>
<th>FRS</th>
<th>NRS</th>
<th>UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>⬤</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Rectangular</td>
<td>–</td>
<td>⬤</td>
<td>–</td>
<td>–</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Narrow</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>LS</td>
<td>–</td>
<td>–</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>FRS</td>
<td>–</td>
<td>–</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>NRS</td>
<td>–</td>
<td>–</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>UC</td>
<td>–</td>
<td>–</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
</tbody>
</table>

Table 10.10: p-values of the avg. fitness for SAT.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Square</th>
<th>Rectangular</th>
<th>Narrow</th>
<th>LS</th>
<th>FRS</th>
<th>NRS</th>
<th>UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>•</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Rectangular</td>
<td>–</td>
<td>•</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>–</td>
</tr>
<tr>
<td>Narrow</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>LS</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>•</td>
<td>–</td>
<td>–</td>
<td>+</td>
</tr>
<tr>
<td>FRS</td>
<td>–</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>NRS</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>–</td>
</tr>
<tr>
<td>UC</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 10.11: p-values of the generations for SAT.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Square</th>
<th>Rectangular</th>
<th>Narrow</th>
<th>LS</th>
<th>FRS</th>
<th>NRS</th>
<th>UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>•</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Rectangular</td>
<td>+</td>
<td>•</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Narrow</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>•</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>LS</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>•</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>FRS</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>NRS</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>–</td>
</tr>
<tr>
<td>UC</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>•</td>
</tr>
</tbody>
</table>

Table 10.12: p-values of the generations for Rastrigin.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Square</th>
<th>Rectangular</th>
<th>Narrow</th>
<th>LS</th>
<th>FRS</th>
<th>NRS</th>
<th>UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>•</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Rectangular</td>
<td>–</td>
<td>•</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Narrow</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>–</td>
</tr>
<tr>
<td>LS</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>•</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>FRS</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>NRS</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>–</td>
</tr>
<tr>
<td>UC</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 10.13: p-values of the generations for Ackley.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Square</th>
<th>Rectangular</th>
<th>Narrow</th>
<th>LS</th>
<th>FRS</th>
<th>NRS</th>
<th>UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>•</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>–</td>
</tr>
<tr>
<td>Rectangular</td>
<td>–</td>
<td>•</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Narrow</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>•</td>
<td>+</td>
<td>+</td>
<td>–</td>
</tr>
<tr>
<td>LS</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>•</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>FRS</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>NRS</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>–</td>
</tr>
<tr>
<td>UC</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 10.14: p-values of the generations for Fractal.
Adresses of the Jury Members

**Président:**
Prof. Raymond Olivier
Decanat Geosciences et de l’Environnement
Université de Lausanne
Collège Propédeutique 1
CH-1015 Dorigny
Suisse

**Codirecteur de Thèse:**
Prof. Marco Tomassini
Institut d’Informatique et Organisation
Université de Lausanne
Collège Propédeutique 1
CH-1015 Dorigny
Suisse

**Codirecteur de Thèse:**
Prof. Andrea Tettamanzi
Dipartimento di Tecnologie dell’Informazione
Università degli Studi di Milano
via Bramante 65
IT-26013 Crema
Italia

**Rapporteur:**
Prof. Vincenzo Piuri
Dipartimento di Tecnologie dell’Informazione
Università degli Studi di Milano
via Bramante 65
IT-26013 Crema
Italia

**Expert:**
Prof. Marc Schoenauer
Equipe TAO - INRIA Futurs
Université Paris Sud
LRI - Bâtiment 490
FR-91405 Orsay Cedex
France

**Expert:**
Prof. Moshe Sipper
Department of Computer Science
Ben-Gurion University
P.O. Box 653
IL-84105 Beer-Sheva
Israel
List of Publications


List of Attended Conferences


Journées Evolutionnaires Trimestrielles, JET 14, 1 April 2005, Lausanne, Switzerland. Local Organizer.


Turing Day, 28 June 2002, Lausanne, Switzerland.


References


Bremermann, H., 1958. The evolution of intelligence, the nervous system as a model of its environment. Tech. Rep. No. 1, Contract No. 477(17), Department of Mathematics, University of Washington, Seattle, WA.


REFERENCES


Hopcroft, J. E., Ullman, J. D., 1979. Introduction to Automata Theory Languages and Computation. Addison-Wesley, Redwood City, CA.


REFERENCES


REFERENCES


