Comparative study of effects of fitness landscape changes in open-ended evolutionary simulations and in Genetic Programming.

David Medernach

BDS Group

Department of Computer Science & Information Systems

Faculty of Science and Engineering

University of Limerick

Submitted to the University of Limerick for the degree of

PhilosophiæDoctor (PhD) 2017

Supervisor: Prof. Conor Ryan
BDS Group
University *of* Limerick
Ireland

Co-Supervisor: Dr. Jeannie Fitzgerald
BDS Group
University *of* Limerick
Ireland

Examiner: Dr. Malachy Eaton
Department of Computer Science & Information Systems
University *of* Limerick
Ireland

External Examiner: Prof. Charles Ofria
Digital Evolution Laboratory
Michigan State University
United States

Chair: Prof. Ita Richardson
Lero
University of Limerick
Ireland

Day of the defense: 24 March 2017

Signature from head of PhD committee:

Abstract

The biological world where natural selection takes place is a world constantly affected by external physical phenomena, whether cyclical and regular, such as the rotation of the earth, or punctual such as when a meteorite strikes the earth. It is recognized that these phenomena affect the evolution of life, but their interaction with natural selection have not yet been fully explored.

This thesis studies the effects of environmental fluctuations via evolutionary simulations. In particular, we propose to study them through simulations of Genetic Programming as well as Artificial Life evolving virtual ecosystems.

We first present the history of natural selection as well as artificial life and genetic programming, focusing on the role of environmental fluctuations in these three fields of research.

We then examine the effects of such fluctuations on virtual ecosystems. We create a new virtual ecosystem which we call HetCA, that is based on cellular automata with heterogeneous transition rules. In this simulation, we test effects of these fluctuations on the evolutionary progress as well as on the level of selection and show that the type of fluctuation determine the level of selection.

Finally, we study the effects of such fluctuations in Genetic Programming, firstly through an extension of Random Interleaved Sampling and then by creating a new method of Genetic Programming which we call Wave. We note that, on the studied problems, Wave is a very competitive method compared to a selection of benchmarks including *non-evolutionary computation based* optimization methods.

Declaration

I herewith declare that I have produced this thesis without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This thesis has not previously been presented in identical or similar form to any other Irish or foreign examination board.

The thesis work was conducted from 2012 to 2017 under the supervision of Prof. Conor Ryan and Dr. Jeannie Fitzgerald at University of Limerick.

Limerick, 2017

David Medernach

Acknowledgements

To begin my thanks to my supervisor **Prof. Conor Ryan** for the opportunity he offered me and the support he gave me throughout my PhD. Thanks also to him for being an example of an open researcher, generous and interested in a multitude of subjects constantly renewed.

Thanks to **Dr. Jeannie Fitzerald** for her support during my Ph.D. Thanks to her for our warm discussions and of course the corrections made. Special thanks to her for not losing patience with my countless mistakes in English.

Thanks to **Prof. René Doursat** and **Dr. Taras Kowaliw** for introducing me to the world of scientific research and for their wise advices.

Thanks to Atif, Fergal and Gopinath with whom I had the opportunity to go to many conferences. Special thanks to **Dr. Atif Azad** who was the co-author of several of the articles in which I participated.

A big thank you to all the administrative team of the University of Limerick and especially to **Nuala Kitson** who has very regularly assisted me in administrative procedures and has recently retired.

My thanks to Alex without whom I would not have embarked on this journey, to Caroline, Pierre, Adrien and Victor for their unconditional support unceasingly renewed, thank you to Dan to whom I apologize for my perpetual unavailability, to the Victors for their visits to Limerick, as well as to the “fourdragz” (Especially Simon) for the incessant discussions and maintaining LOPHISS social life abroad.

Contents

List of Tables

List of Figures

LIST OF FIGURES

List of Publications

Publications

Most of this research has appeared in various peer-reviewed articles as listed below.

1. David Medernach, Taras Kowaliw, Conor Ryan, and René Doursat. *Long-term evolutionary dynamics in heterogeneous cellular automata.* In Proceedings of the 15th annual conference on Genetic and evolutionary computation, pages 231-238. ACM, 2013.

2. R. Muhammad Atif Azad, David Medernach, and Conor Ryan. *Efficient Approaches to Interleaved Sampling of Training Data for Symbolic Regression.* In Nature and Biologically Inspired Computing (NaBIC), 2014 Sixth World Congress on, pages 176-183. IEEE, 2014.

3. R. Muhammad Atif Azad, David Medernach, and Conor Ryan. *Efficient Interleaved Sampling of Training Data in Genetic Programming.* In Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation, pages 127-128. ACM, 2014.

4. David Medernach, Jeannie Fitzgerald, Simon Carrignon, and Conor Ryan. *Evolutionary Progress in Heterogenous Cellular Automata (HetCA).* In Proceedings of the Companion Publication of the 2015 European Conference on Artificial Life, pages 512-519. MIT Press, 2015.

5. David Medernach, Jeannie Fitzgerald, R. Muhammad Atif Azad, and Conor Ryan. *Wave: A Genetic Programming Approach to Divide and Conquer.* In Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, pages 1435-1436. ACM, 2015.

6. David Medernach, Jeannie Fitzgerald, R. Muhammad Atif Azad, and Conor Ryan. *Wave: Incremental Erosion of Residual Error.* In Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, pages 1285-1292. ACM, 2015.

7. David Medernach, Jeannie Fitzgerald, R. Muhammad Atif Azad, and Conor Ryan. *A New Wave: A Dynamic Approach to Genetic Programming.* In Proceedings the Companion Publication of the 2016 on Genetic and Evolutionary Computation Conference, pages 757-764. ACM, 2016.

8. David Medernach, Simon Carrignon, René Doursat, Taras Kowaliw, Jeannie Fitzgerald, and Conor Ryan. *Evolution of Heterogeneous Cellular Automata in Fluctuating Environments.* In Proceedings the Companion Publication of the The 15th International Conference on the Synthesis and Simulation of Living Systems, pages 216-223. MIT Press, 2016.

LIST OF PUBLICATIONS

Part I

Introduction and Background

1

Introduction

Charles Darwin introduced the theory of evolution by natural selection in [Darwin and Wallace, 1858], and since that time the concept has remained unchanged at the highest level. However, there have been numerous and animated debates about its concrete implementation. Evolutionary Computation (EC) has taken inspiration from these debates, for example recently by reusing ideas from epigenetics [Tanev and Yuta, 2003]. We propose here to focus on a point which, although not central to the theory itself, comes up regularly in the polemics that have marked the field of evolutionary biology: environmental fluctuations. That is to say events fluctuating randomly or regularly and modifying the optimal strategy maximizing an individual's fitness.

This thesis studies the effects of environmental fluctuations on natural selection in the context of computer simulations such as Genetic Programming (GP) and "open-ended" artificial life simulations.

The remainder of this chapter is organised as follows: Section 1.1 describes the motivation behind this research; Section 1.2 presents the research questions and the objectives addressed in the thesis; then Section 1.3 lists the contributions; finally Section 1.4 presents the structure of the thesis.

1.1 Motivation

Biology, particularly evolution and natural selection have inspired and given birth to several fields of research in computer science. These areas of research some-

time seek to use principles discovered in living beings to optimize solutions or to solve problems (GP, Neural Networks), sometimes the aim of research is to study the mechanisms of functioning of living beings from new, potentially interesting, angles (Artificial life, Bioinformatics), and frequently these objectives are combined in the field of EC. Here we are interested in a constraint that underpins biological evolution: environmental variation. We will study its effects in the framework of an artificial life simulation and to take inspiration from them to propose improvements in GP.

Artificial life (Alife) is an interdisciplinary field of research named in the 1980s by Christopher Langton [Langton et al., 1989]. Its purpose is to recreate some aspect of life through modeling, computer simulation, robotics, or biochemistry. Alife is also sometimes described as the study of not only "life-as-we-know-it" but also "life-as-it-could-be" [Shanken, 1998]. While the term "artificial life" first appeared in the 1980s, as we shall see in Chapter 2, it is possible to trace the emergence of the relevant concepts back to the works of John von Neumann in the 1950s. However, Alife, as we know it today, has been maturing since the late 1980s. Along the way, there have been several key developments in the field, including the use of virtual evolving ecosystems such as [Ray, 1992] or Avida [Adami and Brown, 1994]. These simulations are used as tools/systems for simulating a form of open-ended evolution even if they do not necessarily satisfy long-term evolutionary dynamics such as observed in the fossil record [Maley, 1999].

GP is a technique, first proposed in [Koza, 1994], inspired by natural selection that allows a program, by an evolutionary algorithm, to gradually optimize a population of programs to increase their capacity to perform a task. In GP and other EC methods evolution tends to create individuals closer and closer to global or local optima. One problem that these algorithms face is premature convergence when most individuals have reached the same local optima and evolution ceases to produce fitter individuals.

A similar phenomenon happens also in the biological world. The theory of Punctuated Equilibrium [Gould, 1972] suggested by Stephen Jay Gould proposes that phenotypic evolutionary changes of a species are not uniform, but rather a succession of stasis and rapid changes (characterized for example by speciation).

The causes of those rapid evolutionary changes remains uncertain but could include: *saltationism*[1] [Blackburn, 1995]; *peripatric speciation*[2] [Mayr, 1982]; or[3], more simply, ecological changes [Milligan, 1986]. Indeed, the fitness landscape is dynamic (Environmental changes such as ecological changes or the competition with a new species produce variations). Environmental changes and fluctuations have been, in the course of the history of the theory of natural selection related to many phenomena such as levels of selections but also development or "niche construction".

In the context of EC, we hypothesize that fluctuations of environment may not only be a means of improving evolutionary simulations, but also of studying their effects.

1.2 Research Aims and Objectives

We wish to verify that the capacity of natural selection to solve very complex problems in the biological world[4] is facilitated by the presence of enviromental fluctuations and also wish to verify that we can derive from it in EC.

1.2.1 Research Questions

Environmental fluctuations, if they are not central in themselves in evolutionary biology, are suspected of being the cause of phenomena which have generated important polemics in biology punctuated equilibrium [Gould and Eldredge, 1977]

[1] This theory consists of important mutations that will move an individual far away from its parents on the fitness landscape. This theory is not without controversy [Mayr, 1942], since modern evolutionary, saltationism it is often considered to be fundamentally incompatible with phyletic gradualism (which theorizes that evolution is uniform and gradual) and therefore is controversial, nevertheless more recent works [Pigliucci, 2007] shows that it plays a role in evolution.

[2] A small modification of Ernst Mayr's allopatric speciation [Jepson et al., 1949] (Speciation caused by the separation of the initial population in two sub-population that are prevented to mate together).

[3] Those potential causes are not necessarily mutually exclusive

[4] Via phenomena such as exaptation, multilevel selection...

or levels of selection [Jablonka et al., 1995]. This thesis aims to reconcile evolutionary biology and evolutionary programming by integrating and studying the potential benefits of fitness landscape fluctuations in the context of computer simulations such as GP and open-ended Alife simulations.

Research Question 1: What is the state of the art in terms of environmental fluctuations in evolutionary biology and EC?

Research Question 2: Can random fluctuations in the fitness landscape reduce overfitting?

Research Question 3: Is it possible and useful to solve a fixed and unique problem by evolutionary programming while using several different fitness functions?

Research Question 4: Does the introduction of fluctuations in EC promote mechanisms analogous to those that they are capable of generating in evolutionary biology?

Research Question 5: Can we observe evolutionary progress in Alife simulations using no explicit fitness function?

Research Question 6: How does an open-ended simulation react to the introduction of environmental fluctuations?

Research Question 7: Does the type (frequency, strength) of environmental fluctuations have an influence on the level of selection?

1.2.2 Objectives

Those following objectives address the aims and research questions.

1. Conduct a survey into the state-of-the-art evolutionary computing and more particularly in GP.

2. Summarize the role played by environmental fluctuations in the development of evolutionary biology.
3. Propose a brief summary of the issues and the history of Alife.
4. Summarize the existing work in terms of evolutionary cellular automata.
5. Creation of an Alife simulation capable of exploring the importance of environmental fluctuations.
6. Test the benefits of changes in the fitness function in evolutionary algorithms such as GP.
7. Specify the notion of evolutionary progress and test it in the context of a simulation that does not use an explicit fitness function.
8. Take inspiration from environmental change in biology to build a GP system solving a problem in successive steps using different fitness functions.
9. Test the effect of environmental fluctuations on selection levels in an open-ended simulation.
10. Extend the Random interleaved sampling to test other forms of interleaved.

1.3 Research Contributions

Scientific publications that emerged from contributions of this thesis are enumerated on page 2. Following are the research contributions of this work in their order of appearance.

Survey of Evolutionary Biology: An abridged history of evolutionary biology by focusing on the role that played the fluctuations of fitness landcscape, which accomplishes Objective 1, and is discussed in Chapter 2.

Survey of EC: An abridged review of methods used in EC focusing GP and Alife, which realizes Objective 2 and 3, is discussed in Chapter 2.

Brief Description on Cellular Automata: A brief description of the principle of cellular automata as well as their use in particular in the context of evolutionary simulations, which fulfils Objective 4, is presented in Chapter 2.

Creation of HetCA, an "open-ended" Alife simulation: An evolutionary simulation where there are forms of evolutionary progress without any explicit fitness function and where the optimal survival strategy is not fixed, which performs Objective 5 and 8, is proposed in Chapter 3.

Extension of interleaved GP: A comparison of the classic one point interleaved sampling with two new types of interleaved GP: random interleaved and size reduction interleaving, which achieves Objective 6 and 10, is performed in Chapter 4.

Creation of Wave: A system of GP solving a problem by a succession of cycles optimizing the residual errors of the previous cycles, which realizes objective 7 is proposed in Chapter 5.

Study of the effects of cyclical environmental fluctuations in HetCA: cyclic and external environmental fluctuations are introduced into HetCA, their effects on the level of selection are studied which fulfils Objective 10 is proposed in Chapter 6.

Extension of Wave: A faster and more biologically realistic version of wave is proposed, which accomplishes objective 7 is presented in Chapter 7.

A glossary of evolutionary biology: A glossary of the specific vocabulary used in evolutionary biology is given in Appendix A.

Very long term HetCA simulations examples: The visuals of different very long term HetCA simulations are presented in the Appendix B.

1.4 Thesis Structure

The research carried out in this thesis is organized in three parts.

Part I: Introduction and Background

The first part of the thesis consists of two chapters which introduce and contextualize the research topic.

Chapter 1 briefly introduces the importance of environmental fluctuations in natural selection. This chapter outlines the aims, research questions, objectives and the research contributions of this thesis.

Chapter 2 this first describes the history of natural selection and in particular the links between environmental fluctuations and some of the controversies that have crossed this field of research. In a second step this chapter describes the different fields of research reusing natural selection in computer science and especially GP and Alife.

Part II: Experiments of environmental fluctuations in computer simulations of evolutionary programming and Alife.

The second part of the thesis consists of five chapters which describe the simulations carried out within the framework of this thesis.

Chapter 3 introduces a new Alife simulation: HetCA. For this experiment a cellular automata using local transition rules is used. Several methods to test the existence of an evolutionary progress are then applied to check that HetCA generates a form of open-ended evolution.

Chapter 4 examines the intrinsic interest of interleaving of different fitness functions between each generation as part of GP. For this, three forms of interleaving are used: one data point interleaving, size reduction interleaving and random interleaving.

Chapter 5 presents Wave, a new method of GP that decomposes a GP simulation into several sequential run subunits using different fitness functions that optimize the residual error of previous runs.

Chapter 6 extends hetCA by introducing cyclical environmental fluctuations. The effects of these fluctuations on the level of selection are then studied through the existence of a genotypic or phenotypic selection.

Chapter 7 extends Wave using a more biologically realistic model. The populations of individuals are no longer systematically renewed. Furthermore, the detection of a stasis period is improved by the use of a validation dataset.

Part III: Conclusions and Future Directions

The third part concludes and recommends future research extensions.

Chapter 8 summarizes the contributions of this thesis and suggests potential future directions.

2

Background

In this chapter we examine the topic of biological and artificial evolution from two perspectives. First, in Section 2.1, we explore some of the most important contributions in the field, from those transformist approaches which preceeded the concept of natural selection developed by Darwin, to much more recent theories such as those concerning multilevel selection. Secondly, in Section 2.2, we discuss those Evolutionary Computation approaches relevant to this thesis, such as GP and Alife, which have been inspired by ideas from evolutionary biology.

2.1 Evolutionary Biology

In this section we provide an overview of the history and mechanisms of *evolutionary biology*. Evolutionary biology is a subfield of biology that focuses on the evolutionary mechanisms that produce the diversity of life we see all around us. By *functional biology* we mean the study of the proximal causes of functional mechanisms in living organisims. We will limit this discussion to the theoretical aspects of evolutionary biology and will only superficially discuss functional biology. We are interested here in evolution *as it is possible* and not necessarily *as we know it* for living things. Therefore, for example, we do not discuss genetics, though it is fundamental to the understanding of evolution of living beings.

2.1.1 From *Fixism* toward *Transmutation of species*

Transmutation of species is the theory that species have changed over time, where some have become extinct while other new species have emerged. This term was first used in [Lamarck, 1809] and is generally opposed to *fixism*, the assumption that all species of living things have always existed, which prevailed earlier.

However, half a century earlier, the classic on the donkey in [*L'Histoire Naturelle, générale et particulière, avec la description du Cabinet du Roi Tome IV*] [de Buffon, 1753], of the French naturalist Georges-Louis Leclerc de Buffon, is often taken as the first *transmutation of species* theory. Buffon develops there the idea that all animals, including humans, may descend from the same animal:

> *"Si l'on admet une fois qu'il y ait des familles dans les plantes et dans les animaux, que l'âne soit de la famille du cheval, et qu'il n'en diffère que parce qu'il a dégénéré, on pourra dire également que le singe est de la famille de l'homme, que c'est un homme dégénéré, que l'homme et le singe ont eu une origine commune comme le cheval et l'âne, que chaque famille, tant dans les animaux que dans les végétaux, n'a eu qu'une seule souche, et même que tous les animaux sont venus d'un seul animal, qui, dans la succession des temps, a produit, en se perfectionant et en dégénérant, toutes les races des autres animaux."*

> "If we admit once that there are families in plants and in animals, that the donkey is from horse family, and that it differs only because it has degenerated, we can also say that the ape is from the human family, that it is a degenerate human, that human and ape had a common origin as the horse and the donkey, that every family in both animals and plants, had only one strain, and even that all animals came from a single animal, which in the course of time, produced by refinement and degeneration of all races other animals."

This is certainly the first time that such ideas were presented so clearly to a broad audience, but the paternity of the idea remains unclear. Buffon claims

to refute this theory and attributes it to Carl Linnaeus, possibly to avoid religious censorship and eventually to cause problems for a competitor in biological taxonomy and species description. Linnaeus was fixist and nothing in his work confirmed the accusation of Buffon but, as explained in [Ostoya, 1954] Pierre Louis Moreau de Maupertuis, had presented very similar ideas a few years earlier in [de Maupertuis, 1751] and later in [de Maupertuis and Trublet, 1754]. Buffon was familiar with the work of Maupertuis of whom he spoke well in public, but he never attributed to him the authorship of this theory, perhaps because Maupertuis was primarily known for his work in mathematics and because his reputation was in no way comparable to that of Buffon.

Figure 2.1: Colorized version of the engraving illustrating the text on the donkey in Volume IV of The Histoire Naturelle, générale et particulière, avec la description du Cabinet du Roi where Georges-Louis Leclerc offers a *transmutation of species* theory.

But Buffon did not propose any mechanism to explain the transmutation of species. This only happened half century later, in [Lamarck, 1809], when Jean-Baptiste Lamarck sought to explain the gradual increase in the complexity and diversity he had observed in the study of fossils, through two concepts widely accepted[1] at the time: the *spontaneous generation*, the idea that simple living organisms appear spontaneously from inanimate matter, and what we now call the *inheritance of acquired characteristics*, the ability for living beings to transmit to their offspring characteristics acquired during their lives.

[1] As explain e.g. in [Bonduriansky, 2012] and contrary to popular belief Lamarck is not at the origin of the concept of inheritance of acquired characteristics.

Spontaneous generation gradually came out of the scientific consensus, *inter alia* thanks to Pasteur's experiments; even if we know know that there is critical methodology problems in Pasteur's experiment as explained in [Latour, 1989].

2.1.2 Natural Selection

Natural selection was presented for the first time in [Darwin and Wallace, 1858], and then further developed in [Darwin, 1859]. Darwin did not regard natural selection as an alternative to the *inheritance of acquired characteristics* but as a complementary mechanism. He even developed, in [Darwin, 1868], a theory of heredity: pangenesis, to explain *inheritance of acquired characteristics*. The theory of *inheritance of acquired characteristics* gradually disappeared from the scientific consensus in the beginning of the twentieth century[1], as the progress of *functional biology*, especially genetics, enabled a better understanding of the mechanisms of heredity.

However, if the laws of heredity were proposed in [Mendel, 1865] shortly after the formulation of natural selection by Darwin; and even if, as explained in [de Beer, 1964] Mendel quickly understood the relevence of these laws to test the hypotheses of Darwin and Lamarck; these rules were only rediscovered much later and subsequently brought together with natural selection in [Fisher, 1930] in what Julian Huxley called *The Modern Synthesis* in [Huxley, 1944].

Despite the *Modern Synthesis*, the exact definition of natural selection remains a debate among biologists and biology philosophers. As explain in [Godfrey-Smith, 2007], the most commonly cited summary of Evolution by Natural Selection was given by Richard Lewontin. Lewontin gave his last definition of natural selection in [Levins and Lewontin, 1985] and claimed a sufficient mecanism for evolution by natural selection is contained in three, necessary and sufficient, propositions:

1. There is variation in morphological, physiological, and behavioral traits among members of a species (the principle of variation).

[1]Except in the USSR where it remained the consensus of Soviet biology until the fall of Lysenko and Khrushchev in 1964.

2. The variation is in part heritable, so that individuals resemble their relations more than they resemble unrelated individuals and, in particular, offspring resemble their parents (the principle of heredity).

3. Different variants leave different numbers of offspring either in immediate or remote generations (the principle of differential fitness).

2.1.2.1 Variation in heritable traits among members of a species

In the the modern synthesis view of evolution random variations occur in two types of events, *mutations* and *chromosomal crossovers*.

A mutation is in biology the alteration of the nucleotide sequence of the genome of an organism. They result from errors during DNA replication and other types of damage to DNA. Since the emergence of molecular genetics they are considered to be the main biological mechanism implementing the "variation in heritable parts" described in natural selection.

Crossovers[1] are, in eukaryotes, a mechanism of exchange of genetic material between chromosomes. They are sources of variations, recombinations and gene duplication in biological organisms. They have probably been initially selected as a DNA repair mechanism and / or as a means to maintain genetic diversity. Nevertheless sexual reproduction and therefore crossover, is not essential to evolution by natural selection. In [Crow and Kimura, 1965] Crow contends that "The advantage of a reproductive system that permits free recombination (sexual reproduction) is greatest for the incorporation of mutant gene with individually small effects, occurring at relatively high rates, and in large populations."[2]

2.1.3 Levels of Selection

It may seem obvious that it is individuals that are selected by natural selection, yet this is not anything trivial and numerous debates around this topic have punctuated the history of evolutionary biology. During these discussions the gene, the cell, the group, and species, among others, were considered units of selection.

[1] Horizontal gene transfer also exist in other organisms.
[2] Crow considers 1000 individuals a small population

In this section we are particularly interested in the selection at the group of individuals level, selection at the gene level and the latest theories envisaging multi-level selections.

2.1.3.1 *Group Selection* or *Kin Selection*?

Another issue that drives research in evolutionary biology is the question of altruism, how behaviours that appear contrary to the evolutionary interest of individuals could be selected by natural selection? An example of such altruistic behaviour might be screaming an alarm to alert the group of the presence of a predator when doing so would incur the risk of attracting the predator to oneself. *Group selection*, the idea that natural selection might operate not only at the level of individuals but also between groups of individuals of the same species, was an early theory.

Indeed, as explained in [Ruse, 1980], the two co-discoverers of natural selection were already debating the issue, and while Darwin did not believe in group selection[1] it found a defender in Alfred Russel Wallace. Yet *group selection* was not generally recognized until much later through its mathematization in [Price, 1972], and is still debated nowadays for example in [Abbot et al., 2011] or in [Queller et al., 2015]. If *group selection* has been debated in the scientific community for so long, and is still disputed, it is because even if a group can have a selective advantage over competing groups thanks to the presence of altruistic individuals within this group, this does not prevent selfish individuals to have a selective advantage over the altruistic individuals within the group.

Usually *group selection* is opposed to an alternative explanation of altruism: *kin selection*, which was generalized and extended in [Hamilton, 1964] as *inclusive fitness*. Kin selection, to explain altruism, develops the idea that increasing the chances of reproduction of related individuals, even at the cost of reducing its own chances of reproduction, is a valid evolutionary strategy. Hamilton's theories lead to the gene and not the individual to be considered as the selection unit of natural selection[2].

[1] Or only for mankind in [Darwin, 1888].

[2] This reductionism to the gene view of evolution is illustrated among others by the thought experiment called the "green-bear effect".

The most complete version of the gene reductionism of natural selection was certainly presented in [Dawkins et al., 1989a], where through the search of a *replicator*[1] on which applies natural selection, R. Dawkins develops the idea that the gene is the only selection unit. It is nevertheless important to note that Dawkins uses the term *cistron* to describe what is commonly called a gene[2], and uses a different definition of the gene:

> "In the title of this book the word gene means not a single cistron but something more subtle. My definition will not be to everyone's taste, but there is no universally agreed definition of a gene. Even if there were, there is nothing sacred about definitions. We can define a word how we like for our own purposes, provided we do so clearly and unambiguously. The definition I want to use comes from G. C. Williams.* A gene is defined as any portion of chromosomal material that potentially lasts for enough generations to serve as a unit of natural selection. In the words of the previous chapter, a gene is a replicator with high copying-fidelity. Copying-fidelity is another way of saying longevity-in-the-form-of-copies and I shall abbreviate this simply to longevity." In [Dawkins et al., 1989b].

In this book, Dawkins also considers the existence of another unit of selection completely irreducible to the gene on which natural selection could also be applied: the *meme*, cultural replicators, copied by imitation. Memes can be ideas, behaviors or any other imitable phenomena. Dawkins uses it to explain things such as the propagation of religions. The concept was further developed among others by S. Blackmore who postulates that they could be the cause of the increase in brain mass in humans in [Blackmore, 2000]. But is still highly controversial and probably suffers from a problem of definition as well as the lack of an identified physical vessel.

[1] Dawkins uses and introduces this very general concept, a replicator is "anything of which copies are made", it is important to note that the copy notion here is a complete copy of the replicator, the inheritance of acquired characters exist here and it is for this reason that an individual is not for dawkins, a replicator.

[2] A fragment, or locus of a DNA sequence which parameter the synthesis of a given RNA.

The polemics about the Dawkins book were virulent and multiple, the most famous example is probably the controversy with Stephen Jay Gould, which is described as follows in [Jablonka et al., 2005b]:

> The controversy between Gould and Dawkins continued until Gould's death in 2002. Like many of the controversies that punctuated the earlier history of evolutionary thinking, it was bitter, venomous, and often unfair. Arguments were pushed ad absurdum, and the ambiguities of language were used and misused to erect and demolish straw men.

Although Dawkins tried to address these criticisms in [Dawkins, 1982] which he considers to be his most important work, intransigent opponents and radical partisans[1] to group selection can still be found today. However many biologists also claim that the two visions are not mutually exclusive, and introduce some multilevel selection.

> "Perhaps the best outcome for the group selection debate is the one that appears to be occurring at present. Although there remain many within the biological community who maintain extreme positions, more and more biologists in disparate areas of research are embracing a hierarchical approach to evolutionary theory. In this context, the diametric contest between the group or the individual as the unit of selection is subsumed into a hierarchy that acknowledges the action of

[1]

> "What Dawkins fails to realise is that his argument in effect invokes group selection! From the selective point of view, genes sacrificing their independence by combining to form higher-level functionalunits, e.g. chromosomes or cells, is strictly analogous to individuals combining themselves into higher-level functional units, e.g. groups. But Dawkins is an implacable opponent of group selection, insisting on the impotence of selection for group advantage as an evolutionary mecha-nism, compared with ordinary individual selection! Clearly, Dawkins has failed to realise that trying to explain the major transitions involves us inlevels of selection issues closely analogous to those on which debate traditionally focused." in [Okasha, 2006b]

> selection at various levels, from the gene to the cell to the individual to the group to the species to the clade." in [Borrello, 2005].

2.1.3.2 Multilevel Selection?

With new advances in functional biology, questions about the selection levels have gradually moved to the *epigenetic* level. Epigenesis intersects a set of molecular mechanisms that modulate gene expression such as for example the methylation of DNA. These mechanisms, very important in the development of living organisms, also allow a form of inheritance which is not directly encoded in the DNA. Epigenetic changes triggered by environmental stimuli can be transmitted to offspring. Many researchers saw this as a return of the inheritance of acquired characters typically amalgamated with the theories of Lamarck, as for example in [David et al., 2009] or in [Jablonka and Lamb, 2002].

Given the definition of the gene used by Dawkins and his generalization by the usage of the term replicator, one may think that a support of heredity not only based on the DNA won't necessarily be a problem for his vision of natural selection focused on genes. And indeed, it is confirmed in [Dawkins, 2004] where Dawkins answers criticisms formulated by Kevin Laland, J. Scott Turner and Eva Jablonka.

> "I agree, insofar as the gene role in Darwinian models does not have to be played by DNA. If I am a triumphalist, it is a replicator triumphalist."

In this text Dawkins explains that he is open to a redefinition of his genetic replicator as well as to the existence of other replicators but he is much more hostile to the existence of *inheritance of acquired characteristics*.

> "The point is that some genetic changes are passed on (otherwise there could be no evolution) but no environmentally acquired changes are passed on (at least not with enough high fidelity to have a chance of surviving into the indefinite future). Or, if they are passed on, they are replicators by definition..."

For their part [Jablonka et al., 2005c] consider the existence of 4 distinct levels of selections: Genetic, Epigenetic, Behavioral, and Symbolic Variations:

1. The Genetic level is close to the views of Hamilton and Dawkins but using a DNA based definition of the gene more or less equivalent to what Dawkins call a cistron.

2. The Epigenetic level is a collection of potential mechanisms of replication such as prions, DNA methylation etc.

3. The Behavioral level is also a collection of potential mechanisms of replication such as inheritance through the transfer of behavior-influencing substance; inheritance through non-imitative social learning; imitative learning; or traditions.

4. The Symbolic Variations focus on symbolic cultural transmission. This level is conceptually similar to Dawkins' memes[1].

Different properties of these inheritance systems regarding the transmission of information are collated in Table 2.1. Jablonka and Lamb insist that there are fundamental differences between those four systems and therefore that the four levels of selection are more than different approaches to the same phenomenon, developing the idea through some thought experiments, that evolution by natural selection could continue even with living organisms whose DNA does not undergo any changes.

[1]There is also a debate on the specificity of culture, while Blackmore made memes a uniquely human phenomenon enabled by our ability to be generalists imitators; Dawkins believes that this also marginally concerns animals taking the songs of saddlebacks as an example of non-human cultural transmission; Finally we find works such [Jayat, 2010], which do not consider the issue under the memes perspective but develop the idea that animal cultures relates quite a number of species such as cetaceans, monkeys, songbirds... [Jablonka et al., 2005c] distinguish imitative learning and traditions both not limited to humans and concerning behavioral inheritance systems from symbolic inheritance systems.

Table 2.1: The re-production of information [Jablonka et al., 2005a]

Inheritance system	Organization of the information	Dedicated copying system?	Transmits latent (non expressed) information?	Direction of transmission	Range of variation
Genetic	Modular	Yes	Yes	Mostly vertical	Unlimited
Epigenetic					
Self-sustaining loops	Holistic	No	No	Mostly vertical	Limitated at the loop level, unlimitated at the cell level
Structural templating	Holistic	No	No	Mostly vertical	Limitated at the structure level, unlimitated at the cell level
RNA silencing	Holistic	Yes	Sometimes	Vertical sometimes horizontal	Limitated at the single transcript level, unlimitated at the cell level
Chromatins marks	Modular and Holistic	Yes (for methylation)	Sometimes	Vertical	Unlimitated
Organism-level developmental legacies	Holistic	No	No	Mostly vertical	Limitated
Behavioral					
Behavior-affecting substances	Holistic	No	No	Both vertical and horizontal	Limitated at the single behavior level, unlimitated for lifestyles
Nonimitative social learning	Holistic	No	No	Both vertical and horizontal	Limitated at the single behavior level, unlimitated for lifestyles
Imitation	Modular	Probably	No	Both vertical and horizontal	Unlimitated
Symbolic	Modular and holistic	Yes, several	Yes	Both vertical and horizontal	Unlimitated

2.1.4 Environmental Fluctuations

Living beings do not evolve in a stable and homogeneous environment. Environmental changes, whether spatial or temporal, may affect their rate of reproduction and survival, thus such fluctuations play an evolutionary role. We chose here to focus only on temporal environmental changes[1]. In natural evolution, environmental changes may include cyclic events such as seasonal changes or the daily alternation of light and darkness, occasional changes such as the appearance of new predators or the potential for new food sources, or more radical modifications such as environmental stresses induced by climate transitions.

One of the first research on environmental changes was proposed in [Levins, 1968]. Richard Levins develops the idea that in an environment with cyclic variation, where different optimal strategies exist at each phase of the cycle, the optimum overall strategy is not necessarily an intermediate strategy. The ratio between the pace of environmental change and the pace of reproductive cycles can then be determinant on the ideal strategy.

> "There is always the danger that by the time the adaptation has taken place the environment has changed. This danger is of course greater when the delay is longer, and is especially so when the population adapts from generation to generation by genetic change through response to natural selection"

He then proposes *development*, i.e. the ability of a single genotype to reach several different phenotypes depending on the environmental conditions, as an answer to this problem, and identifies four ways for natural selection to respond to a variable environment. It may produce:

1. a unique phenotype regardless of the environment;

2. a phenotype that varies as a continuous function of the environment variable;

[1] We chose this form of fluctuations among other things for their potential effects on the selection levels.

3. two different phenotypes, one appearing for environment below a certain threshold while the other appears above the threshold;

4. different phenotypes – the probability of each depending on the environement.

In this book Richard Levins also develops the idea that "The sensitivity of the gene frequency to the environment depends on the shape of the fitness set" and that "phyletic lines which have passed through many different kinds of environments on a geological time scale [...] will therefore be more able to undergo major restructuring" (evolvability).

Over the following decades the effects of environmental changes on natural selection were studied under different perspective such as fecundity [Schaffer, 1974], life expectancy [Leigh, 1981] or climate change [Gross, 1991] probably with the entry of this topic in the public debate in the early 90s.

The issue of environmental variation also appears briefly in the controversy over the punctuated equilibrium proposed in [Gould and Eldredge, 1977]. In an attack against *phyletic gradualism*, which theorizes that most speciations are slow uniform and gradual, Stephen Jay Gould and Niles Eldredge argue that the study of fossils shows that evolution is not linear but a succession of stasis and rapid rare significant evolutionary changes. This concept was very controversial, e.g. Richard Dawkins defended in [Dawkins et al., 1989a] that gradualism as presented was only a strawman, and is still debated today, e.g. [Pennell et al., 2014] proposed that it actually consists of four distinct issues[1]. Regarding environmental changes in their paper on punctuated equilibrium, the authors introduce the idea that environemental change could lead to change in the spatial distributions of phenotypes and thus phenotypic changes without genotypic changes.

> "[...] why must we invoke genetic change mediated by natural selection [...]. For basic dimensions of simple creatures, a purely phenotypic response of an unaltered genotype to changing environments seems just as likely."

[1] "(i) is evolution gradualistic or pulsed? (ii) does trait evolution occur mainly at speciation or within a lineage? (iii) are changes at speciation adaptive or neutral? and (iv) how important is species selection in shaping patterns of diversity?"

Furthermore it is also possible to connect environmental changes[1] to another concept, *exaptation*, whose importance is emphasized in [Gould and Vrba, 1982]. Exaptation describes a shift in the function of a trait during evolution. If exaptation requires no environmental changes, the classic example being the feathers of non-avian dinosaurs potentially having initially evolved as a thermal regulator and subsequently as a means to fly, the transition from the first function to the second function thus constituting a exaptation. Exaptation is often proposed as taking place following a major environmental change such as: "Great Oxidation Event" in [Perry and Oliveira, 2010] or "PermianTriassic Extinction Event" in [Okada et al., 2010].

In [Jablonka et al., 1995], the authors develop the idea that phenotypic memory would be an evolutionary advantage in a changing environment. They discuss the effect of different types of fluctuating environment: random, strictly periodic and temporally patchy. And propose three different strategies: genotypic transmission, plasticity and phenotypic memory (carry-over strategy) are adapted to three different pace of environnemental changes: genotypic transmission for long term adaptation to lasting changes; plasticity for rapid changes occurring on time scales shorter or approximately equal to length of the reproductive cycle of the organism; and phenotypic memory for intermediate changes. The authors also sugested that carry-over strategy/phenotypic memory could be epigenetic in living organisms. Following the same line of thought, [Lachmann and Jablonka, 1996] postulated the existence of phenotypic inheritance induce by fluctuating environment and strongly emphasize the existence of Epigenetic Inheritance Systems (EIS) proposed in [Smith, 1990] described here as a "Lamarckian"[2] inheritance system. They modeled the effects of oscillating variations, such as seasonal or daily cycles, on phenotypical inheritance. Their model predicts that when the environmental cycle is longer than the reproductive cycle, while remaining relatively short otherwise, heritable variations produced by non-DNA inheritance

[1]By using the term "environmental changes" here and not that of "environement fluctuations" we want to emphasize here a certain kind of environement fluctuations: non-cyclic discrete changes.

[2]The authors refers to the inheritance of acquired characters even though as seen in the Section 2.1.3 and 2.1.2 this theory was not really invented by lamarck.

systems are likely to be observed. The authors also make the link between these mechanisms and the emergence of complex organisms:

> "The evolution of EISs in unicellular organisms was probably crucial to the evolution of multicellular organisms with complex development, since EISs maintain the determined state of cell lineages."

Continually varying or cyclic conditions might also explain the origin of EIS, as "epigenetic mutations" are more reversible and occur more frequently than genetic ones.

This idea is repeated in Jablonka et al. [2005c] which we previously discussed in Section 2.1.3, changing environments unmask variations in the capacity of individuals to make adjustments to new conditions, therefore promoting plasticity and multilevel selection. These authors contend that "For a lineage in a constantly changing environment, switching among several alternative heritable states was probably an advantage. While cells in one state survived in one set of conditions, those in other states did better in different circumstances." (*ibid.*, p. 318).

The existence of EIS in living organisms and their benefits overlooked fluctuating environments is taken in very recent work in biology such as [Heard and Martienssen, 2014].

In other recent publications, environmental fluctuations have also been linked to many other central properties and mechanisms of evolution, among which: "robustness" [Visser et al., 2003], plasticity [West-Eberhard, 2005], developmental or "evo-devo" processes [Muller, 2007] and "niche construction" [Laland et al., 2016].

2.1.5 Evolutionary Progress

The popular belief that a form of *evolutionary progress* (EP) emerges naturally as a result of the process of natural selection may seem obvious, but as described by [Shanahan, 2000], it is a controversial view in the field of biology. Early biologists and naturalists to be interested in evolution, such as Buffon, Lamarck or Darwin assumed the existence of a progression towards complexity of living beings and the notion has still nowadays several supporters including Richard

Dawkins and Ernst Mayr while most evolutionary biologists, like for example, William Provine and Stephen Jay Gould strongly criticise this idea: "Progress is a noxious, culturally embedded, untestable, nonoperational, intractable idea that must be replaced if we wish to understand the patterns of history", [Gould, 1988].

There are various reasons for this rejection of the EP concept in biology. For one thing, the notion of EP is sometimes regarded as a remnant of a very anthropocentric view of the classification of living beings (a legacy of Aristotle), which can also be seen in several predecessors of the Darwinian model of natural selection, such as theories proposed by Buffon or Lamarck.

Additionally, the concept of EP is usually linked to the existence of an increase in the complexity of multicellular[1] living organisms. And this idea is challenged in evolutionary biology for various reasons, summarized in [McShea, 1991].

Another argument, which could be seen to weaken the EP hypothesis, has been put forward by researchers such as [Johnson et al., 2012] in which they highlight numerous examples of species losing previously acquired evolutionary traits, possibly induced by ecological changes.

But the reason could simply be that the concept of evolutionary progress has not yet been fully formulated and developed. In that regard, [Shanahan, 2012] proposed a definition of evolutionary progress as "intergenerational directional change embodying improvement in the properties characterising a population of biological entities". A change should be considered directional if it "has a direction over a given time interval" and therefore "if the value of one of its properties increases during that time interval". It is important to note that, according to this definition, evolution acts on a multitude of populations themselves having multiple characteristics. It is, then, quite possible that evolutionary progress exists for a particular lineage of living beings while, in another lineages, for the same trait, there is no evolutionary progress or its direction is reversed.

[1]The transition from unicellular to multicellularity in some organisms is undoubtedly considered as an increase in complexity, the question is whether such events are rare epiphenomena or signs of a general trend.

2.2 Evolutionary Computation

The similarity between certain issues of computing and natural selection dated early days of computing, in [Turing, 1950] discussing the posibility to simulate the human brain, Alan Turing considers:

> "We have thus divided our problem into two parts. The child-programme and the education process. These two remain very closely connected. We cannot expect to find a good child-machine at the first attempt. One must experiment with teaching one such machine and see how well it learns. One can then try another and see if it is better or worse. There is an obvious connection between this process and evolution, by the identifications
>
> Structure of the child machine = Hereditary material
>
> Changes of the child machine = Mutations
>
> Natural selection = Judgment of the experimenter
>
> One may hope, however, that this process will be more expeditious than evolution. The survival of the fittest is a slow method for measuring advantages. The experimenter, by the exercise of intelligence, should be able to speed it up. Equally important is the fact that he is not restricted to random mutations. If he can trace a cause for some weakness he can probably think of the kind of mutation which will improve it."

According to [Fogel, 2006], the earliest published record of this idea in computer science occurred a few years later in [Barricelli et al., 1954].

The objective of this section is to provide an overview of the field of *Evolutionary Computation* while focusing more specifically on *Genetic Programming*, *Artificial Life* and the links maintained by these fields with evolutionary biology. We mean by *Evolutionary Computation*, the set of the re-uses of the principles of evolution by natural selection in computer science.

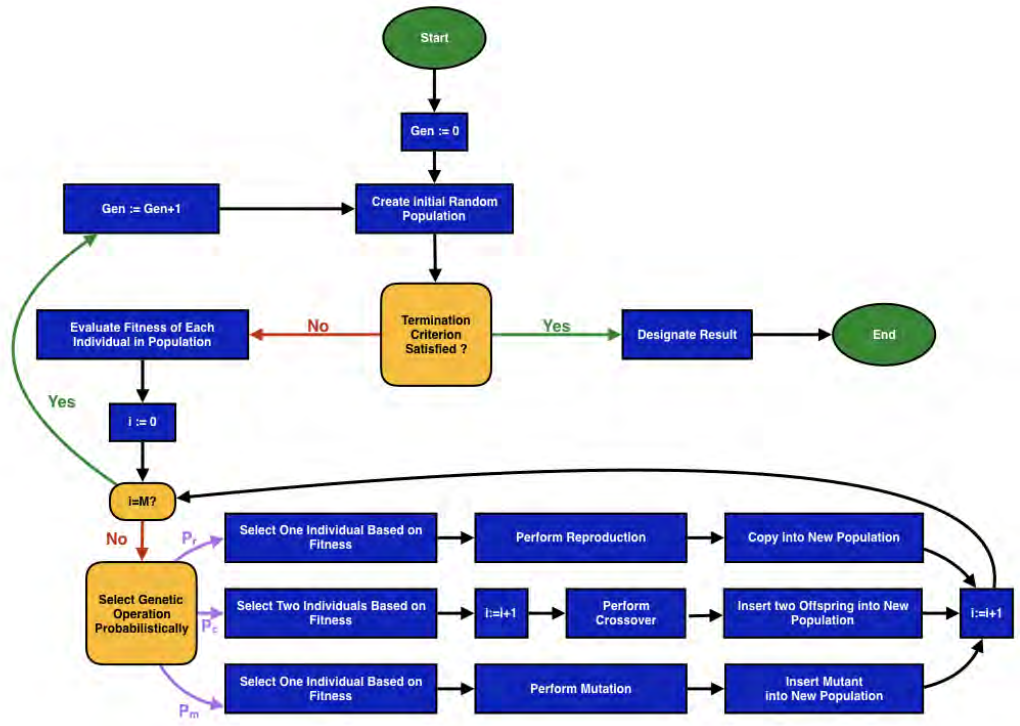


Figure 2.2: Flowchart, re-drawn from [Koza, 1994], of the steps of a "conventional genetic algorithm operating on strings". The index i refers to an individual in a population of size M. The variable Gen is the current generation number. P_m, P_c and P_r respectively show the probability of mutation, crossover and reproduction.

2.2.1 Genetic Algorithms

One of the first uses of natural selection as a mean to find an optimal solution to a problem was proposed in [Rechenberg, 1964], where an aluminium plate flexible at five points was evolved in an open wind tunnel to find a shape with minimum drag. Following this idea, genetic algorithms are to solve a problem by evolving a population of individuals encoded in a binary representation. The most famous formalization of this concept was probably made in [Holland, 1975]. Genetic algorithms, represented as a flowchart in Figure 2.2, are usually implemented as follows:

1. The problem to address is defined as well as a fitness function to evaluate how a potential solution is close to a perfect solution.

2. A population of potential solution is initialized according to certain constraints. Typically, each individual is encoded as a vector represented by a binary string called chromosome.

3. Each chromosome is decoded in a form suitable for evaluation by the previously defined fitness function, then its fitness score is computed.

4. A probability of reproduction is assigned to each chromosome based on their fitness score.

5. A new population is generated from the previous population. The chromosomes of the previous population are selected according to their probability of reproduction then a genetic operator, such as crossover or bit mutation, is applied to generated their offsprings.

6. The process is stopped if a suitable solution is found or if the allocated computational resources have been exhausted otherwise the process returns to step 3.

Since [Holland, 1975] the EC community has focused on the greater importance of crossover with respect to mutations. Typically EC researchers often use a proportion of 0.9 crossovers against 0.1 mutations. However from an optimization

point of view it has not only advantages: crossovers are more computationally expensive than the point mutations, they create *code bloat*[1].

Different crossover operators are used for genetic algorithms, the easiest is to first take two parents, secondly to randomly select a position where to cut the two binary strings, and finally to create two offspring by attaching the first part of the first binary string the second part of the second binary string, and vice versa.

2.2.2 Genetic Programming

The term "Genetic Programming" was used for the first time in [Koza, 1990]. The paradigm of GP, represented as a flowchart in Figure 2.3, is described as follow:

1. Generate an initial population of random compositions of the functions and terminals of the problem (computer programs).
2. Iteratively perform the following substeps until the termination criterion has been satisfied:
   (a) Execute each program in the population and assign it a fitness value according to how well it solves the problem.
   (b) Create a new population of computer programs by applying the following two primary operations. The operations are applied to computer program(s) in the population chosen with a probability based on fitness.
      i. Copy existing computer programs to the new population.
      ii. Create new computer programs by genetically recombining randomly chosen parts of two existing programs.
3. The best computer program that appeared in any generation (i.e., the *best-so-far individual*) is designated as the result of genetic programming. This result may be a solution (or an approximate solution) to the problem.

[1]Code bloat is defined as a growth of a program size without a corresponding gain in performance.

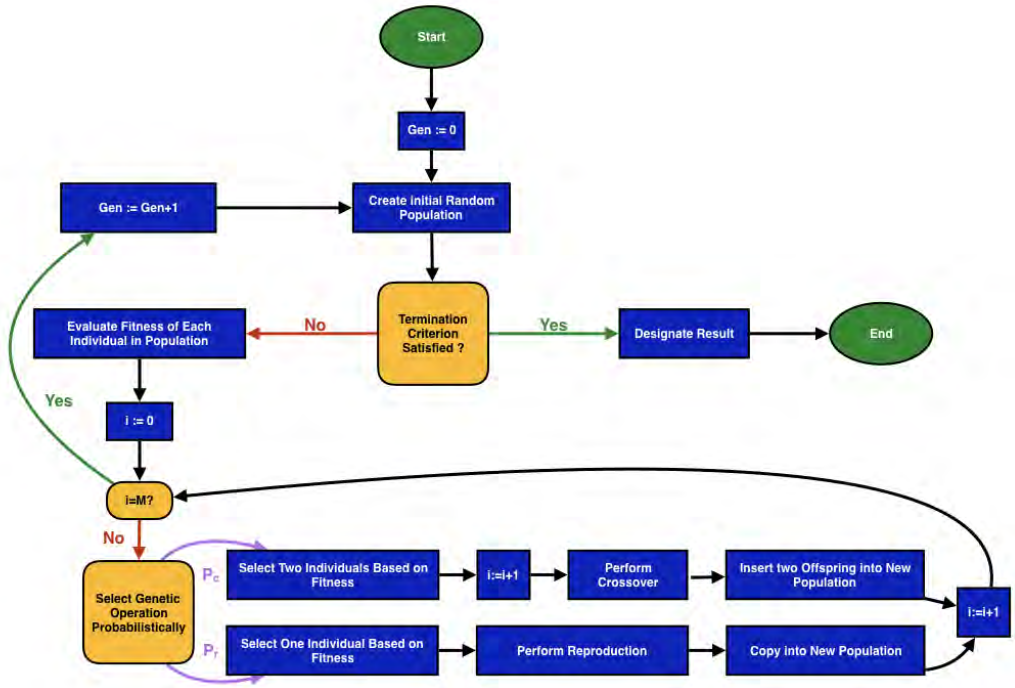


Figure 2.3: Flowchart, re-drawn from [Koza, 1994], of the steps of a a GP. The index i refers to an individual in a population of size M. The variable Gen is the current generation number. P_c and P_r respectively show the probability of crossover and reproduction. The only difference Koza makes with Figure 2.2 is the absence of mutations. Koza was considering them as an optional feature of the paradigm of GP alongside with permutation, editing, encapsulation, and decimation.

The most important innovation of GP is to evolve executable programs represented as trees, instead of binary strings used most of the time by genetic algorithms. Each tree is composed of nodes and leaves; nodes are mathematical or boolean functions such as $+$, $\times$, OR, AND that take arguments, while leaves are terminals, typically numbers or booleans. GP opened up new possibilities compared to genetic algorithms such as the evolution of decision tree, more flexibility in terms of individuals sizes etc. Several other forms of GP appeared following the work of Koza. We present here only a few of them selected by their technical and/or philosophical proximity with the methods we also use.

2.2.2.1 Linear GP

The principles of *Linear Genetic Programming* were presented for the first time in [Banzhaf, 1993] as a way to reconcile GP and genetic algorithms. A summary of the development of this method during the next two decades may be found in [Brameier and Banzhaf, 2007]. Instead of using a tree representation of individuals, LGP evolves sequences of instructions from an imperative programming language or from from a machine language. Typically in LGP, a program will consist of a series of instructions executed linearly, and carried out on a predefined list of variables called registers. A major driving motivation of LGP, is that because of the hierarchical structure of a tree-based GP, the most frequent mutations have the least phenotypic effect, whereas with a linear structure changes all magnitudes have similar frequency.

> "The influence of change in a linear structure can be expected to follow the linear order in which the instructions are executed.[...] The hit rate of operators, on the other hand, in relation to the position of a node or instruction in that sequence is equal for the entire linear genome. Hence, it can be expected that in GP with linear representations, changes of all sizes are equally frequent." - [Banzhaf et al., 1998]

2.2.2.2 Grammatical Evolution

Grammatical Evolution was proposed for the first time in [Ryan et al., 1998]. Its main distinguishing feature is to use a representation of individuals as binary

strings subsequently mapped to program usually in the form of a tree expression through the use of the production rules of a context-free grammar. This separation permits one to use the GE mapping process in conjunction with any search algorithm that operates on binary strings. The binary string is here called *genotype* while the program resulting from the mapping is called *phenotype*. This system has the advantage of being closer to biological evolution by making more likely, for example the existence of neutral mutations altering the genotype but leading to the same phenotype.

2.2.2.3 Boosting

Boosting is a resampling method for improving the performance of any learning algorithm presented for the first time in [Freund, 1990] and further developed in [Freund et al., 1996]. The principle is to repeatedly run a given "weak learning algorithm" on various distributions of the examples in the learning/training data set, the distribution is modified to put more and more emphasis on hardest cases. It has been shown in [Maclin and Opitz, 1997] that boosting usually produces a resulting solution better than any of its individual component but that it is susceptible to noise and can quickly overfit a data set.

It was proposed in [Iba, 1999] and further developed among others in [Paris et al., 2001] and [Paris et al., 2003], to apply this method to GP. For that, a series of GP run is performed, each one using a different distribution of training points. The final result is the geometric median weighted by confidence coefficients of the best individuals of each run.

2.2.2.4 Novelty Search

Novelty Search has met some success in EC to avoid premature convergence but was only used for the first time in GP in 2010 in [Doucette, 2010]. It takes as fitness function the novelty of the individuals measured by comparing their Euclidian Phenotypic Distance or their Phenotypic Hamming Distance to a collection of stored archetypical individuals. More generally, in EC-based *Novelty Search* [Lehman and Stanley, 2011], the objective is to continuously find novel solutions, and consequently the target landscape continues to change.

2.2.2.5 Semantic GP

As described in [Krawiec, 2016], Semantic GP stem from the complexity of the genotype-phenotype mapping. Its aim is to tackle problems such as the production of a program that is functionally indistinguishable from one of its parents, although its source code is different or, conversely, to ensure that changes in the genotype does not result in the creation of an individual phenotypically uncorrelated to his parents. In semantic GP the source code[1] is named syntax while its behavior[2] is named semantic. In practice, the semantic $s(p)$ of a program p is most often based on the tests given with a synthesis task, typically $s(p)$ could be a vector of the predictions of for different training cases available.

Semantic GP seeks to improve the search operators used in GP and is particularly interested in crossover. Among the first works in SGP [Beadle and Johnson, 2008] proposed to describe the semantic action of crossovers while [McPhee et al., 2008] created a technique forcing each crossover to produce children which are behaviourally different to their parents and have the benefit to decrease code bloat.

But one of the most successful technique in Semantic GP is probably Geometric Semantic Programming (GSGP), which uses a crossover operator called geometric semantic crossover that guarantees for any pair of parents that their offspring will have some traits in common with both of them. GSGP have interesting properties such as the fact that, for the Euclidean distance, the geometric offspring is guaranteed to be at least as good as the worst of the parents. GSGP was first proposed in [Moraglio et al., 2012] and thereafter applied successfully to solve real-life problems among others in [Castelli et al., 2013].

Another alternative to randomly combining individuals in semantic GP is ESAGP (Error Space Alignment GP) [Ruberto et al., 2014] which optimally combines individuals with an aligned error vector or, groups three individuals with co-planar error vectors (error vectors existing on the same bi-dimensional plane which intersects the origin of the error space).

[1] Which could also be seen as its genotype.
[2] Which could also be seen as its phenotype.

2.2.2.6 Sequential Symbolic Regression

Sequential Symbolic Regression [Oliveira et al., 2014] (SSR) tackles symbolic regression problems where the target output is not discrete. SSR spreads the task of approximating the training data across a number of GP runs; each such run is termed an *iteration*. At the end of each iteration, outputs of the original problem are modified based on the use of a geometric semantic crossover [Moraglio et al., 2012] on the output of the best evolved solution in the current iteration[1]. Whereas Semantic GP combines individuals at random, SSR optimises the effect of the geometric semantic crossover operator in the next iteration by evolving the best match to the curent solution to solve the problem; however, each iteration is homogeneous and typically uses a large number of generations (50 or 100).

2.2.2.7 Linear Scaling

Different methods of scaling have been proposed to enhance genetic programming by deterministically improving generated individuals. We will use that proposal by [Keijzer, 2003]. Given that $y = gp(x)$ is the output of an individual on the input data x, the scaled individual would be such as $y' = a \times gp(x) + b$. With $a = \frac{\sum_{i=1}^{n}(t_i - \bar{t}) \times (y_i - \bar{y})}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$ and $b = \bar{t} - b \times \bar{y}$ where n is the number of cases, t_i is the target value for the case i and $\bar{y}$ and $\bar{t}$ respectively denote the average output and average target value.

2.2.2.8 Epigenetic GP

To our knowledge the first epigenetic approach to GP was proposed in [Tanev and Yuta, 2003]. The inspiration for this work comes from a particular epigenetic mechanism: the regulation of transcription of DNA by histones[2]. Indeed histones are involved in the cellular specialization and development by affecting the transcription of DNA. An epigenetic mechanism is added to the GP algorithm to evolve the behavior of a team of predator agents in a predator-prey

[1] The r factor is set randomly at the end of each iteration.

[2] Histones are proteins found in eukaryotic cell that helps compacting DNA strands and play a role in DNA methylation.

pursuit problem. The epigenetic mechanism consists of an overlay of histones activating or deactivating certain nodes of the program. Before evaluating the fitness of each program, different random configurations of histones are tested. The best configuration is selected according to a criterion which is not necessarily the same as the fitness function: here the communication between predators is used as a criterion for selection alternative to their ability to catch the target. The states of histones are not transmitted to the offspring as the authors assume that epigenetic changes are not inherited in the biological world:

> "We mimic the biologically plausible hypothesis that the information contained in chromatin is inheritable both through the development of the somatic cells and through the germline but that the epigenetic changes to somatic cells' histones are not believed to be inheritable through the germline. Thus, we regard the phenotype of the somatic cell (subject to beneficial EL via histone code modification) as relevant for fitness evaluation of the individual, while the genotype of the germ cell - as a genetic material involved in phylogenesis."

Another form of epigenetic programming was proposed more recently in [La Cava and Spector, 2015] and [La Cava et al., 2015] and applied to regression problems. Taking into account the recent work in evolutionary biology the model proposed here uses inheritable epigenetic changes:

> "Furthermore, epigenetics allow adaptations to gene expression to be inherited in offspring without explicit changes to the genotype. Indeed, it is now accepted that the previously discredited ideas of Lamarck, i.e. the inheritability of lifetime adaptations, are possible through epigenetic processes."

2.2.2.9 Genetic Programming Issues

Overfitting Inferring a general underlying pattern from a finite training set has long been the challenge for Machine Learning [Bishop, 2006] and GP faces the same challenge. When the learning algorithm is solely guided by a fixed/static training set, it is not uncommon for the learning algorithm to produce a model

that performs very well on the training data but not so well on the *out of sample* or *test* data that still represents the same phenomenon. Such a disparity in the training and test performances implies that the learnt model does not *generalise* well; an alternative explanation is that the model *over-fits* the training data. Realising the importance of this issue for GP, O'Neill et al. [O'Neill et al., 2010] highlighted this as an *open issue* in GP.

Bloat GP also faces the issue of *code bloat* [Blickle and Thiele, 1994; Langdon, 2000; Poli and McPhee, 2008; Soule and Foster, 1998] whereby the evolving expressions grow in size without a corresponding gain even in training performance. Although some evidence [Zhang and Mühlenbein, 1995] suggests that controlling the size can promote simplicity and reduce over-fitting, other studies call for a detailed understanding of the size-complexity relationship in GP [Azad and Ryan, 2010b, 2011; Vladislavleva et al., 2009a] or even reject the notion that controlling the size also reduces over-fitting empirically [Vanneschi et al., 2010; Larkin, 2010].

Premature convergence One of the problems facing researchers who use natural selection as an optimization mechanism in EC, is a decrease in the diversity of individuals during evolution which may lead to premature convergence to a local optimum [Hornby, 2006]. Several biologically-inspired solutions for avoiding premature convergence have been successfully tested. For example, inspired by an evolutionary arms race, [Hillis, 1990] used the coevolution of two populations[1] and [Lessin et al., 2013] employed an intermediate step goal to generate behavioral complexity in evolved virtual creatures. In both of these studies the researchers examined evolution that was goal orientated but also in some sense *open-ended*. Also, in Evolutionary robotics, [Haasdijk et al., 2014], among others, studied the ability to combine, in a single simulation, survival of the organism together with resolution of tasks. Thus, it may be useful to analyze the mechanisms that avoid or ameliorate premature convergence when evolution is open-ended.

[1] A population of solutions of network classifiers and a population of problems are co-evolved.

2.2.2.10 Summary on Genetic Programming

We can see through this partial list of GP methods that, in a few decades, it has strongly diversified and enriched. If the purpose of these innovations was mostly to make GP more efficient this was also accompanied by an interest in some of the latest advances in evolutionary biology.

In particular the idea of adding a learning mechanism has met with great success. We can note that, although the mechanisms are different, methods such as linear scaling can be compared to epigenetic GP. In both cases the idea is to add a process to improve individuals after they have been generated. Of course epigenetic GP uses mechanisms that more closely resemble mechanisms found in biology, for example the idea of disabling some parts of individual's genotype. Nevertheless there are many differences between these mechanisms and evolutionary implications of epigenetics. Note that reuse of epigenetic mechanisms in EC is complicated by the fact that the epigenetic term does not, in biology, refer to a unique mechanism but rather to a collection of mechanisms in which elements have been removed and added during the recent history of biology. If one of the potential features of evolution at the epigenetic level is to take place at a different pace than the evolution at the genetic level it may be noted as a significant difference that in biology there is no selection of random epigenetic patterns during the life of an organism.

For Jablonka, forms of learning with selection that take place over a lifetime and that can be passed from one individual to another exist in biology; but can also take place at the behavioral and the semantic level and are not exclusively parent-to-offspring transmissions. The epigenetic mechanisms allow cell specialization and development but they are roughly deterministic and react to the environment. From this point of view deterministic methods such as linear scaling would therefore be closer to the developmental role of epigenetics. But in biology these mechanisms are themselves supposedly emerged through natural selection; for example selected to allow organisms to adapt faster to environmental fluctuation.

Semantic GP focused on another aspect of natural selection: the fitness landscape. By offering to perform semantic crossovers, that is crossover more or less

based on the phenotype of programes, it ensures that the phenotype of the offspring of a program will be intermediate between that of both parents. The result is a perfect conical fitness landscape ideal from the perspective of an evolutionary process occurring exclusively by *phyletic gradualism*, the idea that evolution occurs continuously and by steady and gradual transformation. But as explained in the Section 2.1.4 *phyletic gradualism* is far from being consensual in biology where the environment and therefore the fitness landscape is fluctuating.

Finally we can relate Sequencial Symbolic Regression, Gradient Boosting and Novelty Search insofar as they all use variations in the fitness function and therefore in the fitness landscape. One can nevertheless note that aside from gradient boosting, which originated outside of the EC community, these methods are very recent.

While, of course, GP is based on a biological analogy it does not pursue the same objectives as biology. Therefore GP methods need not to be biologically accurate in order to be effective.

> "We thus need to go back frequently to the natural example of biological evolution and study what else can be learned, similar to biologists who for the most part are still learning about the complexities and intricacies of the evolutionary process in nature. We are not forced to copy or mimic as much as possible, but we should also not refrain from taking up more ideas from biology if they are useful, either." in [ONeill et al., 2010]

Nevertheless, we believe that environmental fluctuations is an essential feature of biological evolution, in addition closely linked to recent borrowings of GP in biology. Yet it has been little studied in GP, or essentially tackling it as a characteristic of some of the problems[1] to be solved as in [Sheehan, 2000]. That is why we propose to explore it further for "*stationary*" problems.

[1] Those problems as refer as "dynamic", "non-stationary" or "autonomously changing" problems.

2.2.3 Cellular Automata

A pioneer of computer science, John von Neumann took an early interest [Von Neumann et al., 1966] in the possibility of creating a clanking replicator. This interest led von Neumann and Ulam to design in the 1940's *cellular automata* (CA) an abstract framework to study self-replicating machines, which soon became a general model of discrete dynamical systems. Since [Ilachinski, 2001; Wolfram, 2002] CAs have been intensely studied experimentally and theoretically.

A CA consists of a regular grid of cells, each in a state selected from a finite set of possible states. The grid can be in any number of dimensions, but the most commonly used CA are two-dimensional or three-dimensional CAs. For each cell, a set of cells called its *neighborhood* is defined. An initial state is selected by assigning a state for each cell of the CA. A new generation is created according to a transition function, usually consistent for all cells, that uses the current state of each cells and the states of every cell in their neighborhood to, usually simultaneously, determine their new state. In a cellular automaton in two dimensions, two forms of neighborhoods (depicted in Figure 2.4) are usually used:

- *von Neumann neighborhood*, where the four nearest cells of the cell in question are taken into consideration;
- *Moore neighborhood*, where the eight cells closest to the cell in question are taken into consideration.

In [Wolfram, 1984] CAs are divided into four distinctive universality classes according to their dynamics from an initial initative configuration:

- *CLASS I*, where evolution[1] leads to homogeneous fixed points;
- *CLASS II*, where evolution leads to periodic configurations;
- *CLASS III*, where evolution leads to chaotic, aperiodic patterns;

[1]Here term evolution refers to the dynamics of the CA and not to evolution by natural selection.

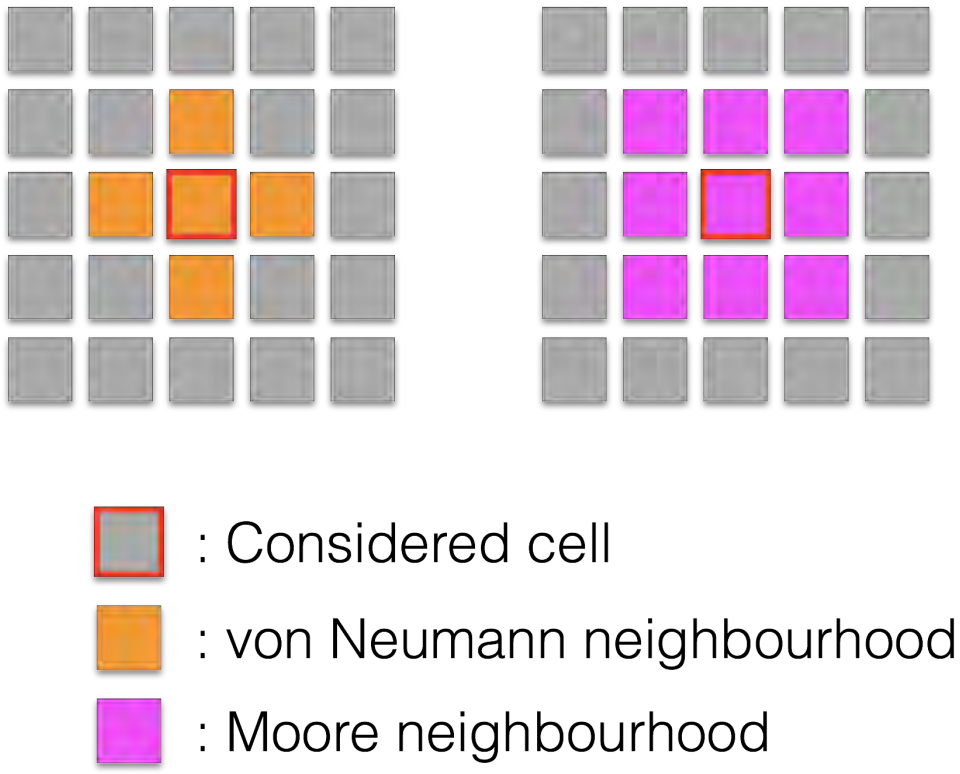


Figure 2.4: The neighborhoods of Moore and von Neumann are the most frequently used for two-dimensional CAs.

- *CLASS IV*, where evolution produces persistent, complex patterns of localized structures. Wolfram suspected this latter class of automata were capable of universal computation.

A key feature of traditional CAs resides in a consistent transition function for all cells, making them good models of homogeneous physical laws or intra-organismal cellular growth. However, some experiments have also explored the use of *heterogeneous* transition functions [Vichniac et al., 1986; Sipper, 1998; Sipper and Tomassini, 1999].

CAs are known to be a model of computation, as they can simulate a universal Turing machine [Wolfram, 2002]. Even some simple CAs have the property of computational universality, such as Conway's Game of Life, a two-dimensional CA in which binary cells switch on or off according to a simple counting rule [Conway et al., 2003]. Furthermore, CAs –or minor variants of CAs– are often used as predictive models of natural phenomena [Deutsch and Dormann, 2005; Wolfram, 2002], Alife models of self-reproduction [Sayama, 2004; Pan and Reggia, 2010; Yinusa and Nehaniv, 2011], and models of morphogenesis, sometimes as part of the creation of engineered design [Kowaliw et al., 2004; Miller, 2004; Tufte, 2006; Kowaliw et al., 2007; Bidlo and Vašíček, 2008].

2.2.3.1 Evolutionary Cellular Automata

Although CAs are generally used in areas very distinct from evolutionary computation John von Neumann had contemplated the connection between his model and natural selection.

> "In addition to this it must be remembered that conflicts between independent organisms lead to consequences which, according to the theory of "natural selection," are believed to furnish an important mechanism of evolution. As was seen at the end of Section 1.7.3, our models lead to such conflict situations. Hence this motive for evolution might also be considered within the framework of these models. The conditions under which it can be effective here may be quite complicated ones, but they deserve study."

Moreover, given the importance of CAs as models, and the size of the space involved, the search for "useful" CA rules is an attractive but difficult problem. There are several theoretical results on the impossibility of predicting analytically the outcome of a given CA rule set starting from a given configuration. The reverse attempt at generating local rules to create a target global pattern is also not possible in the general case. Under these constraints, a natural choice for exploring CAs with desired properties is an evolutionary search. Unfortunately, CAs are difficult to evolve naively [Ganguly et al., 2003; Kowaliw and Banzhaf, 2009], and when used in an application, variants of the traditional cellular automaton framework are often used to improve evolvability such as in [Kowaliw et al., 2004].

There have been several attempts to use representations inspired by GP for the (homogeneous) transition function, often in the context of generating self-replicating structures [Pan and Reggia, 2010; Bidlo and Vašíček, 2012].

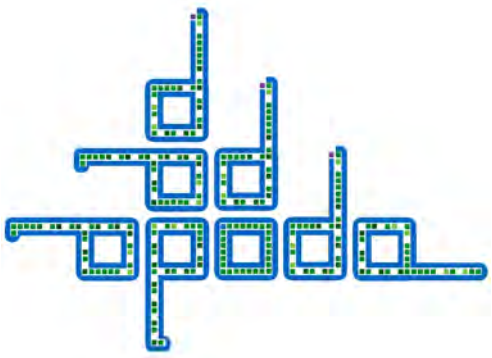


Figure 2.5: Langton loop.

There have also been attempts to evolve self-replicating structures within a CA, such as in [Sayama, 1998] where Hiroki Sayama proposed evolvable self-replicating loops, by adding three properties (mortality of individuals, interactions between individuals, and robustness to environmental variations) to non evolvable self-replicating loops similar to those described by langton in [Langton, 1984] and depicted in Figure 2.5.

2.2.4 Artificial Life

Christopher Langton coined the term "Artificial Life" and defends it as a field of study in itself. He defined it in 1989 as "The study of man-made systems that exhibit behaviors characteristic of natural living systems" in [Langton et al.,

1989]. Nevertheless this idea can be traced back to much further, for example in 1960 A. S. Fraser explicitly defends in [Fraser, 1957] the idea of studying certain aspects of biology through computer simulations:

> Genetic models can be devised, programmed, and tested within weeks or months; certainly with sufficient speed for an experimenter to examine many of the theoretical consequences of his ideas before he devises experiments with live organisms. This would seem to be the field in which this method can be used to the greatest value, though its use in the methodical examination of the importance of variables determining the effectiveness of selection is undeniable.

Similarly John von Neumann is deemed to have claimed that "life is a process which can be abstracted away from any particular medium"[1]. These two quotes reflect two different visions of Alife, the strong Alife and the weak Alife that echo the strong and weak Artificial Intelligence. "Strong Artificial Life" approaches are what Langton proposed in [Langton, 1986] as the ultimate goal of Alife[2]. Just as strong AI this is very strongly criticized perspective for example in [Olson, 1997] where it is defined as "the bold hypothesis that it is possible to go beyond simulation and literally create living organisms with the help of computers". On the other hand, the "Weak Artificial Life" offers to model and simulate the living organism to have a better understanding of their operation. A feature of the biology reinforces the relevance of Alife, indeed biology is not only an experimental science but also a historical science[3] with an interest in the history of life. And in this light, Alife offers us a sandbox to test at will an infinite number of different scenarios.

[1] This quote is often repeated but we have not been able to find its origin and therefore can not attest to its accuracy, the earliest reference to this quote that we found is in [Thalmann, 2005], also this quote is not dated, however, if it is not apocryphal, it is necessarily prior to the death of John von Neumann in 1957.

[2] "The ultimate goal of the study of Alife would be to create 'life' in some other medium, ideally a virtual medium where the essence of life has been abstracted from the details of its implementation in any particular hardware. We would like to build models that are so life-like that they cease to be models of life and become examples of life themselves."

[3] See [Cleland, 2002] for differences between historical and experimental sciences.

As explained by example in [Bedau, 2000] Alife is generally separated into three areas:

1. "Wet" Alife which is part of synthetic biology and seeks, for example, to create artificial cells from biochemical compounds.

2. "Hard" Alife which is part of robotics and seeks for example to create autonomous adaptive and intelligent behavior in the real world.

3. "Soft" Alife which is close to but not limited to EC and seeks for example to create virtual artificial evolving echosystems.

In the remainder of this chapter we will limit ourselves mainly to considerations about soft Alife, and more specifically to Open-Ended Evolutionary Simulations and works on changing environment.

2.2.4.1 Open-Ended Evolutionary Simulations

A major motivation for work in Alife is to create *open-ended evolution*. The properties that characterize the open-ended evolution concept often lack consensus but it can probably be summarized as "Loosely defined, an open-ended evolutionary system is one that is capable of producing a continual stream of novel organisms" as proposed in [Taylor et al., 2016]. Practically speaking there is no particular specified goal in open-ended evolutionary simulations: interesting and sometimes unexpected phenomena emerge from the interaction of individuals and their environment without using an explicit fitness function. [Fogel, 2006] proposed that the first example of work of this kind might be [Barricelli et al., 1954], and consist of a one-dimensional matrix of simple rules which would copy and move.

Tierra Perhaps the best known model is Ray's Tierra [Ray, 1992]. Tierra is based on a programming game: Core Wars. Invented in 1984 by the mathematician and philosopher Alexander Dewdey, Core Wars organizes competitions between programs (written in a specialized language called Redcode) in a virtual environment called Mars (Memory Array Simulator Redcode). Many phenomena obtained with Tierra approximate interesting phenomena in evolutionary biology: predator-prey, parasite-host, cooperation... Nevertheless, although Tom

Ray claimed his approach can lead to the open evolution "The approach has generated rapidly diversifying communities of self-replicating organisms exhibiting open-ended evolution by natural selection", [Bedau et al., 1997] showed that ecosystems similar to Tierra lead to cyclical behavior far away from the diversity of real ecosystems and therefore the "Open-ended evolution" qualifier.

Avida Later, inspired by Tierra, another evolutionary system called Avida became popular and was extended in many directions [Adami and Brown, 1994; Ofria and Wilke, 2004]. Avida works on the same principle as Tierra albeit with very significant differences: The instruction set used is similar to that of Tierra (Redcode from Core Wars), but unlike Tierra which localize instructions and pointers in a one dimension world in Avida every replicator uses a protected memory space. The organisms are located on a two-dimensional toroidal grid and can only perform local actions. Organisms have age and when an organism creates an altered copy of itself, it replaces the oldest organism of its neighborhood. Avida was used, among others subject, to study the emergence of complex features [Lenski et al., 2003; Adami et al., 2000].

Polyworld High-level systems exist as well such as Polyworld [Yaeger, 1994] which focuses on the evolution of the nervous system in a "rich complex ecology system". The simulation particularly stresses energy: any action, as well as the complexity of the nervous system has an energy cost in Polyworld. Food blocks are randomly generated in the environment and the critters[1] execute complex predefined function such as:

- Outputs: move, eat food, attack another critter, eat a dead critter, mate.
- Inputs: see / detect (other creatures / food / possible barriers), their energy level and a random entry.

The genome of each critter is coded for its speed, its color, the initial configuration of its neural network, and the frequency of mutations in its genome. The behavior of the creature develops during its life through learning with Hebb's rule (first

[1]The evolving agents are named critters in Polyworld.

proposed in [Hebb, 1949]). Polyworld facilitates the observation of phenomena such as the emergence of a system prey-predator.

Novelty Search Another quite different approach of open-ended evolution simulation in that it uses an explicit fitness function is novelty search proposed e.g. in [Lehman and Stanley, 2008] where:

> "That is, instead of searching for a final objective, the learning method is rewarded for finding any instance whose functionality is significantly different from what has been discovered before."

If this method was initially proposed in connection with the open-ended evolutionary simulations, as seen in Section 2.2.2.4 novelty search met later success outside this framework. This highlights that research on open-ended evolution are not only interesting in the context of a better understanding of the evolutionary biology mechanisms but also to improve other forms of EC as an optimization method.

In order to combine the multiple definitions of open-ended evolution, recent work has characterized open-ended evolution as a combination of ongoing adaptive novelty and ongoing growth of complexity in such as for example in [Taylor et al., 2016] or as a combination of five main features: continuous production of novelty; continuous increase in diversity; continuous increase in complexity; shifts in individuality such as those often associated with major transitions in evolution; and continuous change in the population in [Vostinar et al., 2016]. The growth of complexity can be linked to the question of evolutionary progress. We have seen in Section 2.1.5 that the existence of evolutionary progress was very controversial in biology.

On the other hand, in the field of Alife, though the creation of artificial long-term EP in open-ended simulation has not received consensus, the existence of evolutionary progress in the living is often taken for granted as illustrated by, for example, [Ciliberti et al., 1999] and its association with the evolution of complexity is presented as one of the goals of Alife by [Bedau, 2003].

These differences in approach may be explained by the proximity of the field of Alife with other evolutionary paradigms of optimization and problem solving which are similarly inspired by natural selection, such as GP and Genetic Algorithms, where the existence of evolutionary progress is not disputed.

It is also noted that the issue of open-enededness in Alife simulations can be connected to the problems that confront the evolutionary programming used as an optimization method e.g. in GA and GP: the continual stream of novel organisms may oppose the premature convergence while bloat possibly being considered a trivial growth of complexity.

2.2.4.2 Environmental Fluctuations in Alife Simulations

In parallel to biological research, a number of studies in Alife, especially evolutionary robotics [Floreano and Urzelai, 2000], have also investigated environmental variations, some of them explicitly defining the environment as a driving evolutionary force [Bredeche and Montanier, 2012].

Others, such as Lipson et al. [2002], showed a correlation between the modularity and the rate of change of external resources, while Yu [2007] observed that populations exploit neutrality to cope with environmental fluctuations and can evolve a type of evolvability under two alternating objective functions. Both of these simulation works relied on GP and explicit fitness functions.

Regarding Open-Ended Evolutionary Simulations, Avida has been used to study the effects of environnemental changes on sleep in [Beckmann et al., 2007] and on the emergence and maintenance of sexual reproduction [Misevic et al., 2010]. The first experiment successfully showed that sleep could appear in response to an environment in which food availability is periodic and declines over time. While the second showed that regular and frequent changes of the effects of potentially beneficial or harmful substrates could, in some cases[1], promote sexual reproduction.

[1]Depending on the magnitude and frequency of changes.

2.3 Conclusion

The existence of environmental changes is evident in biology and if they are not essential to the theory of natural selection as formulated by Darwin or Lewontin, they are connected to many features of natural selection as it occurs for living organisms ie plasticity, development, and maybe "Lamarckian" multilevel selection. Conversely, computer simulations used in EC are stable and free of equivalent of day-night cycles or other environmental crisis unless we implement them. This might be what is needed if we want to reproduce such features of biological evolution.

Through this quick overview of evolutionary biology and EC we hope to have shown that the environnemental fluctuations are an interesting elements to explore further in EC. First because if this issue was not the main object of biology major works since [Levins, 1968], it regularly returns as potential explanatory mechanism in the controversies that glazed evolutionary biologies the last 30 years. Secondly, although if this issue has been explored several times in EC, it have been very little compared to recent debates that have animated evolutionary biology, ie the role of epigenetics, the selection at many levels.

Part II

Experiments of environmental fluctuations in computer simulations of evolutionary programming and artificial life.

3

Long-Term Evolutionary Dynamics in Heterogeneous Cellular Automata

3.1 Introduction

In this chapter, we introduce HetCA (for "heterogeneous cellular automata") a new Alife model based on a discrete dynamical systems framework. We have extended classical 2D cellular automata (CA), our chief modification being the allowance for heterogeneous transition functions. These changes convert a CA system into a new kind of "ecosystemic" model, where different genomes compete for existence. The value of such a model resides precisely in its simplicity: we aim to observe that long-term dynamics can be achieved quite naturally, given appropriate and plausible hypotheses.

We do not introduce here any hard-coded environmental fluctuations. Rather, we seek to measure whether this model itself generates long-term environmental dynamics. To study this open-endedness, first we will show that our simulation is capable of supporting better long-term dynamical behaviour than control models such as homogeneous CA and the Game of Life. In particular, we will exhibit examples of the strategies discovered by our system which are characterized by

the *emergence of competitive behaviour*. Secondly we will study the existence of EP in HetCA.

EP hypothesizes that evolution tends toward a goal such as greater complexity of individuals. It's a truism to speak about EP in terms of EC where evolution is used as an optimisation method to solve problems, but, as seen in Chapter 2 the existence of EP is much less obvious in biology which evolves in an open-ended evolutionary process. To measure evolutionary progress in HetCA we use three traits: the density, size and robustness of evolved individuals. To do so we compare the properties of genotypes taken at different stages of the evolutionary process. The study of these three features allows us to analyze more precisely the nature of evolutionary strategies in HetCA. We hypothesise that the size of the genotype together with the robustness and the density of its phenotype may serve as useful measures of the effectiveness of that genotype, we examine the existence of a correlation between these measures.

This chapter is organized as follows: firstly, in the Section 3.2, we introduce our novel modifications to cellular automata, HetCA , and describe their typical effects. Following this, in the Section 3.3, we present experimental settings and define the measures which we have chosen to use as evidence for the existence of long-term evolutionary dynamics and EP. Then, in Section 4.2.2 we present the simulation results in the and finally, we explore some specific examples of system outputs to discuss their behaviour qualitatively, in the Section 3.5.

3.2 The HetCA Model

Our model HetCA diverges from classical 2D cellular automata in three ways:

- cells have properties of *decay* and *quiescence*;
- each cell contains its own transition function, also called *genotype*, which has the ability to evolve over time and be copied to its neighbours;
- these genomic transition functions are represented in a parsimonious way (see Section 3.2.2).

Here, by "classical" two-dimensional CA we mean:

- a toroidal lattice "world" of cells, $W = \{c = (x, y)\}$, with $|W| = w \times h$ (width and height, here 800×600)
- a neighbourhood for each cell: $\nu(c) = \{c, c_1, ..., c_{|\nu|}\}$, where $|\nu_{\mathrm{V}}| = 5$ denotes the von Neumann neighbourhood and $|\nu_{\mathrm{M}}| = 9$ the Moore neighbourhood.
- a discrete time, $t = 0, 1, ...$
- an alphabet of cell states, $\Sigma = \{s_1, ..., s_{|\Sigma|}\}$
- a state $s(c, t) \in \Sigma$ for each cell $c \in W$ at each time t
- a single system-wide transition function $\varphi : \Sigma^{|\nu|} \to \Sigma$

A CA is initialized with random states across the world. At time $t+1$, each state is updated in parallel according to

$$s(c, t + 1) = \varphi(\mathbf{s}(\nu(c), t)), \tag{3.1}$$

where $\mathbf{s}(\nu(c), t) = \{s(c, t), s(c_1, t), ...\}$ is a $|\nu|$-tuple representing the collective state of c's neighbourhood including itself. Therefore, an uncompressed representation of a CA transition function requires $|\Sigma|^{|\nu|}$ elements, or a listing for every possible collective neighbourhood state, and the total number of such rules is an astronomical $|\Sigma|^{|\Sigma|^{|\nu|}}$. None of these concepts alone is novel: their conjunction, however, is. In short, HetCA is initialized as a world state $\mathbf{s}(W, 0)$ of cells with randomly initialized state values and genomes. We will refer to cells which are neither decaying nor quiescent as *living* cells. For each cell in W, the following processes run in parallel:

- **age:** old living cells become decaying cells, old decaying cells become quiescent
- **copy:** cells can accept genetic material from living neighbours, which can make quiescent cells living cells again.
- **mutate:** a living cell's genetic material can change
- **update:** a living cell executes its genome, i.e. applies its transition function to its neighbourhood

A formal description is available in Algorithm 1. We describe these processes in more detail below.

Algorithm 1 HetCA update rule: accepts a world state $\mathbf{s}(W,t) = \{s(c,t)\}_c$, outputs the next world state $\mathbf{s}(W,t+1)$ and is executed in parallel for all cells.

for each cell in the world ($c \in W$) **do**
  increment cell age $\rightarrow a(c,t+1) := a(c,t) + 1$
  if cell is in *quiescent* state? ($s(c,t) = q$) **then**
    copy genotype from random eligible neighbour $\rightarrow$
      $\varphi_{c,t+1} := (\Phi_{c,t} \neq \varnothing)\ ?\ \mathcal{U}\big[\Phi_{c,t}\big]\ :\ 0$
    if copy happened? ($\varphi_{c,t+1} \neq 0$) **then**
      update state $\rightarrow s(c,t+1) := \varphi_{c,t+1}(\mathbf{s}(\nu_{\mathrm{M}}(c),t))$
      reset age $\rightarrow a(c,t+1) := 0$
    end if
  else if cell is in *decaying* state? ($s(c,t) = d$) **then**
    if cell older than decay end? ($a(c,t+1) > a_{\mathrm{dec}}$) **then**
      set state to quiescent $\rightarrow s(c,t+1) := q$
    end if
  else if cell is in *living* state? ($s(c,t) \in \Lambda$) **then**
    if cell older than life end? ($a(c,t+1) > a_{\max}$) **then**
      set state to decaying $\rightarrow s(c,t+1) := d$
      reset age $\rightarrow a(c,t+1) := 0$
    else
      copy genotype from eligible neighbour $\rightarrow$
        $\varphi_{c,t+1} := (\Phi_{c,t} \neq \varnothing)\ ?\ \mathcal{U}\big[\Phi_{c,t}\big]\ :\ \varphi_{c,t}$
      possibly mutate genotype with probability $p_{\mathrm{mut}} \rightarrow$
        $\varphi_{c,t+1} := \mathrm{mutate}(\varphi_{c,t+1})$ (see Section 3.2.4)
      update state $\rightarrow s(c,t+1) := \varphi_{c,t+1}(\mathbf{s}(\nu_{\mathrm{M}}(c),t))$
    end if
  end if
end for

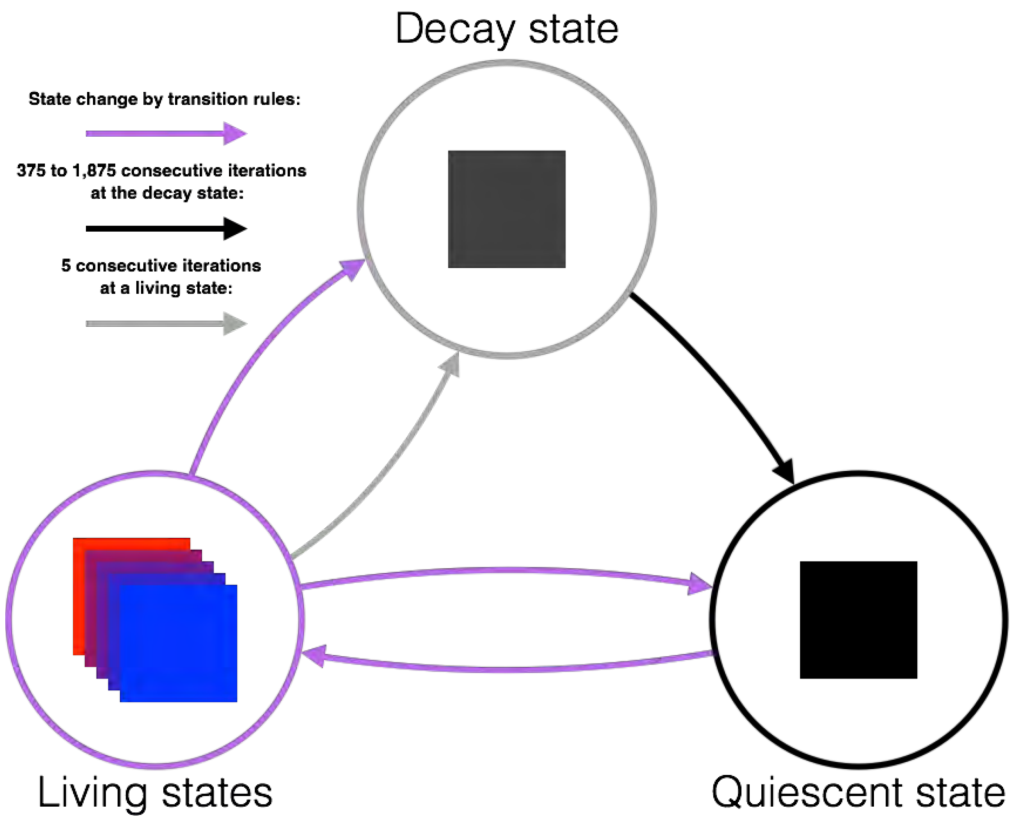


Figure 3.1: Life cycle of cells in HetCA. Here $agemax = 5$ iterations and $decay\,time = 375\,to\,1875$ iterations as specified in Table 3.2.

3.2.1 Cell Quiescence and Decay

Since the HetCA alphabet comprises two special states, "quiescent" and "decay", and the other states are "living", we denote it here by $\Sigma = \{q, d\} \cup \Lambda$, where $\Lambda = \{l_1, ..., l_{|\Sigma|-2}\}$. Our metaphor is that living cells "age" and eventually "die", becoming inert for some period of time. During the decay process, they continue ageing and eventually become quiescent, creating "free space".

A living cell or decaying cell c ages by incrementing an internal counter $a(c, t)$. If a living cell's counter passes a threshold $a_{\max}$, its state is converted to decaying: $s(c, t) = d$, its genotype is discarded, and its age is reset: $a(c, t) = 0$. If a decaying cell's age counter passes another threshold, a_{dec}, then it is turned into a quiescent

cell: $s(c,t) = q$.

Our motivation for including decay and quiescence was to create a form of *competition* for the cells. Decaying matter is a challenge for cell colonies, as it makes simple maximal growth unsustainable: cells need to "learn" to survive around decay, e.g. use it as a means of defending their genetic territory. To encourage this trend, we set the amount of time necessary for the elimination of decay to a much larger value than the lifespan of a living cell: $a_{\text{dec}} \gg a_{\text{max}}$.

3.2.2 Heterogeneous Transition Functions and Genetic Transfer

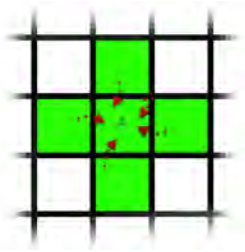


Figure 3.2: Genotype copy at the start of a CA iteration, if cell A is not in Decay it will randomly receive a genotype from any cell shown here in green (von Neumann (VN) neighboring cell A and cell A itself) that is neither in Decay nor Quiescent.

By "heterogeneous" CA, we mean that each living cell c contains its own (potentially unique) transition function, which may also vary over time, thus is denoted by $\varphi_{c,t}$. The world W is initialized with a uniform random sampling of cell states $\{s(c,0)\}_{c\in W}$ and transition functions $\{\varphi_{c,0}\}_{c\in W}$. Then, each cell determines its next state according to its own transition function: $s(c,t+1) = \varphi_{c,t}(\mathbf{s}(\nu_{\text{M}}(c),t))$.

Moreover, a transition function can be randomly *copied* between neighbouring cells, but under only two conditions: (1) from a living cell to a living or quiescent cell, and (2) if this function is "resolved" in the new neighbourhood around that cell, meaning that its output must be a living cell, too. The motivation for this requirement is to discourage random occurrences of decaying and quiescent cell states because these states are "sinks" (as they are associated with a loss of transition function, hence are not updated).

We now describe this process more formally. We will use ν_{M} to resolve transition functions, and ν_{V} to chose neighbourhoods for resolution. Given a transition function φ, we denote by $\mathscr{R}_t(\nu_{\mathrm{M}}(c), \varphi)$ the fact that a neighbourhood $\nu_{\mathrm{M}}(c)$ "resolves" φ at time t and define it by

$$\mathscr{R}_t(\nu_{\mathrm{M}}(c), \varphi) \;\Leftrightarrow\; \varphi(\mathbf{s}(\nu_{\mathrm{M}}(c), t)) \in \Lambda. \tag{3.2}$$

We also denote by $\lambda_t(c)$ the subset of $\nu_{\mathrm{V}}(c)$ containing the living neighbours of c at t: $\lambda_t(c) = \{c' \in \nu_{\mathrm{V}}(c) \mid s(c', t) \in \Lambda\}$. Then, if cell c is living or quiescent at time t, it may receive at time $t+1$ the transition function from one of its living neighbours c' chosen randomly, only if that new function is resolved the current neighbourhood state of c. This reads

$$\begin{gathered} \varphi_{c,t+1} = \mathcal{U}\big[\Phi_{c,t}\big]\,, \text{ where} \\ \Phi_{c,t} = \{\varphi_{c',t} \mid c' \in \lambda_t(c) \text{ and } \mathscr{R}_t(\nu_{\mathrm{M}}(c), \varphi_{c',t})\} \end{gathered} \tag{3.3}$$

is the set of eligible neighbouring transition functions, and $\mathcal{U}$ denotes a uniformly random draw from a set of elements. Note that if the original cell c was quiescent, the set of neighbouring transition functions might be empty. In this case, there can be no copy and c remains quiescent.

3.2.3 Transition Function Representation

We now describe the genomic format of $\varphi_{c,t}$. Given that uncompressed transition functions require an enormous state space $|\Sigma|^{|\Sigma|^{|\nu|}}$, and also recalling that transition functions are notoriously difficult to evolve, we elected to use a form of compression of their genomic representation. To this end, we developed a new representation for CA transition functions inspired by Linear Genetic Programming (LGP) [Brameier and Banzhaf, 2006]. LGP is a desirable choice as its programs are easy to initialize and mutate, fast to execute, naturally modular and resistant to bloat, and make use of neutral code, which is believed to increase evolvability [Brameier and Banzhaf, 2006; Poli et al., 2008].

We name our new representation of φ a **CA-LGP** program. Like any CA transition function, it maps the space of neighbourhood states to a new cell state: $\Sigma^{|\nu|} \rightarrow \Sigma$, but also, like any LGP representation, it provides an

evolvable representation framework in which φ can be decomposed into an alphabet of elementary functions tuned by a few parameters. A CA-LGP program φ consists of:

- a list of $n_{\text{reg}} = |\nu| + |\Sigma| + n_{\text{add}}$ registers, written $\{R_i\}$:
  - $|\nu|$ *neighbourhood registers* holding the state values $\mathbf{s}$, the input to the program
  - $|\Sigma|$ *state registers*, genetically-specified constants
  - n_{add} *additional registers*, also genetically-specified constants
- a list of at most n_{prog} program statements, each of the form $R_i = \text{op}(R_j, R_k)$ for some operator op and some register indices i, j, k.
- a return statement, which returns the state associated with the maximum value state register.

An example of CA-LGP program is shown in Figure 3.3.

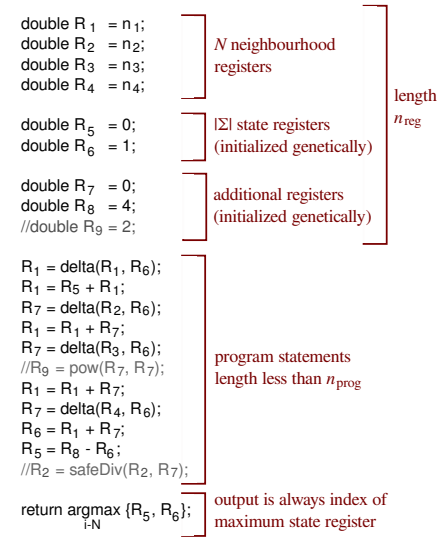


Figure 3.3: Example CA-LGP program in pseudo-java notation. This program accepts a von Neumann neighbourhood, and assumes an alphabet of two cell states, $\Sigma = \{0, 1\}$. It first computes the number of state-1 cells in the neighbourhood, then the number of state-0 cells. Finally, it outputs the central state reflecting the majority. Note the presence of neutral code, displayed in grey.

3.2.3.1 CA-LGP Genetic Initialization

A CA-LGP program can be initialized randomly by:

- choosing a number of additional registers
- specifying initial values for the state registers and additional registers with integers chosen uniformly randomly in the range $\{0, ..., |\nu|\}$. This maximum value is small enough to not encumber evolvability, but large enough to potentially act as a divisor for normalization (say, to compute averages).
- specifying each program statement by selecting four values uniformly randomly for i, j, k, and op, among the available registers and function set (Table 3.1).

Table 3.1: Function set.

op. **name**	**action** on inputs (x, y)
abs	$\|x\|$
plus	$x + y$
delta	1, if $\|x - y\| < 1/10000$; 0 o.w.
dist	$\|x - y\|$
inv	$1 - x$
inv2	safeDiv$(1, x)$
magPlus	$\|x + y\|$
max	$\max\{x, y\}$
min	$\min\{x, y\}$
safeDiv	x/y if $\|y\| > 1/10000$; 1 o.w.
safePow	x^y, if defined; 1 o.w.
thresh	1, if $x > y$; 0 o.w.
times	xy
zero	1, if $\|x\| < 1/10000$; 0 o.w.

3.2.4 Genetic Mutation

Given some particular CA-LGP, we can apply a mutation operator to generate a genetically similar CA-LGP. For this, we randomly choose either a micro- or a macro-mutation:

- Micro-mutation: for each additional register and for each program statement, a mutation is applied with a small probability and re-initializes one component of the statement.
- Macro-mutation: we choose one of the following, with equal probability:

- If the program size is less than maximum n_{prog}, a randomly initialized program statement is added.
- If the program size is greater than 2, a randomly selected program statement is removed.

The probability of a living cell being mutated, i.e. executing the above algorithm, is set to a constant $p_{\text{mut}} = 0.0008$.

3.3 Experimental Setting

As previously mentioned, HetCA is initialized with a uniformly random mixture of cell states and genomes. The vast majority of genomes turn to quiescence or decay within the first few time steps. This is not surprising since two of our cell states, q and d, are sinks, and randomly initialized genomes have no reason to avoid such states.

If any, only a small number of genomes (between one and three) survive this initial extinction, forming small groups of cells in between fields of decay[1]. Once decaying cells turn to quiescent, these surviving clumps quickly grow to cover the entire world. This often leads to a second or even third extinction event, after which the change from decay to quiescence becomes desynchronized. Usually, these early extinction events disappear before time step 10,000.

Following this, various populations come and go, that is, loose clusters of similar phenotypic patterns can be seen throughout the world, with new patterns materializing and old patterns disappearing with time. Occasionally, a new phenotypic pattern will emerge and quickly cover the entire world, indicating that a beneficial mutation has occurred. Informal experimentation convinced us that using different parameter values for the size of the world, and for $a_{\max}$ and a_{dec},

[1]It is not uncommon for all genomes to die following the initial extinction events, either immediately (before iteration 200) or, less frequently, after the first re-growth stage (before iteration 1500). This leaves a trivial world devoid of genomes. Given our particular parameters, these extinction events occur approximately 75% of the time. In these cases we simply restart the simulation, which is not computationally expensive since the run time to an extinction is short.

led to similar results, qualitatively speaking. The values chosen for experimentation in the following sections are largely arbitrarily, and we expect similar values to produce the same trends.

Our motivation for including quiescent and decay types was to encourage competition. First this system adopts some trivial ineffective strategies: a genotype spanning all available space transforming all cells in living cells and neither switching to quiescent would quickly disappear as these cells would turn into decay[1]. As illustrated in Figure 3.4 effective and sustainable survival strategies require the genotype to free space by converting some cells into quiescent cells.

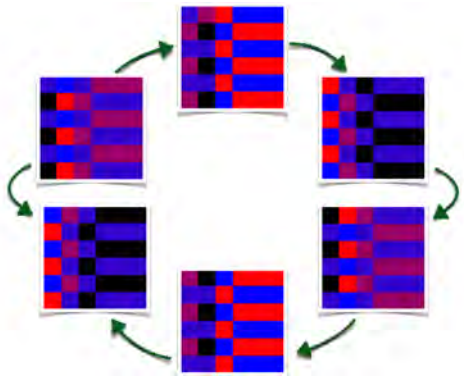


Figure 3.4: Six steps survival strategy from one possible phenotype of a genotype extracted at iteration 300000 in HetCA, tested here in a randomly initialized homogenous CA. This phenotype does not provide cells with the opportunity to grow old enough to decay before an evolutionary step changes them into quiescent cells. In doing so the cells lose their own genotype to facilitate the survival of the genotypes of neighboring cells. The meaning of the colour coding is described in Figure 3.1.

Secondly, we expected genomes to "learn" to use decay as a barrier for their "territory", which would allocate them space to grow but block genetic copy from neighbouring cells. Indeed, in many runs we have observed periods where populations selected voluntary decay with high probability. Since voluntary decay is a form of "genetic suicide", we view this as evidence that indeed, there are evolvable means of exploiting decay for survival advantage. Figure 3.5 shows one such successful example of a species relying on voluntary decay.

Moreover, some emergent properties of HetCA are similar to two of the five major eukaryotic innovations which do not appear to have direct prokaryotic pre-

[1]When they all have reached the maximum number of iterations alive before a cell goes decay.

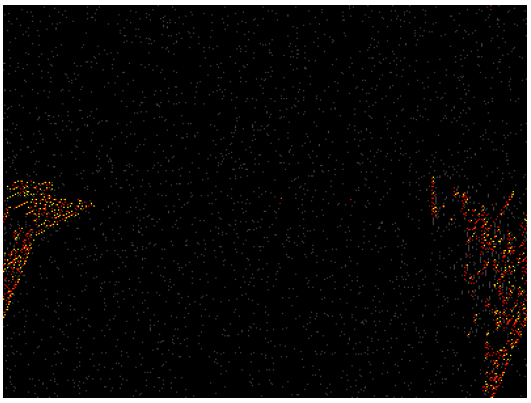
decessors defined in [Smith and Szathmary, 1997] as: (1) the eukaryotic chromatin remodelling machinery; (2) the cell cycle regulation systems; (3) the nuclear envelope; (4) the cytoskeleton; and (5) the apoptosis apparatus [Koonin and Aravind, 2002]. Indeed, in HetCA, the loss of the genotype of the cell as it turns to the quiescent state may seem similar to apoptosis while survival strategies such as the ones depicted in Figure 6.2 might akin cell cycle regulation systems.

3.3.1 Phenotypic Diversity

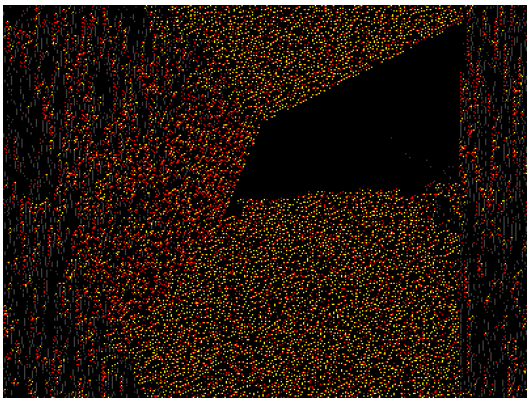
To quantify the qualitative results seen above, we designed a series of *measures of the diversity* of the system. The notion of "genotypic distance" was rejected, however: given that most transition functions φ encounter only a small proportion of all potentially available neighbourhood states, any representation of φ inevitably contains a lot of meaningless information, which would add too much noise to a direct comparison. This difficulty in measuring the distance between genetic programs, or any computer program for that matter, is well known. Therefore, we chose to focus on measuring the *differences in phenotypic patterns* over time, indirectly measuring the arrival of new "species" via the organizational behaviour that they exhibited in the world.

Choice of a phenotypic diversity measure Our goal is to automatically measure the phenotypic patterns associated with the speciation events observed above. This would allow us to detect these events and estimate the number of *species* in our world, keeping in mind that genomically distinct cells can be phenotypically indistinguishable from each other. Formally, a "species" here refers to a family of genomes $\{\varphi_{c,t}\}$ which have identical phenotypic effect in the world (and, naturally, should not be confused with a cell state $s \in \Sigma$, which any cell of any species may potentially assume). Thus, we want to define a metric that shows maximal variance for the events that we consider significant in our qualitative observations.

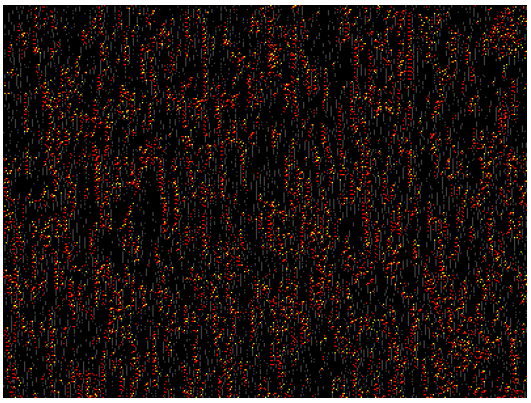
The simplest metrics involve the *number of cells* in each state (i.e. each colour in Figure 3.5) at each time step. We deal here with five different living states (Figure 3.6) and denote these metrics by $\{N_i(t)\}$ with $i = -1, 0, 1, ..., 5$, where



$t = 1350$: Two distinct genomes have generated two colonies. One is sparse, making prodigious use of voluntary decay; the second, less sparse, grows more quickly.



$t = 1950$: The two species have percolated, with the less sparse species occupying more space.



$t = 3300$: The sparse species has eliminated the other species entirely.

Figure 3.5: View of a HetCA run *(best viewed in colour)*. Quiescent cells are drawn in black, decay cells in grey, and living cells are drawn in other colours based on state.

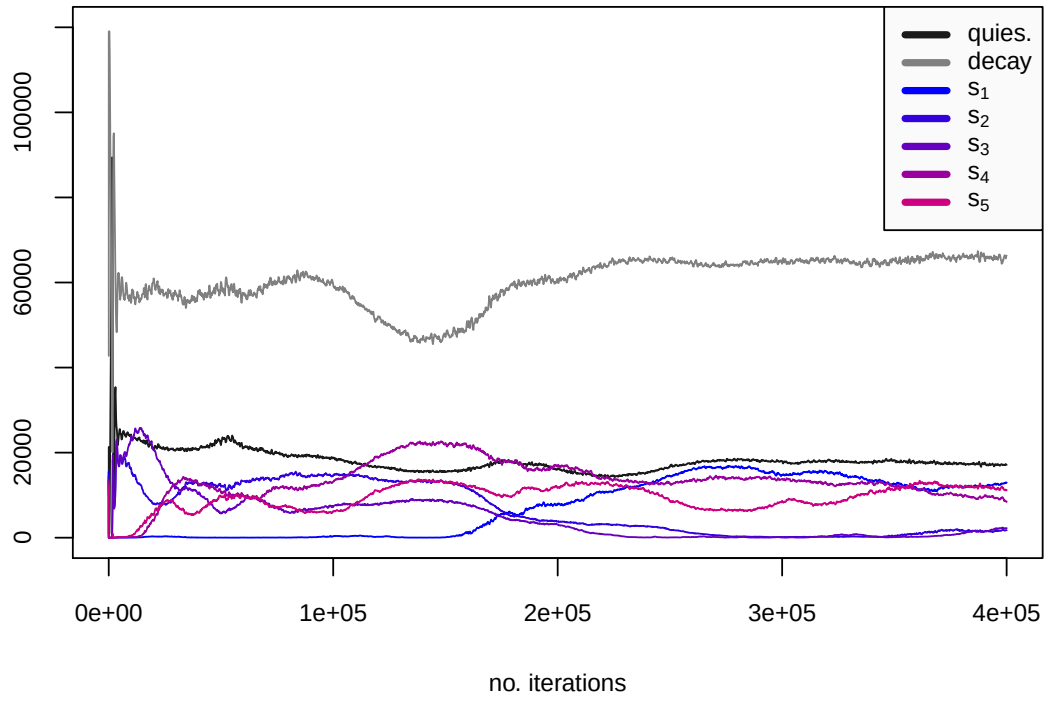


Figure 3.6: Plot of the number of cells by state over time in a typical HetCA run. The steep increases and decreases in the early iterations correspond to initial periods of rapid growth and extinction.

$i = -1$ and 0 represent states q and d respectively (thus here, $|\Sigma| = 7$ and $|\Lambda| = 5$). However, while these time variations in "isostate" population sizes (on the long term) may constitute some indication of genotypic changes in the world, hence changes in species, it is at best highly indirect (again, states are *not* genotypes). We cannot exclude the possibility that several distinct species generate similar proportions of cell states. In fact, given the incremental nature of evolution, this possibility is very likely.

Our first attempt at a better diversity measure is based on *entropy*. We define the spatial frequency of a given state $s_i \in \Sigma$ as $\rho(s_i) = N_i/|W|$, and define the global entropy by

$$H(t) = \frac{1}{\log |\Sigma|} \sum_{i=-1}^{|\Lambda|} -\rho(s_i) \log \rho(s_i). \tag{3.4}$$

Next, we measure the variance of the distribution of cell states. First, we determine the "median" cell state s_{med}, i.e. the state whose frequency of occurrence is median. From this, we write the "gross" cell variance in the world as:

$$\sigma_{\text{gross}} = \frac{1}{\sqrt{|W|}} \left(\sum_{c \in W} \big(1 - \delta(s(c), s_{\text{med}})\big) \right)^{\frac{1}{2}} = \sqrt{1 - \rho(s_{\text{med}})}, \tag{3.5}$$

where $\delta(s_i, s_j)$ is the Kronecker delta [Hotelling, 1933] between two states. We also tried versions of σ_{gross} involving only living cells (i.e. $i = 1, ..., |\Lambda|$) but without significantly different results.

To measure the local organization of cell states, we would like a continuous metric showing the divergence of a particular neighbourhood from the "typical" neighbourhood. However, our cell states are distinct types with no natural ordering, making it difficult to define a "state distance". Initial attempts using δ were not particularly discriminating. Instead, we based our metric on the frequencies of cell states.

First, we compute $s_{\max}, s_{\min} \in \Sigma$, the states which occur with maximum and minimum frequency, respectively. We define the normalized frequency of each state $s_i \in \Sigma$ to be:

$$\widehat{\rho}(s_i) = \frac{\rho(s_i) - \rho(s_{\min})}{\rho(s_{\max}) - \rho(s_{\min})} = \frac{N_i - N_{\min}}{N_{\max} - N_{\min}}. \tag{3.6}$$

Next, for each cell $c \in W$ we compute the *local state frequency mean* and *local state frequency variance* as

$$\mu_{\text{loc}}(c) = \frac{1}{9} \sum_{c' \in \nu_{\text{M}}(c)} \widehat{\rho}(s(c')) \text{ and} \tag{3.7}$$

$$\sigma_{\text{loc}}(c) = \frac{1}{3} \left(\sum_{c' \in \nu_{\text{M}}(c)} \left(\widehat{\rho}(s(c')) - \mu_{\text{loc}}(c) \right)^2 \right)^{\frac{1}{2}}. \tag{3.8}$$

This last measure allows us to quantify how different a particular local neighbourhood is from the expected cell states in a more continuous way. Using σ_{loc}, we can now define the *global mean* and *global variance* (both of the local state frequency variance) over the world as

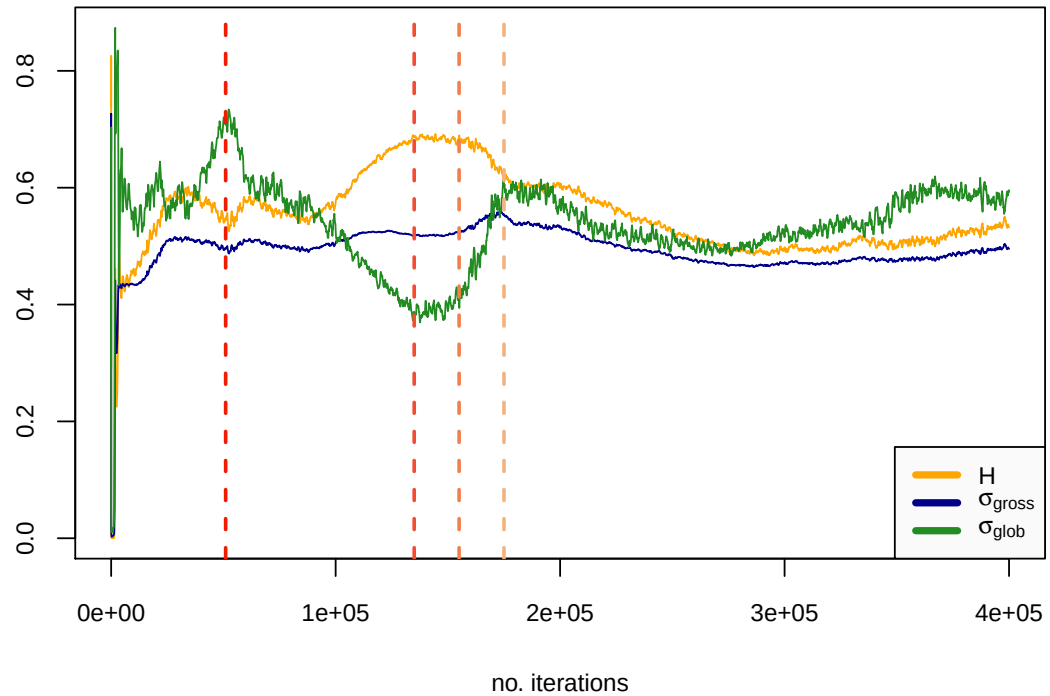
$$\mu_{\text{glob}} = \frac{1}{|W|} \sum_{c \in W} \sigma_{\text{loc}}(c) \text{ and} \tag{3.9}$$

$$\sigma_{\text{glob}} = \frac{1}{\sqrt{|W|}} \left(\sum_{c \in W} \left(\sigma_{\text{loc}}(c) - \mu_{\text{glob}} \right)^2 \right)^{\frac{1}{2}}. \tag{3.10}$$

The global measure σ_{glob} is an indication of the degree of "distinctiveness" of the neighbourhoods of the world relative to the expected neighbourhood. Thus it is sensitive not only to differing proportions of cell states over space, but also to the local structure of those cell states.

Comparison of model versions To demonstrate the capacity of HetCA to generate long-term phenotypic diversity, we contrast it with several control groups:

- **HetCA-a4**: HetCA with $a_{\text{max}} = 4$ and $a_{\text{dec}} = 1850$.
- **HetCA-a7**: HetCA with $a_{\text{max}} = 7$ and $a_{\text{dec}} = 1850$.
- **HetCA-noDec**: HetCA with $a_{\text{max}} = 4$ and $a_{\text{dec}} = 0$.
- **HetCA-noMut**: HetCA-a4 with $p_{\text{mut}} = 0$, i.e. all genomes remain as initialized.
- **ClassicCA**: a randomly initialized classical CA, in which the universal transition function φ was also generated randomly (by CA-LGP).



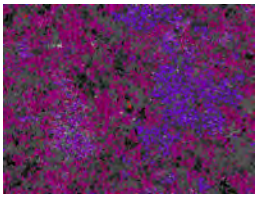


Figure 3.7: $t = 51,000$
Several competing species: high values for σ_{glob}.

Figure 3.8: $t = 135,000$
One species dominating: low values for σ_{glob}.

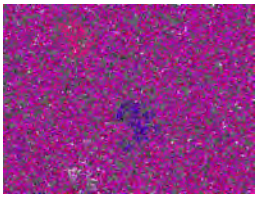


Figure 3.9: $t = 155,000$
Growth of new species (small blue cluster near centre, small pink cluster in upper left): σ_{glob} values increasing.

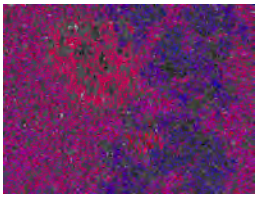


Figure 3.10: $t = 175,000$
A new mix of several competing species: high σ_{glob} values.

Figure 3.11: Correlation between phenotypic events and diversity measures in a representative run: (*top*) a plot of our proposed measures over time; (*below*) screen shots of particular iterations corresponding to significant events.

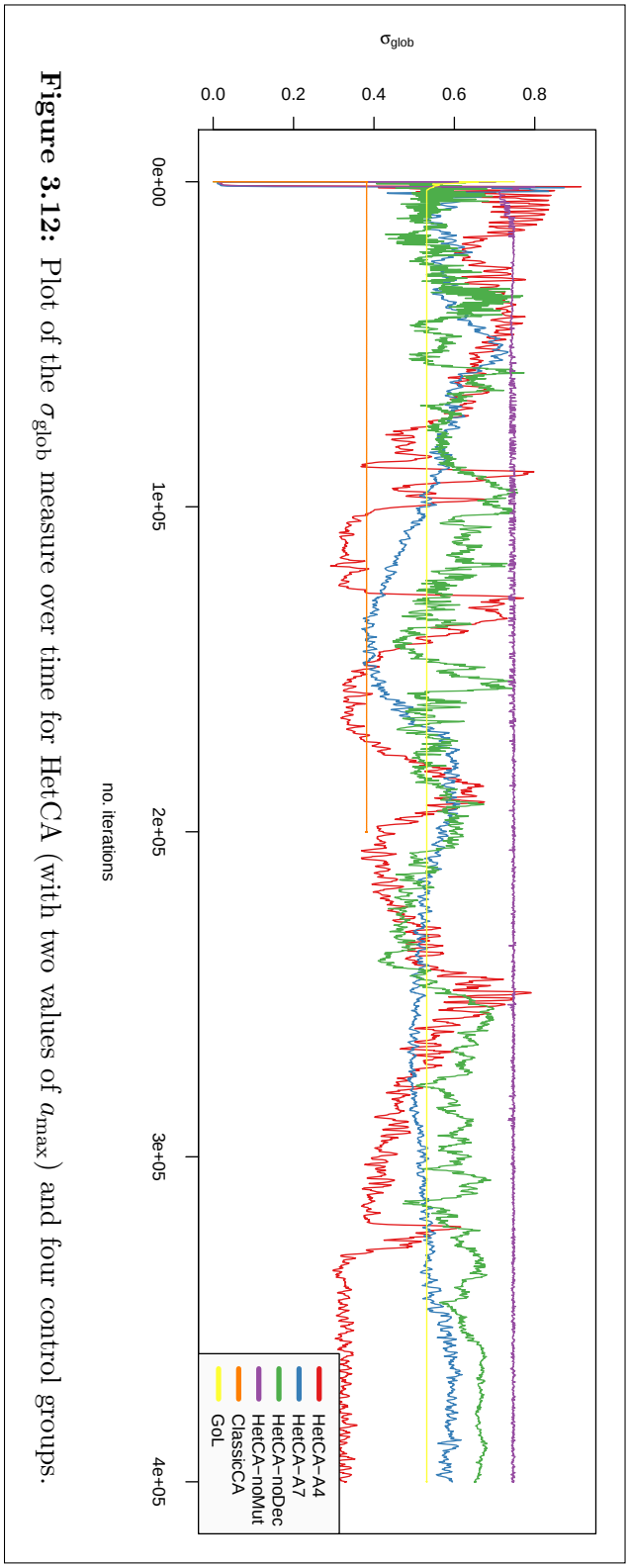


Figure 3.12: Plot of the σ_{glob} measure over time for HetCA (with two values of $a_{\max}$) and four control groups.

- **GoL**: the classic Game of Life in a randomly initialized world and with a binary alphabet $\Sigma = \{0, 1\}$.

First, to develop our understanding, we show some typical evolution graphs for the σ_{glob} metric for each of the above groups (Figure 3.12). In these particular runs, we see that HetCA-noMut, ClassicCA, and GoL are all nearly trivial, in the sense that there are no significant phenotypic events happening in any of them. This was expected and contributes to validating our measure of diversity: since these three controls are all non-evolutionary, they are likely to produce homogeneous behaviour, which is reflected by a flat diversity curve. In contrast, both HetCA curves (red and blue) show very substantive changes over time, while also displaying (noisy) plateaus during certain intervals. The HetCA-noDec curve (green) also shows significant changes over time.

These three non-evolutionary models are theoretically capable of universal behaviour, thus in principle can generate patterns that would maximize our measure. Yet, our intuition is that these sorts of configurations are very rare. Interestingly, the ClassicCA model generates behaviour that might be characterized as "chaotic" by CA standards (i.e. CA class IIIs) [Wolfram, 2002]. Yet, these cases generally do not register as "diverse" by our metric because they typically involve the repetition of similar regions across space.

HetCA-noDec, which is an evolutionary group, generates more interesting results. In this case, the curve is largely random, increasing and decreasing without any apparent pattern. Our hypothesis is that this behaviour is due to a lack of competition, i.e. genomes have little means to defend themselves against

competitors, hence the populations fluctuate randomly.

Parameter	Value
Number of Living states	5
Successive living iterations before decay	7
Number of iterations for decay	375-1875
Direct transition to decay	enabled
Size of the grid	400x300
Grid boundaries	toric grid
Transition Rule (TR)	CA-LGP
Maximum (TR) size	50 program statements
Genotype copy neighboring	von Neumann[a]
Transition rule neighboring	Moore[a]

[a] Using two different types of neighbors allows cells to "see" certain other cells before they can propagate their genotypes to them, thus potentially allowing the cells to "anticipate" contact with a hostile genotype.

Table 3.2: HetCA parameters for EP simulations.

The HetCA-a4 and a7 runs, however, appear to generate larger plateaus with dramatic changes interspersed, the sort of phenomena that could be associated with "punctuated equilibria" [Gould and Eldredge, 1977] of evolutionary innovation.

After looking at these graphs, we now want to compare these groups more quantitatively. To this aim, we define a single-value statistic, the *phenotypic variance*, as follows:

$$V_T[\sigma] = \frac{1}{\sqrt{T}} \left(\sum_{t=0}^{T-1} \big(\sigma(t) - E_T[\sigma]\big)^2 \right)^{\frac{1}{2}}, \tag{3.11}$$

where σ stands for σ_{glob}, T is some total time of observation, and $E_T[\sigma]$ is the mean of $\sigma(t)$ over time, written:

$$E_T[\sigma] = \frac{1}{T} \sum_{t=0}^{T-1} \sigma(t) \tag{3.12}$$

Note that $V_T[\sigma]$ is the variance *over time* of a measure that is itself an instantaneous variance *over space*. Next, we report the experimental *average phenotypic*

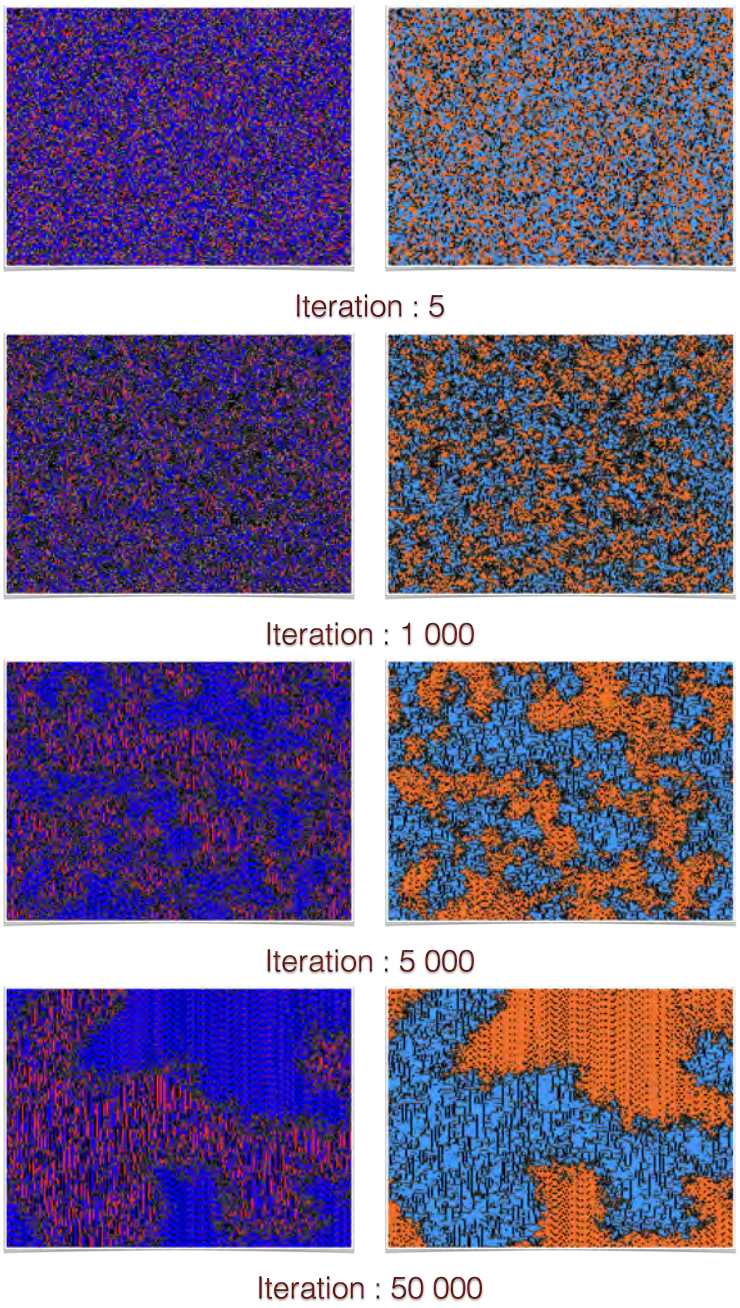


Figure 3.13: Formation of cells clusters in a robustness test. On the right column the cells are depicted with their current states (color coding is described in Figure 3.1), on the left column the repartition of the two genotypes is depicted. Cells that don't curently have a genotype, are represented with their current states on both sides.

variance (APV) $\overline{V_T}[\sigma]$ for each control group calculated over 100 independent runs based on random initial conditions. Figure 3.16 shows for each group a compact boxplot representing the distribution of $V_T[\sigma]$ values over different runs and their average $\overline{V_T}[\sigma]$.

3.3.2 Evolutionary Progress

In this section we analyze three genotypic properties as potential indicators of evolutionary progress: robustness, size and density.

Collection of genotypes In order to assess the existence of EP we created a collection of genotypes at various stages of the evolutionary process, in the following manner: We performed 30 simulations, each on 500000 iterations with the parameters listed in Table 3.2. The possible genotypes of an individual are its transition rules encoded with CA-LGP using the function set depicted in Table 3.1. For each simulation we saved the *most common genotype* (most frequently occurring) in iterations 5, 1000, 5000, 50000 and 500000. We have chosen to use iterations 5, 1000, 5000 and 50000 as they correspond to four distinct stages of the typical evolutionary process in HetCA and are characterized by four very differents environments as shown in Figure 3.14. Iteration 500000 was chosen because it is the final iteration of studied simulations, and iteration 300000 was selected as an intermediate step between iteration 50000 and iteration 500000. The choice of the most common genotype may seem arbitrary, but it was not realistic to process all genotypes at each iteration of the simulation whereas the frequency is a naive, but nonetheless efficient criterion for assessing its representativeness and its success at any stage of the simulation.

Evaluation of genotype robustness To assess the robustness of an individual we measure its ability to survive in different environments. One could compare this measurement with the definition of evolutionary progress proposed in [Dawkins, 1997]: "a tendency for lineages to improve cumulatively their adaptive fit to their particular way of life, by increasing the numbers of features which combine together in adaptive complexes".

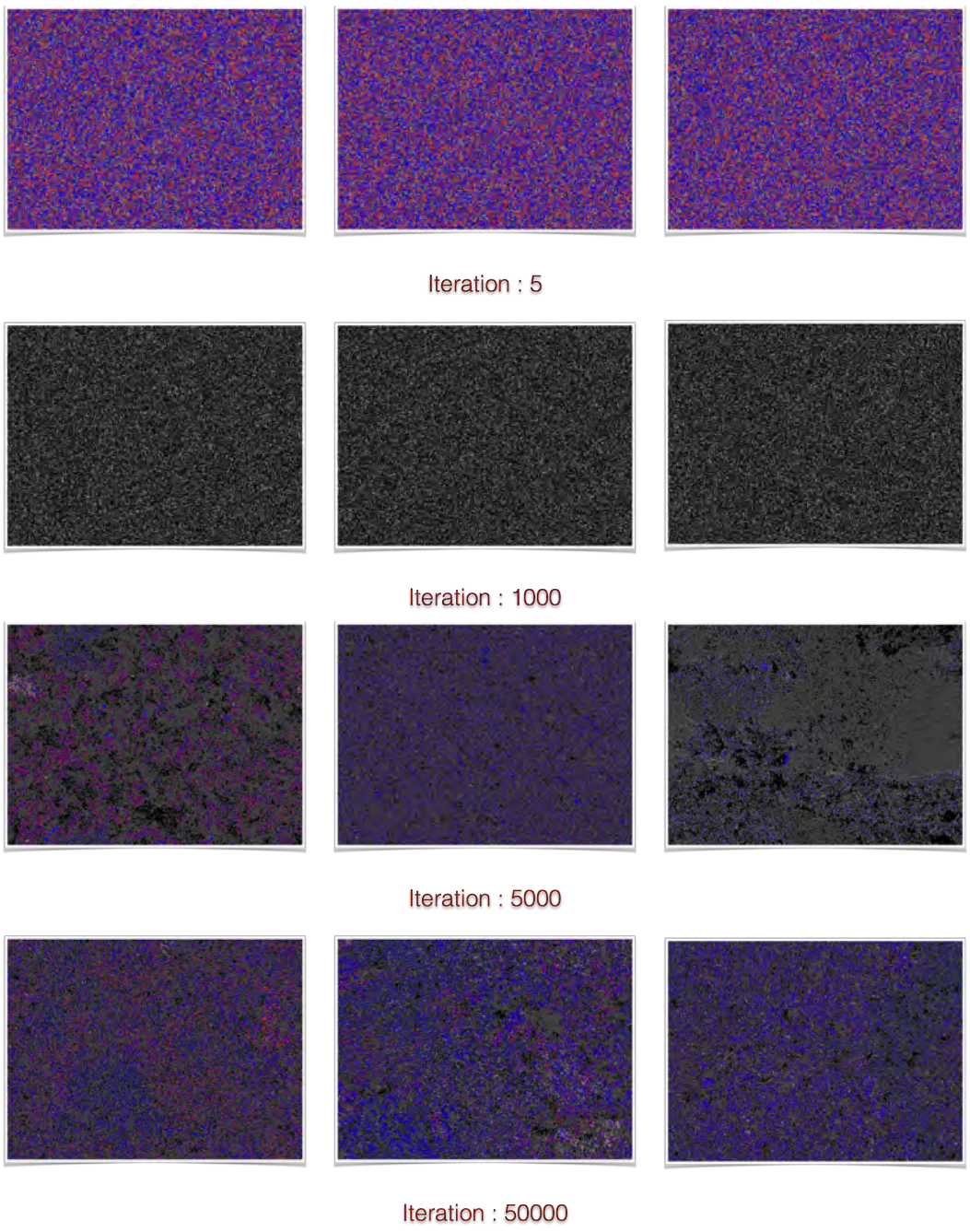


Figure 3.14: Typical HetCA grids at iterations 5, 1000, 5000 and 5000. Living cells are very frequent at iteration 5; at iteration 1000 most cells are in decay or in a quiescent state; and from there the population of living cells increases until iteration 50000. Color coding is described in Figure 3.1.

The robustness of an individual is measured by comparing genotypes in a pairwise fashion in a series of simulations where we disable mutations. We compare each genotype to every other genotype saved from a different run after a *different* number of iterations. We don't compare genotypes from the same runs because they have already competed in their evolutionary history, and making these comparisons could potentially skew results. Nor do we compare two genotypes collected after the same number of iterations of the cellular automata because this not would provide any information about EP[1].

Half of the cells, randomly selected, are initialized with the first genotype, the other half with the second one. All cells are initialized at a random living state. The simulation is stopped either after 50000 iterations[2], as illustrated in Figure 3.13, or when more than 99% of the living cells share the same genotype. After the simulation is stopped, the most frequent genotype is considered to be more robust than the other one due to its dominance of the environment. To conduct this experiment in a reasonable amount of time we choose to use only genotypes from the first 10 simulations, which represents $6 \times 10 = 60$ genotypes and $45 \times 60/2 = 1350$ simulations.

Evaluation of phenotype densities The density of each genotype is evaluated by a simulation where mutations are disabled and each cell is initialized at a random living state with the tested genotype as transition rule, as illustrated in Figure 3.15. The simulation is run for 2000 iterations. The density measure ρ of a genotype is the average number of living cells during the simulation:

$$\rho = \frac{\sum_{i=1}^{2000}(\rho_i)}{2000} \times 100 \quad and \quad \rho_i = \frac{n_{alive}}{S} \tag{3.13}$$

Where ρ_i is the density of the phenotype[3] for the iteration i, n_{alive} is the

[1]This may be interesting, however, to determine whether individuals produced by certain simulations are consistently better or worse than others and to determine the extent to which an individual's characteristics are correlated with those of its encounters.

[2]Long simulations are computationally expensive, they can take several weeks, the limit of 50000 iterations was chosen after informal tests indicating that this one was rarely exceeded and in cases where it was, the trend, in terms of majority genotype, was never reversed.

[3]We call here phenotype, the pattern drawn by the states of cells sharing the same genotype.

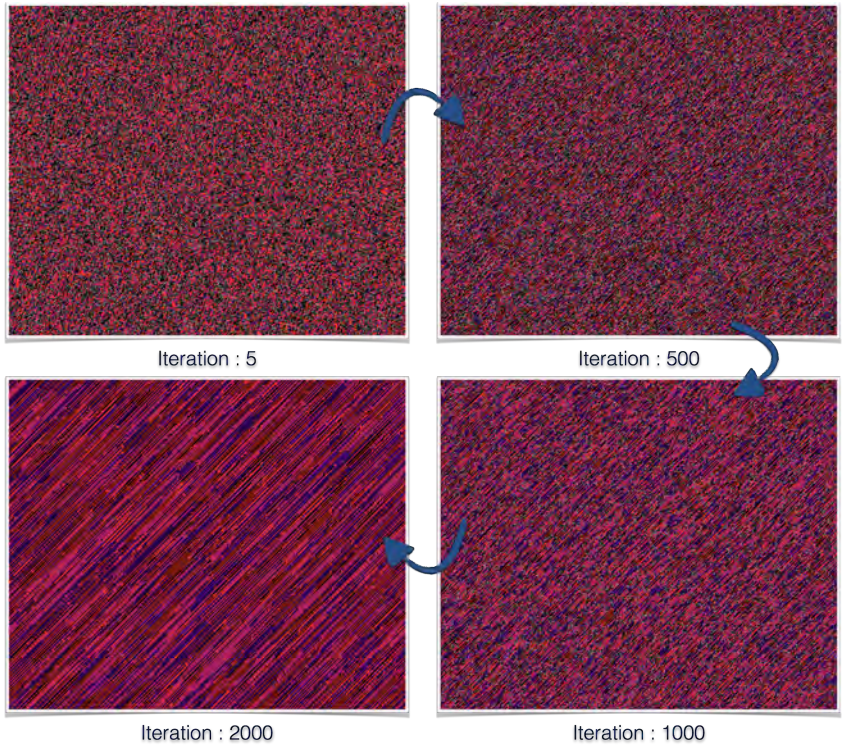


Figure 3.15: Density test. At each iteration the density rho_i is measured as the proportion of living cells. It is the proportion of living cells among all the cells. (color coding is described in Figure 3.1)

number of cells which current state is one of the alive states[1] and S is the size of the grid of the cellular automata[2]. The phenotypic densities of all the $30 \times 6 = 180$ genotypes are computed.

Evaluation of genotype sizes In evolutionary biology, according to [Lynch and Conery, 2003] the increase of size of individual genotype is caused by genetic drift and linked to population size (N_e)[3]. Similarly, in evolutionary computation size is frequently studied, if only because of the potentially high computational cost associated and a potential correlation with overfitting [Fitzgerald and Ryan, 2014]. More specifically, in evolutionary algorithms (EA), different studies show that size is not correlated with the ability of individuals to solve the presented task. We use the number of program statements (n_{prog}) as a measure of the genotype size. Sizes of all the $30 \times 6 = 180$ genotypes are computed.

Note that two of the three studied traits[4] are bounded and directly measurable while robustness is a relative criterion.

3.4 Results

3.4.1 Phenotypic Diversity

Both the ClassicCA and GoL groups have consistently low average phenotype variance. The HetCA-noMut group begins with high a phenotypic variance, but then decreases with time. This is likely due to an initial period of conflict between the randomly generated genomes, which later diminishes in intensity following the initial growth and extinction events. Thus, as we expected, there is little long-term variance in any of these three groups.

The two control groups closest in their σ curves over time (Figure 3.11) are the HetCA-a7 and HetCA-noDecay groups (blue and green). This is also reflected in Figure 3.16 by their APV clearly greater than all the other groups. Furthermore,

[1] Alive states are those states that are neither Quiescent state nor Decay state.

[2] $300 \times 400 = 120000$ cells in those simulations.

[3] An increase in the size of the genotype corresponding to a reduction of the size of the population.

[4] Size of the genotype and density of its phenotype.

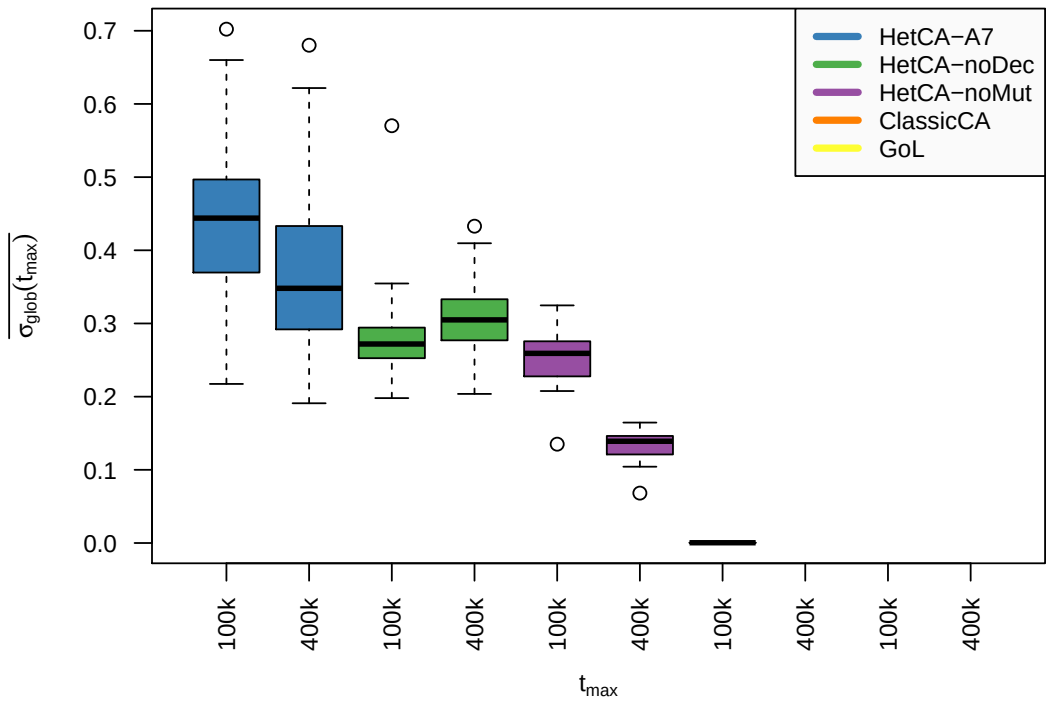


Figure 3.16: Comparison of values for $V_T[\sigma]$ by group, each for two durations $T = 100\text{k}$ and $T = 400\text{k}$.

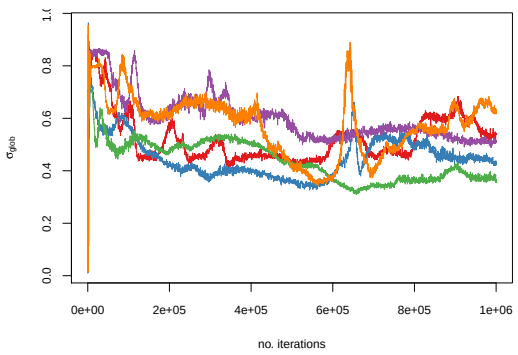


Figure 3.17: Plot of σ_{glob} over time for five independent long runs of HetCA-a7.

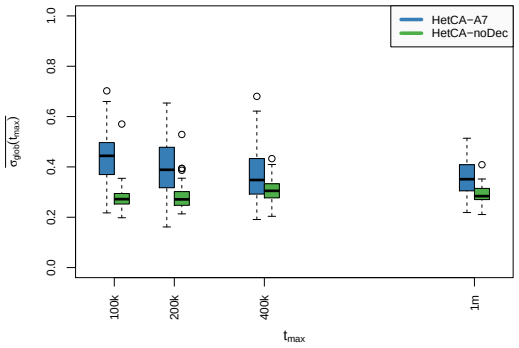


Figure 3.18: Comparison of values for $V_T[\sigma]$ by T for the HetCA-a7 and HetCA-noDecay groups.

HetCA-a7 is significantly higher than HetCA-noDecay (confirmed by a Welch's two-sample t-test yielding a P-value $p < 0.01$, even in the closer case of $T = 400$k time steps). On the other hand, however, it seems that the APV of HetCA-a7 is decreasing as the averaging window widens, while the APV of HetCA-noDecay is increasing.

To explore this further, several very long runs of $T = 1$M time steps were executed in both groups. In theory, given that the world is discrete and its states are finite, there must exist some point in time at which the dynamics of the world should loop back and become cyclic. However, due to the huge numbers of location-genome-state combinatorial possibilities, even these very long runs were not enough to reveal these types of cycles (Figure 3.17).

The phenotypic variance for the two groups is contrasted more precisely over different durations in Figure 3.18. Here, it is clear that the APV of HetCA-a7 initially decreases while that of HetCA-noDecay increases. By duration 400k, however, both values have plateaued. By duration 1M, there is no significant difference in phenotypic variance inside either group, compared to duration 400k. Still, at duration 1M, the phenotypic variance of HetCA-a7 is significantly greater than that of HetCA-noDecay (with P-value $p < 0.05$, where this slightly greater uncertainty is probably due to less data points on these long runs), this suggests that the decay promotes greater diversity over the long term.

3.4.2 Evolutionary Progress

Age	Robustness	Ending iteration
5	4%	295
1000	27%	3793
5000	34%	4873
50000	70%	4685
300000	80%	5157
500000	87%	5377

Table 3.3: Robustness of genotypes collected at different stages of evolution: Iteration is the number of iterations of the cellular automata after which collection of genotypes took place. The robustness is the proportion of the comparative tests where the tested genotype was the most prevalent. The final iteration is the average number of iterations before the comparative tests terminated.

Robustness In accordance with the hypothesis proposing the existence of evolutionary progress, Table 3.4 shows that during the pairwise comparison of the genotypes collected at different stages of the simulation, the oldest genotypes are frequently the most robust. The scores are higher than 89% in eleven out of fifteen cases, and even the smallest margin of 59% reported for the comparison of genotypes collected after 5000 and 1000 iterations, is the only non-significant using the binomial test at $p = 0.05$. Table 3.5 shows that not counting the tests that reach 50000 iterations slightly increases the win rate of the oldest genotypes. Not surprisingly, Table 3.3 indicates that robustness increases with increasing number of iterations. The inclusion or non-inclusion of simulations where no genotype had reached 99% dominance of living cells does not significantly impact the results, this shows that 50000 iterations are sufficient to perform this test.

	5	**1000**	**5000**	**50000**	**300000**	**500000**
5	-	347	339	375	34	384
1000	87%±7	-	11397	2586	2172	2465
5000	98%±2	59%±10	-	3962	3468	5203
50000	96%±4	97%±3	96%±4	-	8890	7612
300000	100%	100%	97%±3	68%±10	-	11224
500000	98%±2	95%±4	93%±5	76%±9	66%±10	-

Table 3.4: Pairwise comparison of the robustness of genotypes collected at different stages of evolution: The robustness of genotypes is shown at the bottom left of the table bellow the diagonal, whereas average final iterations are shown at the right above the diagonal. The robustness is the proportion of the comparative tests where the row genotype was more prevalent than the column genotype.

	5	**1000**	**5000**	**50000**	**300000**	**500000**
5	-	340	333	368	34	381
1000	87%±7	-	6015	2055	1646	2471
5000	98%±2	63%±10	-	2880	2411	3676
50000	96%±4	97%±3	96%±4	-	6955	6649
300000	100%	100%	98%±2	70%±10	-	7441
500000	98%±2	95%±4	94%±5	78%±9	67%±10	-

Table 3.5: Pairwise comparison of the robustness of genotypes collected at different stages of evolution without tests reaching 50000 iterations: The fields to the left and bottom of the diagonal detail robustness of genotypes whereas, those in the top right of the table diagonal show the average final iteration. The robustness is the proportion of the comparative tests where the row genotype was more prevalent than the column genotype.

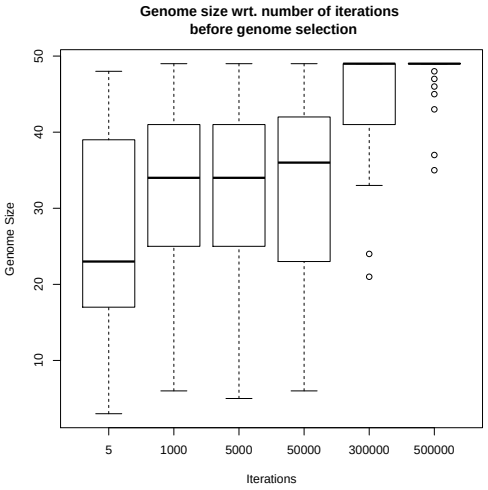


Figure 3.19: Size of genotypes collected at different stages of evolution: Iterations is the number of iterations of the cellular automata that occurred before the collection of the genotype, The size is expressed as the number of program statements (n_{prog}) used in genotypes.

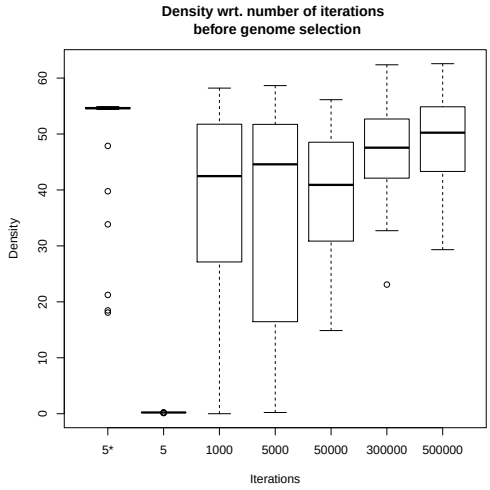


Figure 3.20: Density of genotypes collected at different stages of evolution. 5* is the density computed at iteration 5 before extinction of living cells.

Size Results in Figure 3.19 do not show significant differences between the size of the genotypes collected in iterations 1000, 5000 and 50000 using the Welsh test at p = 0.05. The genotypes collected at iterations 5 are significantly smaller while those taken at iterations 300000 and 500000 are significantly larger. The average size of the genotypes, selected in iterations 300000, is very close to the maximum of 50 and does not significantly increase at iteration 500000.

Density Figure 3.20 shows that phenotypic density is higher for genotypes collected later in the evolutionary process. However there was no significant difference, using the Welsh test at p = 0.05, between iterations 1000 and 5000. The difference in density, 47% against 48%, is very small between iterations 300000 and 500000 even though this is the second longest interval. It should be noted that, for genotypes extracted at iteration 5, live cells have been completely extinguished before the simulation reaches 2000 iterations. It has also occurred for two

genotypes extracted at iteration 1000 and one extracted at iteration 5000. The average density of these same genotypes collected at iteration 5 before extinction is 46% which would rank their density between those extracted at iterations 50000 and iterations 300000.

3.4.3 Qualitative analysis

The pairwise comparison of the genotypes collected also facilitates some qualitative remarks on the nature of the interactions observed in HetCA. As can it be seen in Figure 3.13, groups of cells sharing the same genotype are formed very quickly. This reinforces the hypothesis of cooperation between cells sharing the same genotype. Similarly, the emergence of cells in self-decay at the edge of two clusters is very visible in some simulationd as shown in Figure 3.21. By doing this, those cells do not release space for cells sharing the same genotype and lose their own genotype, so this behavior is very rare and most likely usually counter selected. A hypothesis explaining the selection of such a strategy would be the creation of a barrier of cells in decay to block the progress of a hostile genotype. It is also interesting to note that during robustness testing, the genotype taking advantage early in the simulation is not necessarily the one that will dominate over the long term. This is probably explained by the progressive construction of patterns[1], as illustrated in Figure 3.15, and it could make HetCA an interesting model of *open-ended evolutionary developmental biology*. This reinforces the hypothesis of the existence of complex strategies HetCA, and shows that the survival phenotype changes with development. Figure 3.22 shows an example of the diversity of evolved strategies where the density of the blue pictured genotype is very low but it appears to be efficiently competing against the genotype in orange by massively propagating inside an orange genotype cluster when contact occurs between these two genotypes.

[1] Attractors of the genotype.

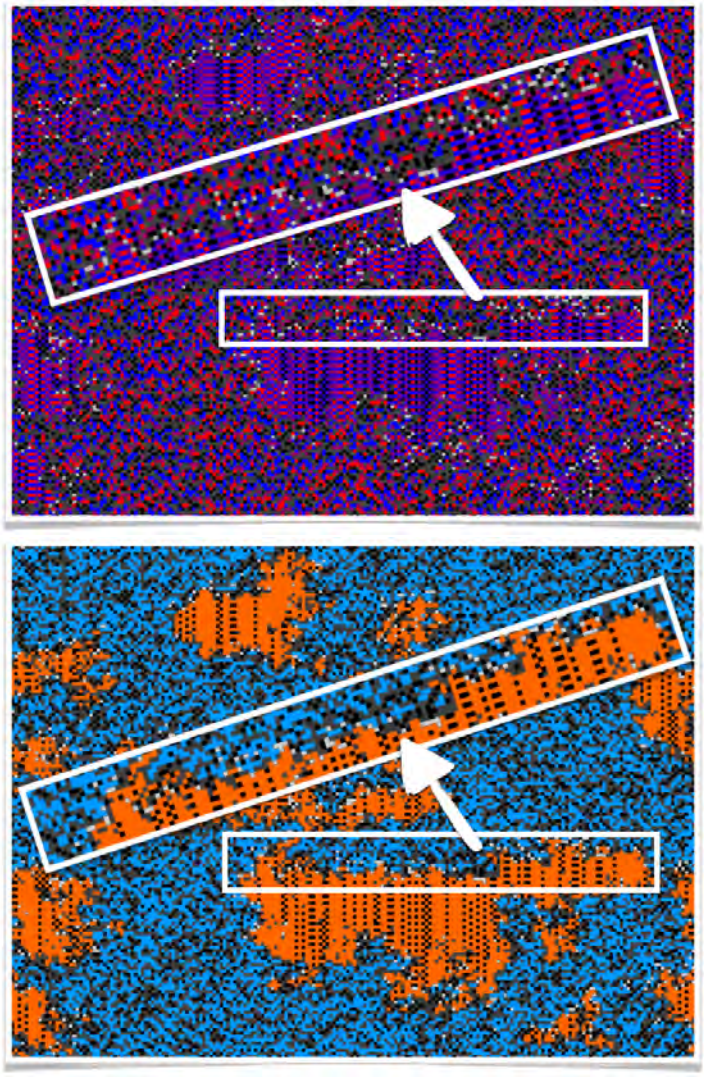


Figure 3.21: Altruistic decay: The light gray cells are cells in auto-decay, they appear here in areas contested by the two genotypes, helping to create a barrier between them. On the top image, cells are depicted with their current states (color coding is described in Figure 3.1), on the bottom image the repartition of the two genotypes is depicted. Cells that don't have a genotype (in decay or quiescent state) are represented with their current states on both sides.

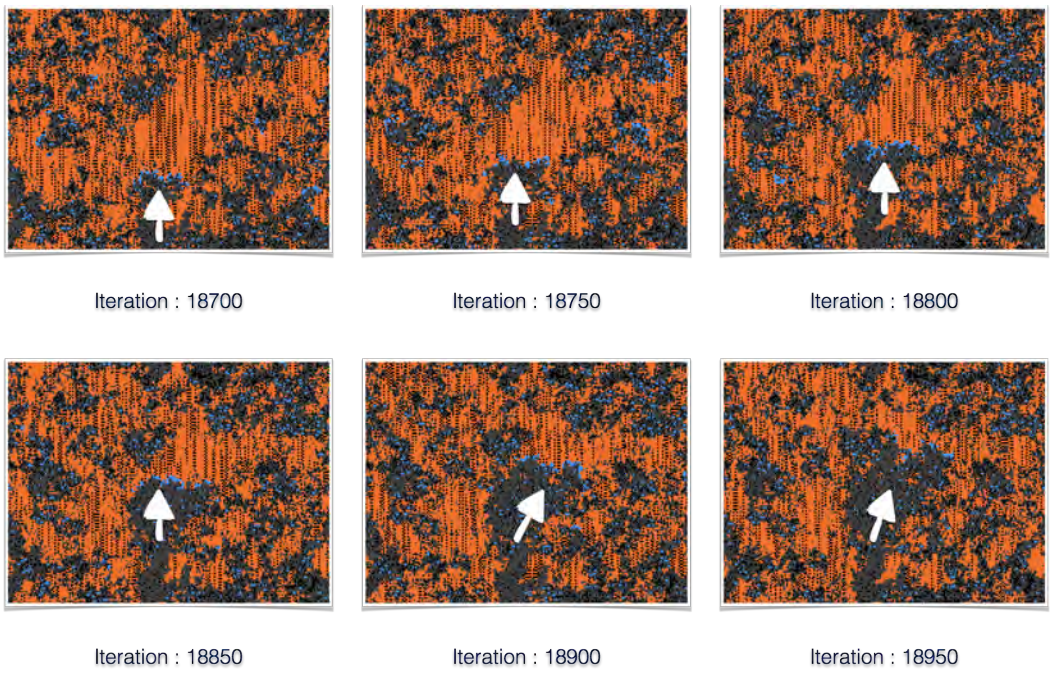


Figure 3.22: Evolutionary strategy: In these images we can see that cells with blue genotype, despite their low density, are effectively able to eliminate cells with the orange genotype when they are in contact.

3.5 Discussion

We have presented a simple extension of cellular automata, HetCA, consisting of three important changes from classical CA: transition function heterogeneity, transition function mutability, and cell decay and quiescence. Our hypothesis was that these changes would generate long-term dynamics. To test this, we developed a measure of phenotypic diversity correlating with our intuition regarding significant phenotypic events.

Results showed that our model *HetCA was capable of long-term phenotypic dynamics*, sustaining a high level of variance over very long runs. Moreover, *HetCA displayed greater behavioral diversity* than classical cellular automata, such as the Game of Life. Finally, comparison between model variants showed that *all three changes were instrumental* in the generation of this long-term diversity.

The measures of evolutionary progress that we use here are pretty tough compared to the definition proposed by [Shanahan, 2012]. Not only are the different genotypes collected within the same simulation not necessarily generated from a single evolutionary lineage, but we also collect individuals from different simulations and thus from independent evolutionary processes.

Nevertheless, at first sight we detect evolutionary progress in HetCA, between iteration 5 and 500000. There are several periods of stasis[1] during this process but the direction of progress is never reverses and therefore changes in the three traits studied here are directional. Weak or nonsignificant differences between the traits analyzed in iterations 1000 and 5000 could be explained by the fact that there are relatively few evolutionary steps between them. However, a significant difference exists between the genotypes selected at iteration 5 and those selected at iteration 1000 although the number of iterations between them is only 995 iterations. This is likely due to the differences between the typical environment in iterations 5 and 1000 as depicted in Figure 3.14. At iteration 5, the critical part of the selection is the ability to survive and reproduce as quickly as possible in the "primordial soup" of the early iterations of HetCA, whereas at iteration 1000 the greater part of the cells are in decay and selection is made on the ability of a small group of cells to survive without saturating a reduced space.

[1] For robustness and density.

The increase in cell density seems quite logical. The density of the phenotype being an obvious evolutionary advantage in HetCA. It is interesting to note that while there maybe periods of stasis in the increase of density, between iterations 1000 and 5000 the values for robustness continue to increase. This demonstrates that possible evolutionary strategies are not limited to density increase.

However, if the three measures used here show directional changes, small modifications of the experimental protocol can reverse this trend. Especially if the density measurement was stopped before the extinction of genotypes collected at iterations 5, in which case, they would have had a higher density than genotype collected at iteration 50000 as is shown in Figure 3.20. It is also quite difficult to assess whether the study period, 500000 iterations is sufficient to observe potential change in trend, and the EP seems to be slowing down in the last 200000 iterations.

The fast increase of genotype size between iteration 5 and 1000 could be the result of differences in size between random genotypes with which we initialize the simulation, those with viable strategies being maybe longer on average than others. We tested this hypothesis in the Table 3.6, it is checked but does not seem sufficient to explain this difference alone. An additional explication could be genetic drift, because as illustrated in Figure 3.14 the population, N_e, is very small during this interval.

	Average Size	**Proportion**
Effective survival strategy	29	0.5%
Ineffective survival strategy	25	99.5%

Table 3.6: Survival strategy and size: We randomly generated genotypes then tested, one by one, their ability to survive on 40×30 size grids (in simulations without mutations and where all cells are initialized with the same genotype). Genotypes extinct before iteration 5 have not been taken into account; Genotypes extinct before iteration 100 are considered as having ineffective strategy; genotypes still having living cells at iteration 100 are considered using an effective strategy. We performed this test on 100000 genotypes.

3.6 Conclusions

In this model the concept of phenotype is very close to the concept of environment: the phenotype of a genotype constituting literally the neighborhood and thus the environment of other cells whenever they share the same genotype or not. We can see, by the existence of long term evolutionary dynamic at the phenotypic level, that HetCA happens to generate new environements constantly, we think that this feature is not only a measure of its open-endedness but also creates a self-supply evolutionary pressure toward open-endedness. Furthermore we are interested in the ability of resistance to environmental changes of individuals based on their evolutionary history. Our robustness test involves testing each genotype in different environment consisting of other genotypes' phenotypes. The fact that this robustness actually increases with the "evolutionary age" of a genotype is interesting because, as we saw in Chapter 2, robustness to environmental change is often linked in contemporary evolutionary biology to central features of natural selection as it place for living organisms such as epigenetic mechanisms, plasticity or development.

4

Environmental Fluctuations in Genetic Programming with Interleaved GP

4.1 Introduction

In this chapter, to begin to study the benefits of environmental fluctuations in GP, we opt for a simple method, consisting of alternating between two fitness functions every other generation.

4.1.1 Random Interleaved Sampling

Similar methods have already been used in GP. *Random Interleaved Sampling* (RIS) [Goncalves and Silva, 2013] is a technique inspired by [Gonçalves et al., 2012] recently introduced by Gonclaves and Silva to reduce *overfitting* the training data in GP [Koza, 1992]. In this method GP alternates between using the entire training data set and just a *single* training data point in alternate generations to evaluate the fitness of the evolving individuals; the method chooses a different data point in each such generation that uses a single data point.

The motivation behind choosing a different training example for the single instance generations is to discourage overfitting, whereas the presentation of the entire training data set at every alternate generation still allows GP to learn

a good fit for the training data. The authors argue that this interplay between different objectives allows GP to learn models that generalise beyond the training data. We consider that this alternation between different methods of assessing the fitness of individuals is a kind of environmental fluctuations.

4.1.2 Two new types of interleaved

While varying the data sets during evolution is certainly useful if it reduces overfitting, it is also useful because it reduces the computational expense of a GP run: evaluating a large population of individuals on just a small subset of a much larger training set is significantly cheaper. This makes RIS particularly interesting because with this method half the generations only use a single data point; thus, the computational expense of GP reduces substantially with this technique. Even so, a question arises as to whether there is merit in using a single data point: after all, the information gathered from a single point is only minimal while still requiring a potentially expensive fitness evaluation[1]. Also, a related question is: in which extent the effectiveness of this method is it due to the one point sampling and in which extent is it due to the interleaving itself, and therefore the fluctuations themself? To answer the questions raised above, this chapter takes a two pronged approach. First, in order to ascertain the efficacy of using a single data point in interleaved generations, we compare the results with those from using only random search in the interleaved generations. This method, that we call *Interleaved-Random*, does not invoke the fitness evaluator at all in half the maximum number of generations which is an advantage; moreover, since random search is blind to the fitness landscape, an intriguing associated question is if it can help reduce code bloat and thus further reduce the computational expense. Thus, we believe that Interleaved-Random is a useful benchmark for Interleaved Sampling.

Next, instead of using random search, in a bid to reduce the computational expense even further, we specifically reduce tree size in the interleaved generations

[1]Sometimes merely invoking the fitness evaluator contributes significantly to the cost of a fitness evaluation, for example, when an external system call is required to evaluate the individuals.

using a method we term *Interleaved-Size*. As with the random approach, this method does not invoke the fitness evaluator in the interleaved generations, rather it preferentially selects *smaller* individuals.

4.1.3 Methodology and Structure

To ascertain the efficacy of those various approaches, we compare the results on training fitness, test fitness (performance on out of sample data), the disparity between the training fitness and the test fitness as a measure of overfitting and the average sizes of the evolving trees. We also use standard GP as a benchmark for this study. We report these results on four high-dimensional problems from the symbolic regression domain.

This chapter is organised as follows: Section 4.1.1 explains the exact implementation of RIS that we use in this chapter; Section 4.2 discusses the experimental setup and the problem suite used in this study, details the results and discusses their significance; Section 4.3 summarises the lessons that we draw from this investigation.

4.2 Experiments

Several configurations of RIS are presented in [Goncalves and Silva, 2013], each interleaving the generations in a different way. However, broadly, these configurations are termed *deterministic* or *probabilistic*. Indeterministic interleaving, the fitness function switches from using a full data set to the single point every generation in a deterministic fashion. In the probabilistic interleaving, however, a pre-specified probability prefers the single data point over the full training set (or vice versa).

For example, a probability of 50% approximates the behaviour of deterministic interleaving: on average, the generations using the full data set and the single point are evenly split. The results in [Goncalves and Silva, 2013] show that the best performers were deterministic interleaving, and probabilistic interleaving with 50% and 75% probabilities of selecting a single data point; the respective

performances of these setups were indistinct. Therefore, in this chapter we use deterministic interleaving as the baseline method.

Parameters	Values
Population Size	50, 250 and 500
Generations	200
Operator Probabilities	Xover: 0.9; Point mutation: 0.1
Tournament Size	10
Max. depth	17
Max. size	100
Replacement	generational
Functions set	$\{+, -, \times, /\}$
Terminal set	{Input variables} $\cup$ {Constants}
Constants	-1.0, -0.5, 0.0, 0.5 and 1.0
Fitness	$RMSE$
Initialisation	Ramped half & half (max. initial depth = 8)

Table 4.1: Configuration parameters for the GP runs.

We compare the performance of RIS with Interleaved-Random, Interleaved-Size and normal GP without RIS. In Interleaved-Size to reduce the size we choose to target an ideal size of 15 instead of 0; therefore, we subtract 15 from the size of an individual and take the absolute value of the result as the fitness of the individual. Thus, for example, an individual of size 14 grades the same as that with a size 16.

Although 15 is an *ad hoc* choice for this investigation, it is based on some initial experiments which showed that minimising the size to 0 generated poor results whereas 15 produced acceptable results.

Table 4.1 lists the values of various parameters for the various GP setups. In order to run the experiments to 200 generations, as in [Goncalves and Silva, 2013], in a reasonable time, we cap the maximum size to 100 for every setup. This is to counter excessive bloat, particularly in standard GP, that makes it impractical

to run to 200 generations. Again, 100 is an *ad hoc* choice, but as the results later show, it is a reasonable choice.

We note the following statistics to compare performance:

1. the median training fitness of the best individuals in the final generations. The best individual is the one with the minimum Root Mean Squared Error (RMSE) on the training data;

2. the median test fitness of the best individuals (best on training) in the final generations;

3. the median overfitting of the best individual in the final generation. As in [Goncalves and Silva, 2013], we measure overfitting as the absolute difference between the training and the test fitness of an individual;

4. the median of the median overfitting in the final generations;

5. and the average tree size of the evolving individuals as an indicator of the computational overhead of each setup.

We compute the statistical significance of the performance difference using the *Mann-Whitney U* test at $p = 0.05$.

Note that the population size can be an important factor in the performances of the various setups because this dictates the amount of time allocated to each interleaved generation: the larger the population size the more individuals each interleaved method processes before a *normal* fitness guided search resumes, thus affecting the future of evolutionary search in the concerned run. Therefore, we consider three population sizes for each setup: 50, 250 and 500. Except for Interleaved-Size which minimised size, running 200 generations, even with a maximum tree size of 100 took substantially longer for population sizes of 250 and 500 than that of 50. Therefore, we conducted 100 runs for every setup with the population size of 50 but only 50 runs for the greater population sizes.

4.2.1 Problem Suite

We consider four high-dimensional problems (from 7 to 241 input variables) from the domain of symbolic regression for this study.

The first problem, `Toxicity` is from the domain of pharmacokinetics. In this problem the objective is to predict the the amount of compound required to kill 50% of the considered test organisms. The data set contains 234 instances, each comprising of 627 features. This problem was used by [Goncalves and Silva, 2013] to validate the performance of RIS.

The second problem, `Bioavailability`, is in the same domain as the first problem and was also used in [Goncalves and Silva, 2013]. Here the objective is to predict the percentage of an orally submitted dose of a drug that effectively reaches the systemic blood circulation from 359 instances of 241 input variables. Both the first and the second problems were first tackled with GP in [Archetti et al., 2006].

In the third problem, `Concrete Strength`, the objective is to predict a quantitative value of compressive strength of concrete. This strength is a highly nonlinear function of eight input variables; the data set comprises of 1030 instances. The data source is UCI Machine Learning Repository [Frank and Asuncion, 2010] and the problem itself is detailed in [Yeh, 1998].

In the fourth and final problem, `Yacht`, the objective is to predict the hydrodynamic performance of sailing yachts from dimensions and velocity. The data is composed of 308 instances and has 7 input variables. Again, the data source is UCI Machine Learning Repository.

At the beginning of each run we randomly split the data set into two sets of identical size. One of them is used as the training data set (used to evolve our populations) and the other one is used as the testing data set.

4.2.2 Results

Results are outlined in Tables 4.2-4.6. In each table, `None`, `Rand`, `Size` and `1-pt` refer to normal GP without interleaved generations, Interleaved-Random, Interleaved-Size and RIS (using a single point in interleaved generations) respectively.

Population	50	250	500
	`Toxicity`		
None	**196.781**	**179.940**	**168.701**
Rand	**197.153**	186.165	177.431
Size	199.681	194.785	191.786
1-pt	202.686	188.050	182.915
	`Bioavailability`		
None	**3.12219**	**2.71019**	**2.51290**
Rand	**3.10574**	**2.72302**	**2.51203**
Size	3.31092	3.07110	3.01300
1-pt	3.79225	3.13761	3.01923
	`Concrete Strength`		
None	1.34963	**0.68349**	**0.59883**
Rand	1.24018*	**0.68985**	**0.62178**
Size	**1.13341**	0.99090	0.97943
1-pt	1.30121	0.74049	0.75160
	`Yacht`		
None	**0.85711**	**0.38853**	**0.33140**
Rand	**1.01708**	**0.36760**	**0.33415**
Size	1.19632	0.78604	0.68262
1-pt	1.34099	1.04762	0.65027

Table 4.2: Training Fitness of the best individuals in the final generation for each of the four problems. The best fitness are highlighted in bold face, those which are not significantly different are marked with an asterisk (*) as detailed in 4.2.2.

Population	50	250	500
	`Toxicity`		
None	204.748	199.402	209.679
Rand	**196.276**	192.208*	201.343
Size	206.139	**195.690**	**190.409**
1-pt	206.892	207.957	212.608
	`Bioavailability`		
None	**3.27848**	2.94743	2.88460
Rand	**3.26689**	**2.85854**	**2.70530**
Size	3.34733	3.17819	3.02524
1-pt	3.96089	3.34340	3.22562
	`Concrete Strength`		
None	1.32205	0.70387*	**0.60527**
Rand	1.27259	**0.68247**	**0.65586**
Size	1.13441	1.01797	0.98263
1-pt	1.30947	0.75832	0.77909
	`Yacht`		
None	**0.88911**	**0.42583**	**0.34942**
Rand	**1.00206**	**0.39604**	**0.33085**
Size	1.17339	0.83782	0.73834
1-pt	1.36252	1.02317	0.71299

Table 4.3: Testing Fitness of the best individuals in the final generation for each of the four problems. The best fitness are highlighted in bold face, those which are not significantly different are marked with an asterisk (*) as detailed in 4.2.2.

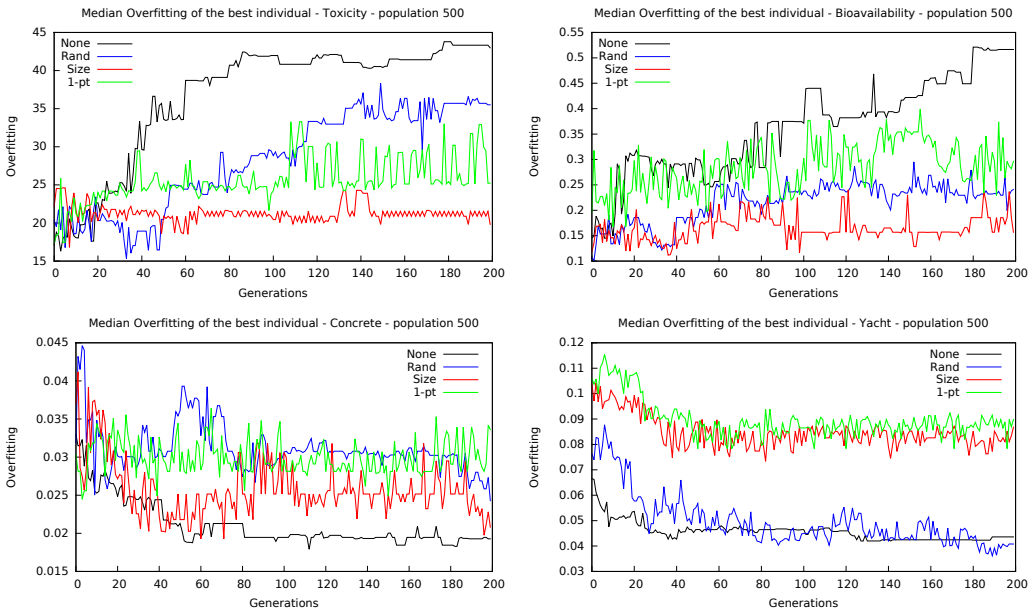


Figure 4.1: Median of Overfiting of best individual: per generation for all the problems.

Statistically significant results are highlighted as follows:

- The numbers highlighted in **bold face** are statistically significantly different from the other numbers in that column. If two or more numbers are significantly different from the rest while not being different mutually, then all such numbers are given in the bold face.

- Sometimes a number may be significantly superior to only some of the other numbers in that column but not to all. In this case, the superior number is highlighted in bold face as above but those which are not significantly different are marked with an asterisk (*).

4.2.2.1 Training Fitness

Table 4.2 gives the training performance of the best individuals from all the setups. As expected, normal GP is usually the best or amongst the best performers

on this statistic. This can be attributed to continuous exposure to the entire data set. However, rather surprisingly, Interleaved-Random also performs just as well as normal GP. The performance of each of the setups appears to improve with the increasing population size; however, we are not highlighting the significance of difference across rows.

Sometimes a number may be significantly superior to only some of the other numbers in that column but not to all. In this case, the superior number is highlighted in bold face as above but those which are not significantly different are marked with an asterisk (*).

4.2.2.2 Testing Fitness

Table 4.3 gives the results on the test sets. In this instance we can not see a *single* consistent winner as might be expected from the Interleaved setups. However, interleaved setups in *some* form are among the best performers throughout. Moreover, as with training results, there appears to be a trend of improving test set performance as we increase the population size from 50 to 500. Also note that Interleaved-Size and Interleaved Random perform *at least* as well as RIS.

4.2.2.3 Overfitting

Table 4.4 estimates overfitting of the best trained individuals. This is a key statistic as it gives an estimate of the predictive accuracy of the best trained individuals by measuring the disparity in training and test set performances. In this case, normal GP performs the best only on the `Yacht` problem. Rather surprisingly, Interleaved-Sampling does not consistently improve over normal GP. However, as in Table 4.3, interleaved methods in one form or the other perform at least as well as standard GP.

An important result is that despite minimising size, Interleaved-Size performs at least as well as the more *informed* Interleaved-Sampling. This is also the case with the training and test fitnesses. The results do not indicate a clear effect of changing the population size on overfitting though.

Table 4.5 gives the overfitting of the median individuals in the final generation. Again, we draw the same conclusions as from Table 4.4: interleaved methods in

Population	50	250	500
	`Toxicity`		
None	22.6673	24.1284	43.325
Rand	20.2454	20.9253*	**34.284**
Size	22.0368	**20.2185**	**19.824**
1-pt	22.4657	20.3742	**25.227**
	`Bioavailability`		
None	0.16730	0.22574	0.44006
Rand	0.14764	**0.17124**	0.22994*
Size	0.13925	**0.17914**	**0.11040**
1-pt	0.16503	0.19926	0.29134
	`Concrete Strength`		
None	0.03018	0.02215	**0.01844**
Rand	0.02720	0.02121	0.02384*
Size	0.03169	0.02821	0.02082*
1-pt	0.03095	0.02789	0.03361
	`Yacht`		
None	**0.08174**	**0.05788**	**0.04359**
Rand	**0.08616**	**0.06225**	**0.04078**
Size	**0.09642**	**0.06370**	0.08677
1-pt	0.14529	0.09217	0.09136

Table 4.4: Median Of Overfitting of Best Individuals of the best individuals in the final generation for each of the four problems. The lowest overfittings are highlighted in bold face, those which are not significantly different are marked with an asterisk (*) as detailed in 4.2.2.

Population	50	250	500
	`Toxicity`		
None	22.6673	23.7266	43.32
Rand	20.2454	21.8396*	**30.28**
Size	22.0368	**20.4898**	**23.16**
1-pt	22.4657	20.9969	**28.693**
	`Bioavailability`		
None	0.16373	0.22416	0.44006
Rand	0.14940	**0.15817**	0.22524
Size	0.16204	**0.15172**	**0.15673**
1-pt	0.16335	0.19675	0.26887
	`Concrete Strength`		
None	0.03317	0.03109	**0.01844**
Rand	0.02737	0.02332	0.02907*
Size	0.02988	0.02581	0.02494*
1-pt	0.03650	0.02976	0.02928
	`Yacht`		
None	**0.08174**	**0.05938**	**0.04400**
Rand	**0.07978**	**0.07057**	**0.04955**
Size	**0.09273**	0.08526	0.09428
1-pt	0.14088	0.09879	0.09433

Table 4.5: Median of median Overfitting in the final generation for each of the four problems. The lowest overfitting are highlighted in bold face, those which are not significantly different are marked with an asterisk (*) as detailed in 4.2.2.

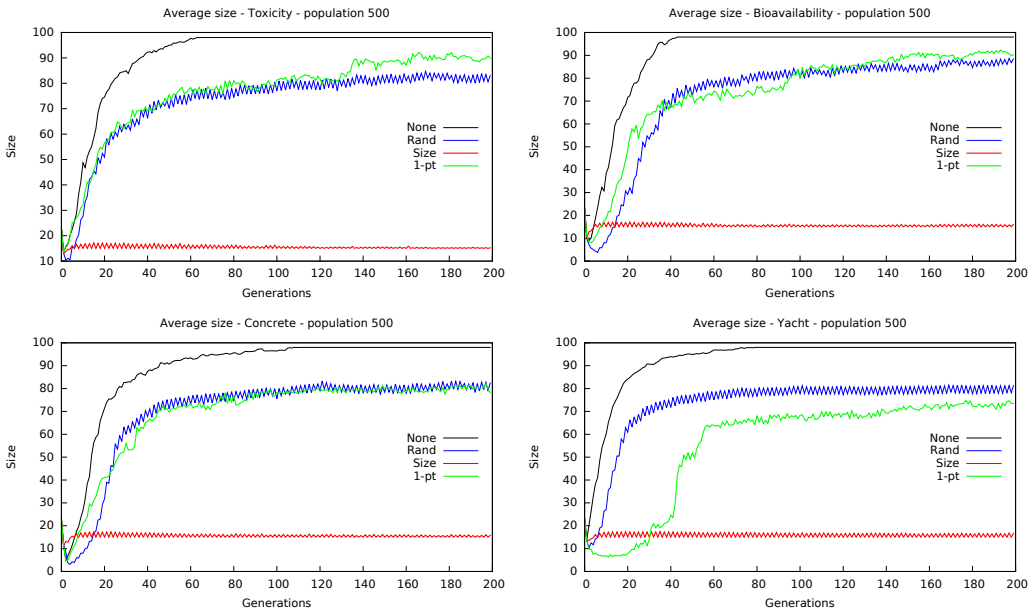


Figure 4.2: Average size per generation for all the problems: Interleaved-Size is consistently producing significantly smaller individuals.

some form perform at least as well as normal GP. Likewise, Interleaved-Size is no worse than Interleaved-Sample.

4.2.2.4 Size

Finally, Table 4.6 compares the average tree sizes in the final population. The average tree size is an estimate of the computational efficiency of various setups as clearly the approach that maintains the smallest sizes is the cheapest. In this case, we get a clear result: Interleaved-Size that competes well with the other approaches on previous statistics, outperforms the rest of the setups. The tree sizes are not only significantly smaller, they are often *substantially* so.

4.2.3 Discussion

The results show that the interleaved use of the training set in *some form* performs at least as well as standard GP on testing fitness and overfitting thus substantiating the idea earlier introduced in [Goncalves and Silva, 2013]. This indicates that there is merit in this approach particularly when the interleaved methods process much less data than the standard GP.

Section 4.1 introduced Interleaved-Random as a useful benchmark with a possibility to have smaller trees as the search is not fitness guided. However, rather surprisingly, the statistics for Interleaved-Random are often closer to the standard GP than the other two counterparts. Although the tree sizes with Interleaved-Random are no smaller than normal GP, the approach still gains over standard GP due to savings in data processing.

What is not clear from the results on the testing fitness and overfitting is that amongst all the methods which method is consistently the best. However, among the interleaved methods, both Interleaved-Random and Interleaved Size are at least as good as RIS. Moreover, it is remarkable that Interleaved-Size and Interleaved-Random that do not use any data at all in the interleaved generations are competitive with respect to the other two methods. This leads us back to the question: is there a great merit in using a single data point in the interleaved generation given that the information available is only minimal while the fitness evaluation can still be expensive? From the results obtained here, the

Population	50	250	500
	`Toxicity`		
None	98.61	86.39	98.65
Rand	89.17	81.52	83.57
Size	**16.38**	**16.05**	**15.71**
1-pt	84.60	79.86	82.02
	`Bioavailability`		
None	65.09	89.45	98.61
Rand	64.06	79.98	89.17
Size	**17.25**	**16.53**	**16.38**
1-pt	56.15	75.88	84.60
	`Concrete Strength`		
None	34.50	70.93	89.30
Rand	36,96	72.91	82.44
Size	16.35	**16.88**	**16.74**
1-pt	36.87	70.33	70.13
	`Yacht`		
None	72.43	92.58	97.03
Rand	61.10	82.80	82.16
Size	**16.44**	**17.19**	**16.79**
1-pt	21.87	46.62	48.12

Table 4.6: Average Size in the final generation for each of the four problems.

answer appears to be: No. Instead, the idea of interleaving generations seems robust enough even with other fitness measures, that is, random search and size minimisation.

As further evidence of the relative overfitting behaviour of various methods, we present Figure 4.1 that displays the median overfitting of the best individual throughout evolution. For space constraints we only present results for population size 500; this is also the size used in the original study on RIS. We only consider the relative trends here and note that Interleaved-Size, regardless of its rank in performance, maintains a relatively stable profile, unlike standard GP, which on `Bioavailability` and `Toxicity` tends to increasingly overfit towards later generations. Interleaved-Random appears to worsen overfitting towards the end of runs in `Toxicity`; however, this is statistically indistinct with RIS and Interleaved-Size. The figure also shows that Interleaved-Size performs consistently at least as well as RIS, and that this fact is not merely limited to the final generation.

To break the tie, we consider the results on tree sizes. In this case, we see a clear result in that Interleaved-Size consistently produces much smaller individuals than the rest of the setups. Although we do not have the exact results for the run-times, we noticed that the runs with Interleaved-Size finished earlier than the those with the other setups. Therefore, Interleaved-Size not only saves the effort in data processing but also successfully utilises the interleaved generation to counter bloat which is can be a limiting factor for GP runs. Detailed results are visible in Figure 4.2 for the population size 500. The figure shows that Interleaved-Size maintains a tight lid on size growth *throughout* evolution in a manner that totally prevents bloat.

These results show that interleaved use of training data is a powerful idea for GP. While further investigation is necessary to combine the merits of all the different approaches that we have tried here, it is clear that in Interleaved-Size we have a qualitatively competitive approach that is computationally efficient.

4.3 Conclusions

This chapter investigates alternative approaches to interleaving the use of training data set in GP to discourage overfitting and improve the computational efficiency of GP. The central question here is: whether using a single data point instead of the entire training set in alternate generations is central to the success of RIS as proposed previously.

Two alternatives to the previously proposed RIS are tried: interleaving the use of entire training data set with a simple random search (Interleaved-Random), and interleaving the use of training data set with minimising just the tree size (Interleaved-Size).

The results indicate that while Interleaved-Random produces trees of sizes similar to those from Interleaved-Sampling and standard GP, Interleaved-Size produces *substantially* smaller trees without sacrificing the efficacy of the Interleaved-Sampling. The smaller sizes reduced the GP run times significantly.

More generally the results indicate that while interleaved use of the training data set is indeed a useful idea, Interleaved-Random and Interleaved-Size perform just as well across on a range of problems without calling the fitness function *at all* in the interleaved generations. This is particularly useful because a fitness evaluation can be expensive even with a single data point in situations such as when an expensive simulation is needed to evaluate even a single data point. These elements suggest that environmental fluctuations in themselves may have useful effects in GP or other evolutionary computation based optimization methods.

5

Wave: Incremental Erosion of Residual Error

5.1 Introduction

Performance curves of GP typically rise steeply in the early generations before flattening off as evolution progresses. Later generations gradually accumulate small improvements; however, these improvements typically spread over a large number of generations and often accompany *code bloat* [Langdon and Poli, 1998]. Thus, there are diminishing returns as the run progresses. Not surprisingly then, a lot of GP literature innovates to improve the quality of GP runs, in particular, to extend improvement enjoyed by earlier generations.

However, a question naturally arises; given that GP performs most efficiently in the earliest generations, why bother with the later generations at all? Instead, can GP not leverage the speed of the initial generations repeatedly through multiple re-starts which *build upon* the progress made in the previous run or runs?

In this chapter we investigate an approach to problem solving with GP that is analogous to ecological change in nature. Specifically, we propose modifying the fitness landscape in order to make a problem more malleable for GP. We intend to accomplish this by interrupting the GP run when we detect that the solution does not improve anymore, then resume it after changing the fitness function and the population. This method could either be seen as a way to introduce ecological

changes or environmental fluctuations with a radical modification of the fitness landscape or/and as a form of *saltationism* [Blackburn, 1995].

We propose a new method, *Wave*, which uses the environmental fluctuations as its main principle of operation. To change the fitness function we rely on the best individual detected so far and create a new population to optimize its residual error. The creation of a new population after modifying the target that uses the previous evolutionary result as a black box unmodifiable part of the individuals (which could be biologically realistic, as elements which appeared early in the evolutionary process are usually less likely to be modified), is akin to a high level of mutation in the hope of eliminating long periods of stasis from of the evolutionary process.

This could be supported by the theory of *Punctuated Equilibrium* [Gould, 1972] (see Figure 5.1) which suggests that evolutionary changes are not necessarily uniform, but may instead emerge from long periods of *stasis* followed by rapid change (characterized for example by speciation). The biological literature reports multiple causes for such evolutionary changes, for example, *saltationism* whereby important mutations quickly relocate offspring to a distant and better point on the fitness landscape than the parent, *ecological changes* [Milligan, 1986] in the environment (and therefore in the fitness landscape) which may encourage a species to adapts to consume a new resource which can result in a rapid and important phenotypic change, and/or *peripatric speciation* [Mayr, 1982] where the geographical separation of asymmetric groups can allow the smallest to rapidly evolve phenotypic differences. The research question then is whether we can affect artificial punctuated equilibrium in GP through repeated *restarts* such that multiple GP runs co-operate to produce an effective combined solution.

We take the view that the truly interesting and useful problems and applications are so difficult that every computing cycle expended on GP should contribute to the eventual solution; multiple GP runs for the sake of generating statistics are all well and good, but when it comes to large scale problems rather than simple benchmarking, no cycle should be left behind.

Cascading evolutionary computation (EC) runs have existed for some time. The most similar method is *Sequential Symbolic Regression* [Oliveira et al., 2014]

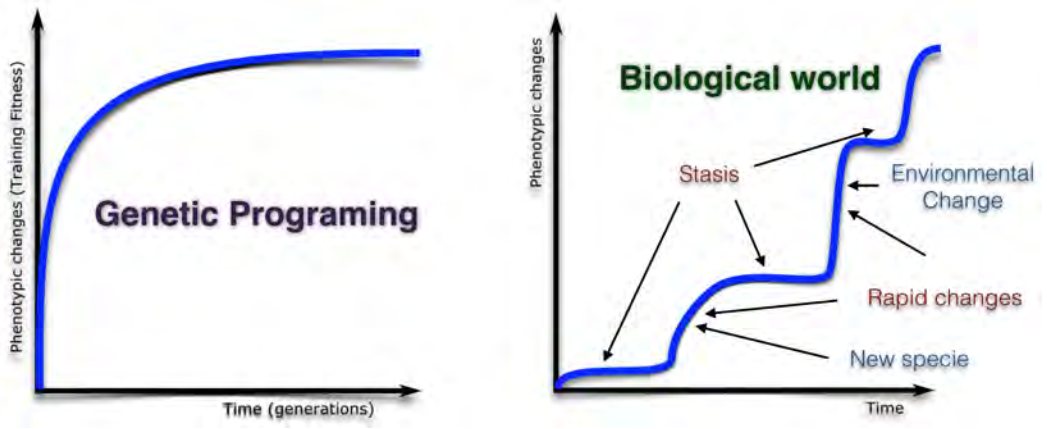


Figure 5.1: This graphic depict the typical differences between phenotypic changes in GP and in nature.

which also tackles symbolic regression problems and spreads the task of approximating the training data across a number of GP runs. *Wave*, the proposed system, differs from other approches because it embraces heterogeneity and leverages different parameters (population size, use of LS, etc.) by permitting the constituent runs to be significantly different from each other. Similar to the manner in which a natural wave goes through several *periods*, we aggregate improvements, eventually ending with a joint solution which is the sum of solutions presented at the end of each period. This is similar to creating an ensemble, except that the joint solution does not comprise of independently evolved components.

This has some biological parallels where not only does the environment change over time, but the evolving entities themselves *cause* the environmental changes; moreover, changing target data after every period and using heterogenous settings for each period causes a change in the fitness landscape. This chapter tests whether this metaphor works towards producing an efficient GP system that uses computationally cheap generations to match or even improve upon the results that GP normally produces over extended runs with significant code growth. We find that *Wave* results in an efficient GP system, which matches and often improves over the standard GP and only at a fraction of the cost.

The chapter reads as follows: Section 5.2 introduces Wave; Section 5.3 discusses the experimental setup and the problem suite used in this study; Section 5.4 details the results and discusses their significance; and, finally, Section 5.5 concludes the chapter highlighting the achievements.

5.2 Wave

Wave (see Figure 5.2) uses a collection of heterogenous GP runs to create a dynamic fitness landscape. In GP, the fitness landscape is the direct consequence of the data points used to compute the training fitness. With Wave, instead of performing one GP run (with a fixed parameter setting) over a large number of generations, we use a succession of smaller heterogenous (in terms of parameter settings) runs, and simply linearly sum their best results (without semantic operators). Each short run stops when it ceases to produce a significant improvement. At the end of each of these short runs we modify the fitness landscape so that the next one *builds* on the previous work. These short runs could thus be seen as periods of a wave progressively eroding the fitness landscape, thus using a *divide and conquer* approach. At the end of each period the best evolved individual is used to *reset* targets of our data set, thus creating a new fitness landscape; this resetting results by posing the *residual* between the target and the best evolved output as the new target for the next period. The next period starts afresh with a new population.

First we formalise the terms for the Wave system; this will facilitate describing the experimental settings and discussing the results.

A **period** (as shown in Figure 5.2) is similar to a normal GP run (randomly initialized at the first generation and terminated when the last generation is reached) except that at the end of each period the best individual is used to compute new target values; this creates a new fitness landscape for the subsequent period. Thus, if t is the target value at the start of a period, and f is the best evolved function for this period, then t' is the new data set for the next period such that $t' = t - f$. We terminate a wave when $MaxP$ periods have been processed.

While Wave is our proposed system, *a* **wave** is a particular collection of periods where the target varies across periods in the manner described above.

Notice, although given the definitions, a standard GP run can be described as a wave composed of a single period, to avoid confusion, we do not describe such runs as waves; instead, we refer to them as standard GP runs.

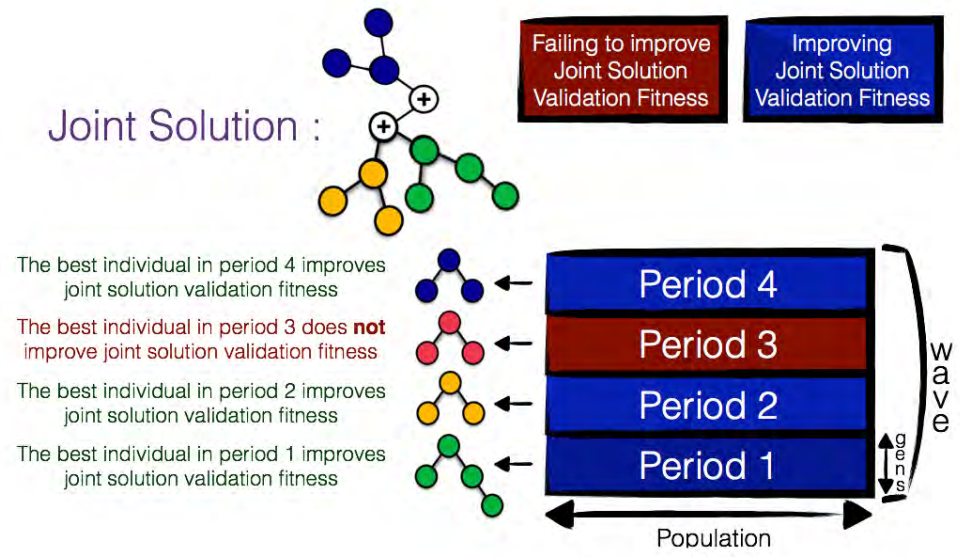


Figure 5.2: A simple Wave setup is depicted, where the population size and the number of generations stay fixed across the periods. Although the joint solution simply adds the best results of various periods, each best result is added to the joint result only if it decreases the cumulative error thus far. Width of periods is proportional to the size of the population, height of periods is proportional to the number of generations.

5.2.1 Updating Target Values

As illustrated in Figure 5.2, the final product, we shall call it joint solution, of a wave results from summing the outputs of the best individuals of successive periods, where each period *potentially* optimises a unique target data t'. We say

potentially because sometimes the target data may not be renewed for a new period. This happens if the current period, optimising over a current target set t, fails to produce an individual f_i such that $(f_i - t)^2 < (0 - t)^2$. In other words, adding f to the joint solution retains (at best) or worsens the error of the joint solution. In this case, we do not add f_i to the joint solution (we deem f_i *ineligible*) and, instead, launch the next period to again optimise over the same target set t. Therefore, $f_{joint} = \sum_{i=1}^{n}(f_i)$ where f_{joint} is the joint solution of a wave, n is the number of periods that produced eligible functions and f_i is the function from such a period i.

5.2.2 Heterogenous periods

Key to Wave's design is the use of heterogenous periods. Each wave can use completely different settings, even to the extent of using different search algorithms, as long as the output of the particular wave can be combined with the others.

In this work we consider four aspects that we vary throughout the entire run. These are the use of a flexible end point for periods, varying the size of the population and / or the number of generations and alternating the application of LS.

5.2.2.1 Flexible ending of periods

We decide that a period should end if the best fitness has not improved significantly during the last few generations. Using the punctuated equilibrium analogy we stop a period during a stasis phase but not during a phase of rapid change. Therefore, to decide whether we are in a stasis phase, we compare the improvement over the last two generations (current improvement) with that over the three generations before the last two (previous improvement). If the current improvement is less than 0.5% of the previous improvement, we end the period. However, each period still undergoes a certain minimum number of generations before we take action. Therefore, the following conditions 5.1 and 5.2 formalise the period stopping criteria:

$$g_c > g_m \tag{5.1}$$

$$(B^F(g_c) - B^F(g_c - 2)) \leq (B^F(g_c - 2) - B^F(g_c - 5))/200 \tag{5.2}$$

where g_c is the current generation, g_m is a minimum number of generations before a period stops and $B^F(g_c)$ is the best training fitness at generation g_c.

5.2.2.2 Varying population size and minimum number of generations

As Figure 5.3 shows, it is possible to vary the number of generations and population size for each period. Given that each problem with standard GP may require a different value for these two parameters [Piszcz and Soule, 2006], varying them across different periods of a wave may make sense. Thus, we observe the effect of increasing the population size and the minimum number of generations before stopping a period, but only when the previous period failed to improve the fitness of the joint solution. These changes are *cumulative*.

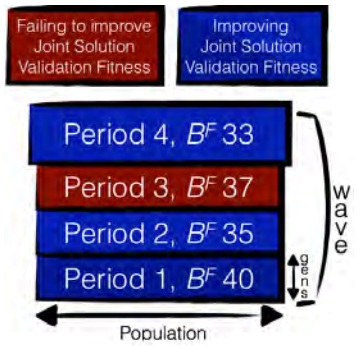


Figure 5.3: Adaptive Wave settings is depicted. The periods vary in population size & number of generations (periods in red failed to improve joint solution). Width of periods is proportional to the size of the population, height of periods is proportional to the number of generations.

5.2.2.3 Alternating between using and not using Linear Scaling

Linear scaling (LS) optimises the slope and intercept of an evolving function[1] and has been effective on symbolic regression problems. We propose to use LS alternately as a means of generating environmental fluctuations. This is also a way to question if LS *consistently* improves performance: in three data sets used in this study (Yacht, Concrete and Poly-10) standard GP outperformed *scaled* GP, as it can be seen in Tables 5.3, 5.7 and 5.4. Therefore, we also report experiments where periods alternate between using and not using LS. In these cases the initial period uses LS[2].

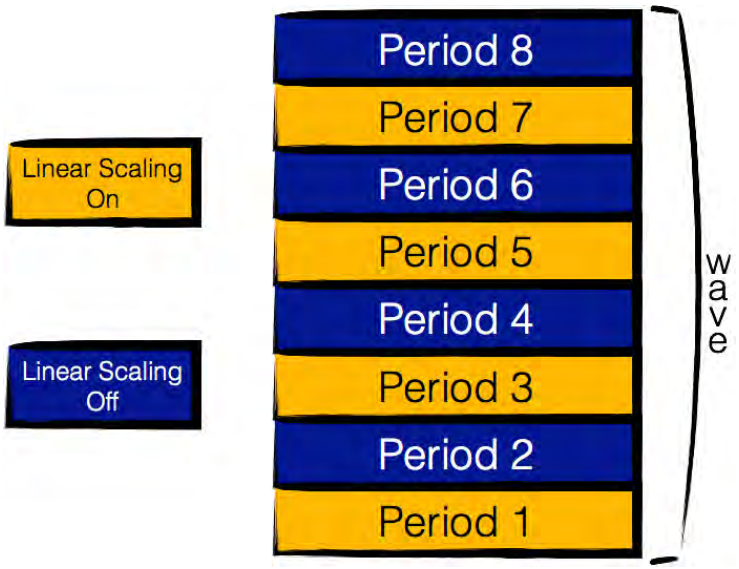


Figure 5.4: Alternating use of LS: Wave activates or deactivates the LS alternately at the beginning of each period.

[1]Therefore it changes the environment where individuals evolve.
[2]Exploratory experiments showed that Wave performs better this way.

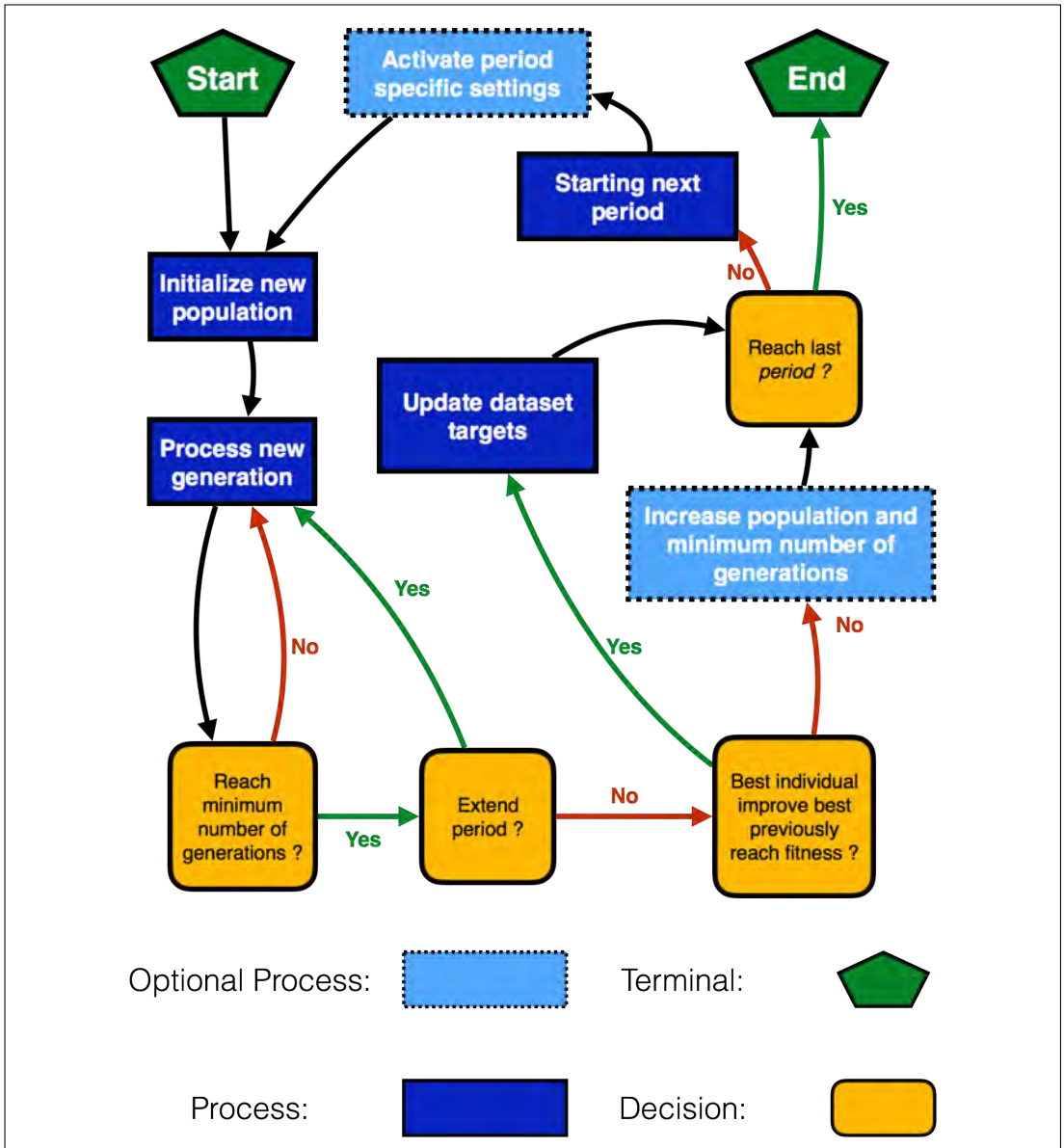


Figure 5.5: A flowchart summarizing the different steps of Wave: With the settings used in this work, *activate period specific settings* only occurs while alternating between using and not using LS and *increase number of generations & population* while varying population size and minimum number of generations.

5.3 Experiments

Table 5.1 lists the configuration parameters that we consistently use across all the experiments, except where stated otherwise.

Parameter	Value
Replacement Strategy	Generational
Operator Probabilities	Xover: 0.9; Point mutation: 0.1
Tournament Size	10[a]
Max. depth	17
Max. size	100
Functions set	$+,-,\times,\div$[b]
Terminal set	Inputs and constants -1.0, -0.5, 0.0, 0.5 & 1.0
Fitness	RMSE
Initialisation	Ramped half & half
Max. initial depth	8

[a] A relatively high tournament size but recently successfully used in [Goncalves and Silva, 2013] and [Azad et al., 2014].
[b] Protected division.

Table 5.1: Common Parameters: These parameters are used for Wave and GP benchmarks.

5.3.1 Problem Suite

For this study we have used three multi-dimensional data sets from the UCI Machine learning repository [Bache and Lichman, 2013] and two mathematical functions. Those from the UCI are **Concrete Strength** where the objective is to predict the compressive strength of concrete and data set includes 1030 instances each having 8 inputs, **Yacht** where the objective is to predict the hydrodynamic performances of a yacht; the data set includes 308 instances each with 7 inputs, and **Powerplant**, where the task is to predict the net hourly electrical energy output of a power plant and the data set includes 9568 instances and 4 inputs.

The two mathematical functions are **Poly-10 [Poli, 2003]** ($y = x1 * x2 + x3 * x4 + x5 * x6 + x1 * x7 * x9 + x3 * x6 * x10$) and **Div-5 [Vladislavleva et al., 2009b]** ($y = \frac{10}{5+\sum_{i=1}^{5}(x_i-3)^2}$). For each function, we randomly generate 500 data

points in the range $[0; 1]$ and for each wave we randomly split the given data set into two subsets of equal size for training and testing purposes.

5.3.2 Benchmarks

The benchmarks we have chosen to compare Wave with include both EC-based and non EC-based machine learning methods. The EC-based methods are standard GP, both with and without LS, with a population size of 500. In all cases, each run spans 100 generations. The non-EC method is multiple linear regression (MLR), an efficient method to solve regression problems which, in an award winning recent effort, has been put forward as a tough benchmark for GP[Arnaldo et al., 2014]. For all experiments we split the data randomly into equal partitions for training and testing purposes. At each generation, each individual's fitness is computed both on testing and training data set. Selection, of course, is conducted using only training fitness.

5.3.3 Naming Conventions

The naming convention we adopt for Wave setups has the following format:

$$Wave : PeriodsNumber : Setting - P : PopulationSize$$

where $PeriodsNumber$ is the number of periods in each wave, $P : PopulationSize$ is the population size of the first period of the wave (e.g. $P : 500$ is a population of 500 individuals) and $Setting$ indicates whether LS was used (LS) or not (NS). However, $LS : NS - P$ indicates alternating between scaled and normal periods. Similarly, standard GP settings follow a similar pattern, that is, $GP : Setting - P : PopulationSize$.

Method	LS [a]	Pop[b]	Gen [c]	G. Inc[d]	P. Inc [e]	Period[f]
GP:LS-P:500	On	500	100	NA	NA	NA
GP:Norm-P:500	Off	500	100	NA	NA	NA
Wave:25:LS-P:100	On	100	10	1	18	25
Wave:25:NS-P:100	Off	100	10	1	18	25
Wave:200:LS-P:25	On	25	5	0	0	200
Wave:25:LS-P:500	On	500	10	0	0	25
Wave:25:NS-P:500	Off	500	10	0	0	25
Wave:25:LS:NS-P:500	Alt[g]	500	10	1	18	25

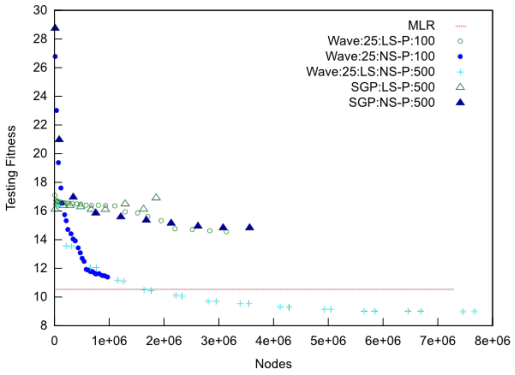
[a] Linear Scaling
[b] Initial population size
[c] Initial minimum number of generations
[d] Number added to current minimum number of generations after a Period failed to improve training fitness
[e] Number added to current population size after a Period failed to improve
[f] Number of consecutive Periods
[g] Alternation between LS and non LS, Gen 1 with LS. training fitness

Table 5.2: Configurations of the different Wave and GP.

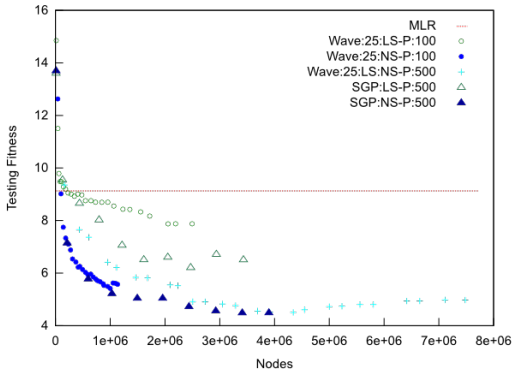
5.4 Results and Discussion

Due to phenomena such as over-fitting, the best test fitness is not necessarily reached either at the last period of a wave or at the last generation of an GP run. Therefore noting only the end of run results can be *unrepresentative* for any setup, and may require tailoring the end of runs to each setup. To avoid that and to make a fair comparison between GP and Wave, we measure the various run statistics at the end of each period for Wave experiments and every ten generations for GP; we call these reporting times *moments*. At every particular moment we measure the curent generation or the current period, the training fitness, the testing fitness and the number of nodes processed.

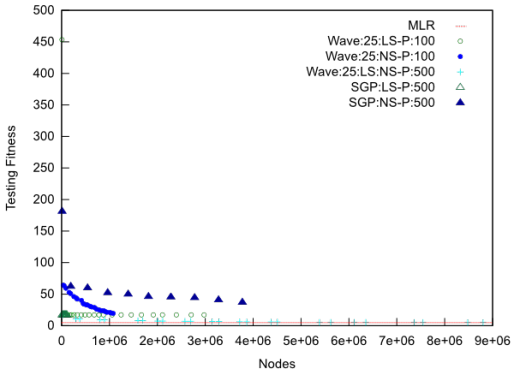
In this study, at every moment, we report median values for training and test fitness as the median is a robust measure of central tendency and represents data more meaningfully [Hoaglin et al., 1983]. We also present the best moment (in terms of testing fitness) for each configuration and the total number of nodes processed and is expressed in nodes, 10^{10} nodes. By presenting the best moment, instead of a consistent, pre-determined fixed moment (or the last generation for



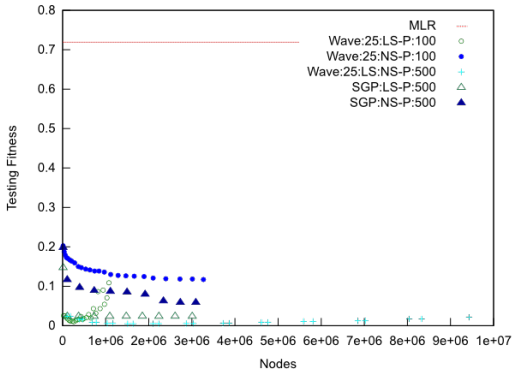
(a) Concrete testing fitness per nodes processed.



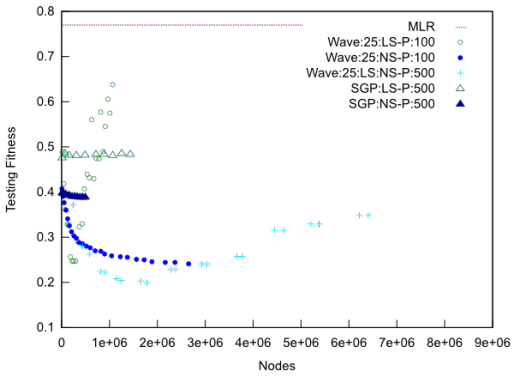
(b) Yacht Testing per Nodes processed.



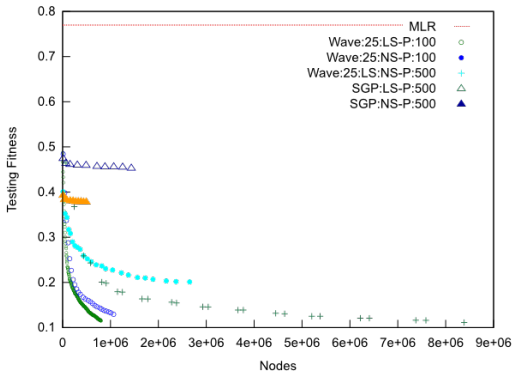
(c) Powerplant testing fitness per nodes processed.



(d) Div-5 testing fitness per nodes processed.



(e) Poly-10 testing fitness per nodes processed.



(f) Poly-10 training fitness per nodes processed.

Figure 5.6: Fitness / number of nodes is evaluated for each moment. The fitness is computed on the testing data set for Figure 5.6a to 5.6e and on the testing data set for Figure 5.6f.

GP), we avoid reporting at an unrepresentative point in time. We can however, see whether such a moment comes very early without evolving much or at a reasonably later point which tests whether the algorithm in question can battle over-fitting.

For each data set, except powerplant, the reported results for Wave and GP experiments are median value on 100 runs (or waves) of each. For the powerplant data set, we report results only for 30 runs at each configuration[1]. We also run MLR 100 times[2].

Due to the heterogeneity of the settings investigated, it can be difficult to compare them in a fair manner on testing fitness alone, as we cannot expect a wave with a small fixed population to do better than one with a large dynamic population. However, the smaller Waves may nevertheless be useful if they can achieve acceptable testing fitness with a modest computational cost. To look into the Trade-Off between efficiency and computational cost of the respective systems, Figure 5.6 plots the testing fitness against the number of nodes processed since the beginning of a run at each moment.

5.4.1 The speed/accuracy Trade-Off

Usually testing fitness is the key criterion to compare the performance of machine learning algorithms; however, with a deluge of data (or *big data*), training efficiency also becomes important, particularly when initial training runs are conducted to estimate the distribution of the data or salience/relationships/dependencies within the data.

In cases such as these, it is more important to have *Fast Learners* that have a reasonable approximation to the solution, as they will keep the response time low. Thus, training speed of the cheapest Wave setup (Figure 5.6), Wave:200:LS-P:25, is important for future applications.

In Tables 5.3-5.7 we report the best median training and testing fitness among all moments for each setting on each data set. Lowest testing fitness among

[1]This is due to a high number of data points resulting in substantial computational cost.

[2]MLR method is deterministic but we use random training and testing data sets so result may vary between runs.

Wave settings and among GP settings are in bold. Additionally, we report the **Trade-Off Wave** which, at some moment, produced the best test fitness while consuming fewer nodes than the best GP moment. If the Trade-Off Wave also has better test fitness than the best result with GP, clearly it is better both in terms of solution quality and expense.

We also note the **Fastest Good Wave**; this is a wave setup which outperforms the best GP moment on test fitness and consumes fewer nodes than any other wave.

Finally, we also report results from the MLR benchmark. Of course, in this case we can not compare the nodes used, neither can we record moments.

To test the statistical significance of reported differences in performance, we use the Mann-Whitney U test at $p = 0.05$.

5.4.2 Discussion

The results show that Wave with a population size of 500 with alternating LS performs the best among all the configurations investigated, except on Yacht, those results are statistically significant. This is rather surprising because one would expect LS to be more effective if used consistently. While the best moment for the best Wave setup consumed a higher node requirement than GP, Figures 5.6 shows that this setup is the best after consuming a similar number of nodes.

Also notice that while the *best* moment of the best Wave setup is computationally more expensive than that with GP, the *Trade-Off Wave* and the *Fastest Good Wave* are by definition cheaper for almost all of the problems tackled.

Although the cheapest Wave setup, Wave:200:LS-P:25, is not among the best performers, Tables 5.5 and 5.6 demonstrate that it performs quite reasonably on some data sets, despite a very small population size of only 25. This configuration can produce better solutions on out-of-sample data than GP, but even when it does not, the performance is not drastically worse, showing that it can operate reasonably with limited computational resources.

While Figures 5.6 depict the good performance of Wave with a population size 500 in terms of test fitness, we see also that Wave with a population size of 100 achieves very good results with considerably fewer node evaluations. Therefore,

the latter configuration may prove to be a good choice if computational resources are scarce. We also note that if for a particular data set GP with LS does better than GP without LS, Wave with LS produces even better results.

Method	Moment	Train	Test	Nodes
GP:LS-P:500	71g	5.16411	6.21251	2468k
GP:NS-P:500	81g	4.09004	**4.48998**	3410k
Wave:25:LS-P:100	23p	7.42025	7.87332	2056k
Wave:25:NS-P:100	22p	4.01469	5.41762	999k
Wave:200:LS-P:25	200p	9.02013	9.30088	450k
Wave:25:LS-P:500	25p	6.48936	7.08987	3229k
Wave:25:NS-P:500	9p	3.38198	4.73403	2464k
Wave:25:LS:NS-P:500	16p	3.35759	**4.50582**	4342k
MLR	NA	8.86019	9.11015	NA
Trade-Off Wave:				
Wave:25:LS-P:500	9p	3.38198	4.73403	2464k
Fastest Good Wave:				
NA[a]				

[a] Wave does not perform better than GPs on this data set therefore such a result does not exist.

Table 5.3: Yacht data set: Experimental results.

Results indicate that Wave is an efficient method. The best Wave moment is significantly better than the best GP moment on four of the five data sets studied, and the improvement on testing fitness is substantial. Also as shown in Table 5.8 the average tree sizes at the end of a wave are usually significantly lower with Wave than with GP; note that we limit the number of nodes to 100 for GP and that the difference could be greater without this contrain.

The Fastest Good Wave results show that Wave not only achieves significantly better training fitness, but also produces at least equivalent testing fitness in four

Method	Moment	Train	Test	Nodes
GP:LS-P:500	51g	14.1436	16.1199	671k
GP:NS-P:500	81g	14.6654	**14.8268**	3080k
Wave:25:LS-P:100	25p	14.3886	14.5569	3139k
Wave:25:NS-P:100	25p	10.3821	11.3939	970k
Wave:200:LS-P:25	12p	16.2683	16.3699	402k
Wave:25:LS-P:500	21p	13.8497	13.9208	1753k
Wave:25:NS-P:500	14p	8.72476	10.1065	4418k
Wave:25:LS:NS-P:500	24p	7.71554	**8.98308**	7463k
MLR	NA	10.3123	10.5693	NA
Trade-Off Wave:				
Wave:25:LS:Norm;p:500	13 p	9.08195	9.69879	2950k
Fastest Good Wave:				
Wave:25:LS-P:500	2 p	14.4465	14.5430	133k

Table 5.4: Concrete data set: Experimental results.

of the five data sets, with significantly fewer node evaluations. For each data set at least one moment of a wave reaches equal or statistically better testing fitness than the Best Standard GP moment, at a fraction of cost, even if the final combined program can reach huge sizes.

The best Wave moment is also significantly better than MLR on four data sets, and the difference in the remaining Powerplant data set is not statistically significant.

In terms of training fitness, Wave with a larger population always reaches significantly better fitness than either GP or MLR. However, there is some evidence of over-fitting with this setup, as can be seen in Figure 5.6f.

The setting which emerges as the most efficient is that which alternates between use and non-use of LS (Wave:25:LS:NS-P:500). This outperforms the chosen benchmarks on four out of the five data sets, and among Wave setups, is only outperformed by Wave:25:LS-P:500 on the poly-10 problem.

The heterogenity within the Wave proves particularly useful because, rather surprisingly, LS does not always produce the best results, even when employed

Method	Moment	Train	Test	Nodes
GP:LS-P:500	1g	17.0788	**17.1299**	11k
GP:NS-P:500	91g	37.8269	37.2500	3775k
Wave:25:LS-P:100	2p	17.0738	17.0571	23k
Wave:25:NS-P:100	25p	18.8342	19.5098	1077k
Wave:200:LS-P:25	181p	17.0157	17.1246	403k
Wave:25:LS-P:500	2p	17.1188	17.0372	99k
Wave:25:NS-P:500	25p	10.7321	11.3897	7530k
Wave:25:LS:NS-P:500	24p	4.86973	**5.15350**	8480k
MLR	NA	5.12806	5.13028	NA
Trade-Off Wave:				
Wave:200:LS-P:500	3 p	17.0561	17.0781	6k
Fastest Good Wave:				
Wave:200:LS-P:500	3 p	17.0625	17.0779	6k

Table 5.5: Powerplant data set: Experimental results.

with GP. A mixed approach using the Wave paradigm appears to be a more flexible strategy that produces consistently good results.

The Powerplant data set is particularly interesting because both GP:LS and GP:Norm seem inefficient on this data. GP:LS does not appear to evolve at all and while evolution is apparent with GP:Norm, it is very slow. However, the combination of these two configurations in Wave:25:LS:NS-P:500 reaches good fitness in its early periods while also maintaining its ability to evolve as seen on Fig 5.6c.

5.5 Conclusions

In this chapter, we present a novel approach to GP which we call Wave, where we use a small number of generations in each wave, compute the residual and delegate the outstanding improvement required to further runs. In this way we

Method	Moment	Train	Test	Nodes
GP:LS-P:500	71g	0.02377	**0.02395**	2234k
GP:NS-P:500	81g	0.05738	0.05897	2731k
Wave:25:LS-P:100	7p	0.00698	0.00999	258k
Wave:25:NS-P:100	25p	0.10738	0.11704	3267k
Wave:200:LS-P:25	6p	0.02829	0.03099	15k
Wave:25:LS-P:500	6p	0.00442	0.00484	1050k
Wave:25:NS-P:500	23p	0.07700	0.08441	2143k
Wave:25:LS:NS-P:500	9p	0.00424	**0.00480**	1508k
MLR	NA	0.71604	0.71736	NA
Trade-Off Wave:				
Wave:25:LS:Norm;p:500	10 p	0.00424	0.00480	1636k
Fastest Good Wave:				
Wave:25:LS-P:100	3 p	0.01808	0.02098	78k

Table 5.6: Div-5 data set: Experimental results.

systematically leverage the inherent behaviour of GP by which rapid improvement typically occurs in early generations and we avoid the code growth associated with prolonged evolution.

The results of this preliminary investigation are encouraging, showing that Wxzave could be a promising paradigm in the search for improved performance and scalability in GP. The Wave paradigm offers an additional saving in computational effort as there is no need to re-evaluate the fixed (black box) subexpressions. This provides the potential for increasing the total complexity of individuals without additional cost. The wave approach is also flexible, and we anticipate many possibilities for further study. For example, wave could be totally or partially composed of periods using other EC methods such as Interleaved Sampling. Another approach would be to vary evolutionary parameters such as function set or genetic operator types and/or probabilities during a wave.

In fact, the wave paradigm is so flexible that it does not necessarily need to be limited to EC and could, for example, be combined with MLR or other optimization methods.

Method	Moment	Train	Test	Nodes
GP:LS-P:500	1g	0.47447	0.47591	11k
GP:NS-P:500	71g	0.37882	**0.38868**	383k
Wave:25:LS-P:100	7p	0.35781	0.45138	32k
Wave:25:NS-P:100	25p	0.20110	0.24104	2653k
Wave:200:LS-P:25	9p	0.20627	0.24712	220k
Wave:25:LS-P:500	6p	0.16587	**0.18736**	1097k
Wave:25:NS-P:500	24p	0.17971	0.22286	2824k
Wave:25:LS:NS-P:500	10p	0.16333	0.19930	1777k
MLR	NA	0.76773	0.77197	NA
Trade-Off Wave:				
Wave:25:LS-P:100	7 p	0.20627	0.24712	220k
Fastest Good Wave:				
Wave:25:Norm;p:100	3 p	0.35231	0.37657	51k

Table 5.7: Poly-10 data set: Experimental results.

There are, of course, some minor caveats, the most significant of which is that as the initial best sub-expression is systematically selected it is possible that if this first wave is not a good one we risk polluting the remaining periods. Informal experiments with alternating LS and normal GP tended to confirm this idea. Further work should address this potential issue.

In this initial study, we have focused on applying wave to symbolic regression problems. It would be interesting to investigate how the wave approach could be applied to other learning tasks such as classification.

Settings	Yacht	Conc	P-10	Div-5	PowP
GP:LS-p:500	93.0	20.1	32.6	84.3	2.0
Wave:25:LS-p:100	11.0	2.0	2.0	40.9	2.8
Wave:25:NS-p:100	32.8	14.2	5.0	26.6	13.4
GP:NS-p:500	92.3	99.9	5.0	8.0	97.1
Wave:200:LS-p:25	14.2	2.4	2.4	2.0	2.2
Wave:25:LS-p:500	27.6	87.3	4.6	65.8	2.1
Wave:25:NS-p:500	47.3	36.2	30.9	10.2	33.8
Wave:25:LS:NS-p:500	21.3	8.0	14.9	34.0	2.0

Table 5.8: Average size of individuals at the end of each period or GP run.

6

Evolution of Heterogeneous Cellular Automata in Fluctuating Environments

6.1 Introduction

In natural evolution, environmental changes may include cyclic events such as seasonal changes or the daily alternation of light and darkness, occasional changes such as the appearance of new predators or the potential for new food sources, or more radical modifications such as environmental stresses induced by climate transitions.

We wish to study the effects of such fluctuations in a model closer to the natural world without using an explicit objective function and determine if these fluctuations promote phenotypic selection over genotypic selection. To that goal, we propose an open-ended experimental setup allowing us to systematically and quantitatively measure the influence of cyclic environmental fluctuations on the course of the evolution of cellular automata (CA). We show that such fluctuations lead to the emergence of processes similar to those exhibited by EIS.

We showed in Chapter 3 that HetCA could exhibit long-term phenotypic dynamics, a high level of variance over very long runs, greater behavioral diversity than classical CA, and "evolutionary progress" Shanahan [2012] on three criteria: robustness, size and density of the genotype [Medernach et al., 2015b].

Finally several of the potential units of selection that have been proposed during the evolutionary biology debate over the units of selection in natural selection [Lloyd, 2012; Okasha, 2006a], that we refer to in Chapter 2.1.3, are potentially included in HetCA. There is genotypic selection of the transition rules, but also phenotypic selection of cell groups able to replicate patterns such as the ones found in the Game of Life. This point is important when one is interested in environmental fluctuations because we anticipate that the existence of frequent environmental fluctuations will promote phenotypic selection over genotypic selection.

The Chapter is organized as follows: first the implementation of environmental fluctuations in HetCA is described in Section 6.2; next, we specify the computational setup used to study environmental fluctuations in Section 6.3; this is followed by a report on the experimental results and a discussion of their implications in Section 6.4. Finally, we propose a qualitative analysis in Section 6.5 and a conclusion in Section 6.6.

6.2 Experimental Setup

As in the previous version of HetCA 3, the genotype of an individual consists of its transition rules encoded in a custom CA-LGP program using the function set listed in Table 6.1. Such a program maps the space of neighborhood states to a new cell state, while providing an evolvable representation framework based on an alphabet of elementary functions. Individual genotypes are modified by micro-mutations (change in one component of a statement) and macro-mutations (addition or removal of an entire statement) of the corresponding CA-LGP programs.

Previously, the new genotype of a living cell c during genetic transfer was chosen randomly among candidate genotypes according to a uniform distribution. While, to be candidate, a genotype must either be the genotype of c, or be the genotype of a living cell, in c's von Neumann neighborhood, that return a living state for c's current Moore neighborhood[1]. In the present study, to

[1]There may be no candidate genotype, in the case of a quiescent cell surrounded by cells in the quiescent or decay state, in which case the cell concerned remains quiescent.

operator	**action** on inputs (x, y)
abs	$\lvert x\rvert$
plus	$x + y$
delta	1, if $\lvert x - y\rvert < 1/10,000$; 0 o.w.
dist	$\lvert x - y\rvert$
inv	$1 - x$
inv2	safeDiv$(1, x)$
magPlus	$\lvert x + y\rvert$
max	$\max\{x, y\}$
min	$\min\{x, y\}$
safeDiv	x/y if $\lvert y\rvert > 1/10,000$; 1 o.w.
safePow	x^y, if defined; 1 o.w.
thresh	1, if $x > y$; 0 o.w.
times	xy
zero	1, if $\lvert x\rvert < 1/10,000$; 0 o.w.

Table 6.1: Function set.

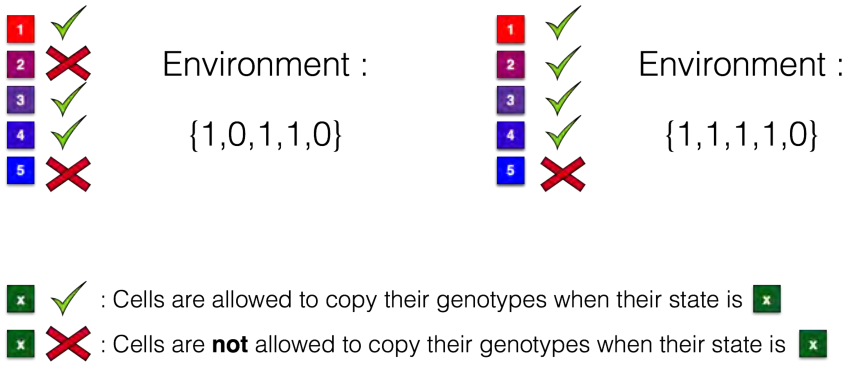


Figure 6.1: Visual explanation of the notation to describe an environment.

Name	**Short Name**	**Cycles**	**Transitions**	**Environment list: propagation probabilities $E = \{K(S_1), ..., K(S_5)\}$ of the living types**
Stable Environment	(SE)	NA	NA	$\{1,1,1,1,1\}$
Short-cycle Fluctuations	(ScF)	100	1	$\{0,0,1,1,1\}$, $\{1,1,1,0,0\}$
Light Fluctuations	(LF)	5,000	1	$\{1,1,1,1,0\}$, $\{1,1,1,0,1\}$, $\{1,1,0,1,1\}$, $\{1,0,1,1,1\}$, $\{0,1,1,1,1\}$, $\{1,1,1,1,1\}$
Strong Fluctuations	(SF)	5,000	1	$\{0,0,1,1,1\}$, $\{1,1,1,0,0\}$, $\{0,1,0,1,1\}$, $\{1,0,1,1,0\}$, $\{0,1,1,0,1\}$, $\{1,1,0,1,0\}$, $\{1,0,1,0,1\}$, $\{0,1,1,1,0\}$, $\{1,0,0,1,1\}$, $\{1,1,0,0,1\}$, $\{1,1,1,1,1\}$

Table 6.2: Stable and fluctuating environments descriptions.

introduce environmental variations we vary the likelihood of propagation of a genotype according to its cell state S_c. In this new setup, the probability $P(c)$ of a candidate genotype of being selected becomes $P(c) = K(S_c)/\sum_{i=1}^{n} K(S_{c_i})$, where $K(S)$ the state distribution and n the number of candidate genotypes. Therefore an environment E is characterized by the propagation probabilities of the 5 living states: $E = \{K(S_1), ..., K(S_5)\}$, as depicted in Figure 6.1. To mimic environmental fluctuations we initialize the simulation with $K(S_i) = 1$ everywhere, then regularly modify those values every f iterations starting from iteration 3,000 of the CA. We introduce three types of environmental fluctuations (Table 6.2).

Short-cycle fluctuations (ScF) iterate through two rather different environments, $\{0,0,1,1,1\}$ and $\{1,1,1,0,0\}$, every f iterations of the CA (for each

environment). We choose $f = 100$ to remain within the range of frequency described by Lipson et al. [2002] and Yu [2007]. Here we consider that a successful reproductive cycle for a cell involves passing through the quiescent state. This should take between 2 iterations (alternating between quiescent and living) and 7 iterations (after which a living cell decays and can no longer receive a genotype for a long period of time).

Light fluctuations (LF) iterate through 6 environments every $f = 5,000$ iterations (for each environment). The first 5 environments each prohibit a different living state (out 5 possible ones) from spreading its genotype; the last one gives an equal chance to all living states.

Strong fluctuations (SF) iterate through 11 environments every $f = 5,000$ iterations (for each environment). The first 10 environments each prohibit a different pair of living states (out 10 possible ones) from spreading their genotypes; the last one gives an equal chance to all pairs.

The rationale behind ScF is their analogy with circadian rhythms in certain bacteria. The intention is to mimic the highly regular cycles during which these organisms have enough time to reproduce repeatedly. LF, by contrast, are more similar to seasonal fluctuations, while SF resemble ecological crises. However, owing to the variety of both biological temporal rhythms and reproductive cycles, the relevance of these analogies remains rather limited.

6.3 Simulations

For each of the three types of environmental fluctuations and the stable (non-fluctuating) environment (SE) we performed 50 simulations, giving a total of $4 \times 50 = 200$ runs. Each cell of the CA was initialized in a random state, then each cell in one of the 5 living states was initialized with an individual randomly generated genotype. Each run lasted 500,000 iterations under the parameters listed in Table 6.3.

Parameter	Value
Number of living states	5
Successive living iterations before decay	7
Number of iterations during decay	375 to 1,875
Direct transition to decay	enabled
Size of the grid	500×500
Grid boundaries	toroidal grid
Transition Rule (TR)	CA-LGP
Maximum TR size	50 statements
Genotype copy neighborhood	von Neumann (4)
Transition rule neighborhood	Moore (8)

Table 6.3: HetCA parameters used in all experiments.

6.3.1 Genotype Size

We used the number of program statements (n_{prog}) as a measure of genotype size and computed the average size of all current genotypes of a run every 2,500 iterations. We then reported the average and standard error of the mean (SEM) among all 50 runs sharing the same settings.

6.3.2 Phenotype Comparison

If environmental changes lead to the emergence of phenotypic selections (similar to the EIS) using easily reversible phenotypic mutations [Jablonka et al., 2005c], then phenotypes from different individuals of the same lineage observed while environmental conditions are similar should also be relatively similar even though individuals from their lineage evolved in other environmental conditions between these measures. By contrast, if the adaptation to each environmental change is done exclusively through the selection of classical, irreversible genotypic mutations, these phenotypes should be quite different, despite the potential evolutionary convergence. This is why we developed a metric to measure phenotypic proximity between two iterations of the CA. To do this we simply compared the distributions of living cells over the different living states. Thus the phenotypic

difference $\sigma(t_1, t_2)$ between two iterations t_1 and t_2 was calculated as follows:

$$\sigma(t_1, t_2) = \sum_{S=1}^{5} \left| \frac{N(S, t_1)}{N(t_1)} - \frac{N(S, t_2)}{N(t_2)} \right| \tag{6.1}$$

where $N(S, t)$ is the number of cells in living state S at iteration t and $N(t) = \sum_{S=1}^{5} N(S, t)$ is the total number of living cells.

Every 5,000 iterations of the CA we performed two phenotypic comparisons between the current iteration $t_1 = t$ and an iteration in the past, $t_2 = t - \Delta t$. In one scenario, the temporal distance Δt was a multiple of the periodicity f, so that we compared two similar environments: $E(t_1) = E(t_2)$. We chose $\Delta t =$ 60,000 in the SE, ScF and LF cases and 55,000 in the SF case. In another scenario, we introduced an additional single-period shift such that we compared two dissimilar environments: $E(t_1) = E(t_2 + f)$. Here Δt was respectively equal to 60,100, 65,000 and 60,000 in the ScF, LF and SF cases.

6.3.3 Diversity

We use the "true diversity index" of order two [Jost, 2006] [Tuomisto, 2010a], [Tuomisto, 2010b] to measure phenotypic and genotypic diversity at every iteration t of the CA. Phenotypic diversity therefore reads:

$${}^2D_p(t) = \left(\sum_{S=1}^{5} \left(\frac{N(S, t)}{N(t)} \right)^2 \right)^{-1} \tag{6.2}$$

while genotypic diversity is:

$${}^2D_g(t) = \left(\sum_{g=1}^{G(t)} \left(\frac{N(g, t)}{N(t)} \right)^2 \right)^{-1} \tag{6.3}$$

where $G(t)$ is the number of distinct genotypes at iteration t and $N(g, t)$ is the number of cells sharing genotype g at iteration t. Note that $N(t) = \sum_{S=1}^{5} N(S, t) = \sum_{g=1}^{G(t)} N(g, t)$.

6.3.4 Homogeneous Test

We also collected the *most common genotype* (MCG), i.e. the most frequently occurring one, starting at iteration 2,500, then 102,500 and again every 100,000 steps until iteration 500,000, thus creating 6 sampling points (it was shown by Medernach et al. [2015b] that during the initial iterations of HetCA the MCGs were unlikely to exhibit any viable survival strategy). All MCGs collected in that way were exported and tested alone in homogeneous conditions, i.e. where all cells were initialized with that genotype and no mutations could occur. Since each collected MCG from any one of the four environments (SE, ScF, LF and SF) was tested again in all four environments, we performed a total of $4 \times 50 \times 6 \times 4 = 4800$ runs. The maximum duration of these runs was set to 60,000 iterations. Sometimes the genotype was not adapted to the environment and all living cells became extinct (i.e. transitioned to decay or quiescent state) before the end of the simulation. We considered a homogeneity test to be successful if living cells did not all become extinct before 60,000 iterations. In the Section 6.4, we report the success rate of those simulations, along with statistics on the final iterations of the failed runs.

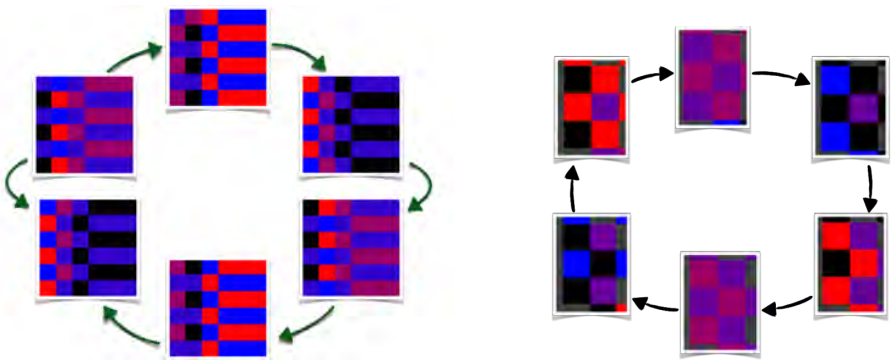
6.3.5 Phenotypic Disturbance in the Homogeneous Test

In HetCA, to survive in the long term genotypes must regularly "release" cells by transforming them into quiescent cells without genotypes. This generates patterns and cycles easy to observe in homogeneous simulations where a single genotype is tested (Figure 6.2a). It is also observable, although with greater difficulty, in standard heterogeneous HetCA simulations (Figure 6.2b). This is why, to characterize phenotypes we found it useful to measure these cycles as well as their irregularities. At every iteration $t > 8$ of the homogeneous genotype test, we compared the sequence of states S_t and S_{t-1} of each cell to its former sequences during the 8 previous iterations of the CA. We assessed whether this sequence of two states was repeated and with what periodicity $p = \min\{p \in [2, 7]\}$, i.e. such that $(S_t, S_{t-1}) = (S_{t-p}, S_{t-p-1})$. We used a sequence of two states because, if the genotype of a cell adopts a stable strategy, i.e. repeats a sequence of states, this sequence must contain a minimum of two states in order to ensure the survival of

the genotype—the quiescent state and one of the living states. We chose to limit the comparison to the 8 previous iterations in order to reduce the computational cost and because of the limit of 7 consecutive live iterations before decay, for a successful regular phenotype, a maximum periodicity of 7 iterations for the quiescent state. We performed this measure only if there was at least one living state and no decay among the last two states. We report, for each iteration t of the CA, the phenotypic disturbance

$$P(t) = \sum_{p=1}^{7} \left| \frac{N(p,t)}{N(t)} - \frac{N(p,t-1)}{N(t-1)} \right| \tag{6.4}$$

where $N(p,t)$ is the total number of cell with periodicity p at iteration t. This measure is rough but interesting because, unlike phenotypic differences, it is not directly based on states and therefore is less likely to be correlated to a states probability of propagation.



(a) (b)

Figure 6.2: Examples of 6-step survival strategies: (a) Genotype extracted from a HetCA simulation in a stable environment (SE) with random homogeneous initialization. (b) Genotype produced by short-cycle fluctuations (ScF).

6.4 Results

6.4.1 Genotype Size and Genotype Mutations

Figure 6.3 shows the evolution of genotype size under the 4 types of environmental fluctuations. The imposed size limit of 50 program statements tends to blur the differences between the different scenarios since most simulations converge to this limit. Yet, ScF clearly restricts the size of the genotypes more severely than SE, LF and SF. This size reduction by ScF could be a way to increase the impact of genotypic mutations on the phenotype. This is because, even though the number of statements potentially affected by mutations in LGP increases proportionally with genotype size, and could have a larger effect on the phenotype, there can also be a "buffer" effect brought by information redundancy in longer genotypes, which would in fact stabilize the phenotype. Hence, mutations in more compact genomes might end up being more impactful[1].

Figure 6.4 depicts the number of mutations separating the current MCG from individuals created during initialization. It is not surprising that more mutations are selected for when environmental fluctuations are introduced, and their number seem to depend more on the strength of these fluctuations (i.e. the contrast between two successive propagation probability patterns E) rather than their periodicity. We also note that the proximity of SF and ScF indicates that their differences in size are not explained by differences in the number of selected mutations.

6.4.2 Phenotypic Comparison

Figure 6.5 shows the phenotypic comparison $\sigma(t_1, t_2)$ between dissimilar environments, i.e. at t_1 and $t_2 = t_1 - \Delta t$ such that $E(t_1) = E(t_2 + f)$. One can see that the impact of environmental fluctuations decreases quickly for ScF while it remains very high for other types of environmental fluctuations. We also note that the phenotypic difference of ScF remains most of the time lower than the phenotypic differences of the SE. This suggests the selection of a single phenotype, robust in

[1]This possibility is strengthen by an analysis of the effective length of most common individuals that we couldn't report here.

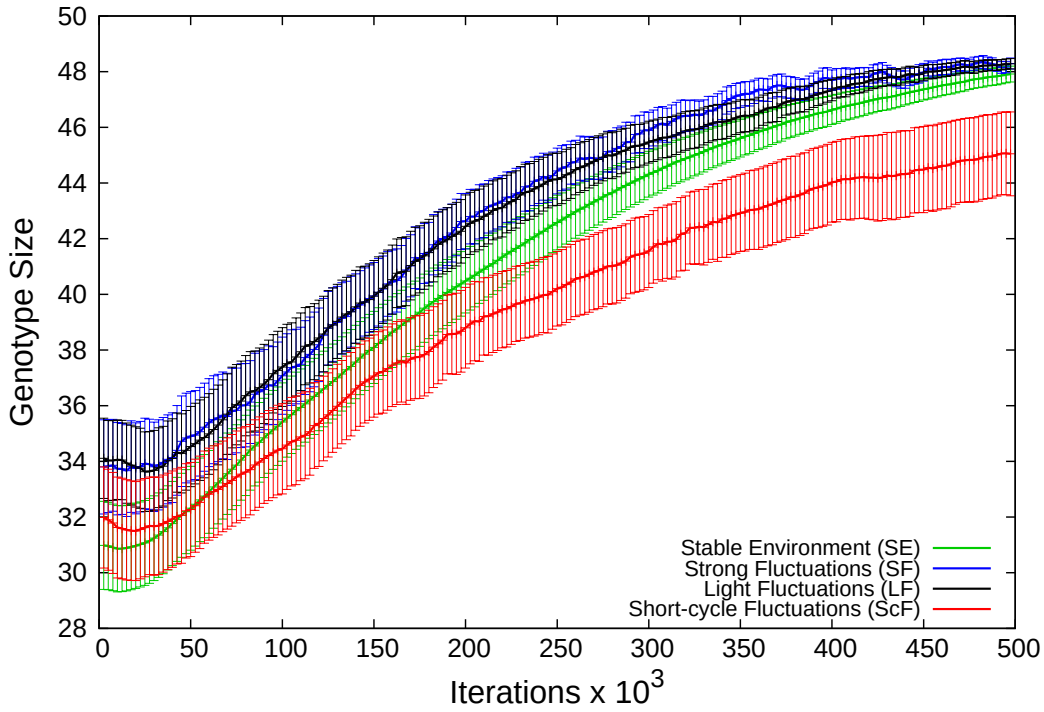


Figure 6.3: Average (± SEM) genotypes size in number of LGP statements.

both environments. Figure 6.6 shows $\sigma(t'_1, t'_2)$ between similar environment, i.e. such that $E(t'_1) = E(t'_2)$. It can be observed that phenotypic differences in LF and SF are much lower than they are in Figure 6.5.

6.4.3 Phenotypic and Genotypic Diversity

Figures 6.7 and 6.8 depict the average phenotypic and genotypic diversities, previously described in Section 6.3.3, ${}^2D_p$ and ${}^2D_g$. The generally low phenotypic diversity of ScF suggests the existence in this configuration of a dominant phenotype, which remains rather stable over time, whereas the relatively high phenotypic diversity of LF and SF combined with a relatively low genotypic diversity might suggest the existence of strong phenotypic selection, hence some form of plasticity.

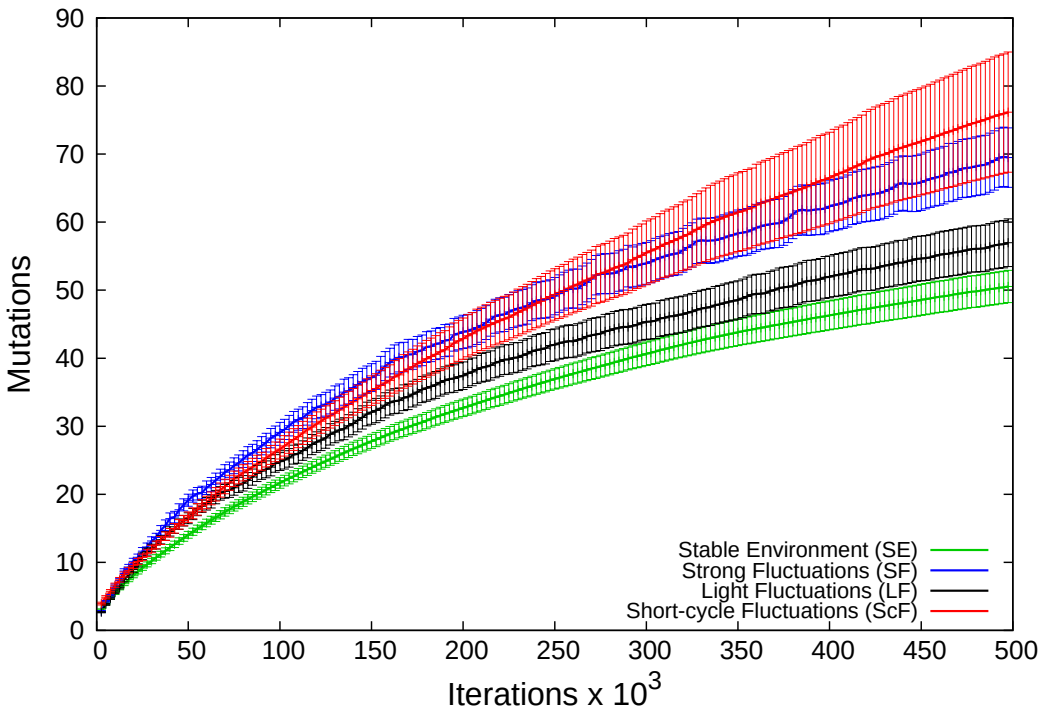


Figure 6.4: Average (± SEM) number of ancestral mutations involved in producing current MCG.

6.4.4 Success Rates of the Homogeneous Test

Success rates of genotypes in different homogeneous simulations are reported in Figure 6.9 using a normal approximation with a 95% confidence interval. The fact that SE offers the lowest challenge is not surprising. Similarly the fact that SF is the least conducive to success is also expected. A comparison of the levels of difficulty between LF and ScF is less clear, however, since ScF performs significantly better in its own settings while on the contrary all other tested fluctuations are slightly more efficient in LF, it is probably reasonable to think that LF is a more difficult environment than ScF. It is also noteworthy that individuals from LF and SF seem relatively robust in various environmental configurations, while those from ScF seem fragile outside the environmental conditions in which they evolved. Moreover, among genotypes collected from iteration 102,500, these same

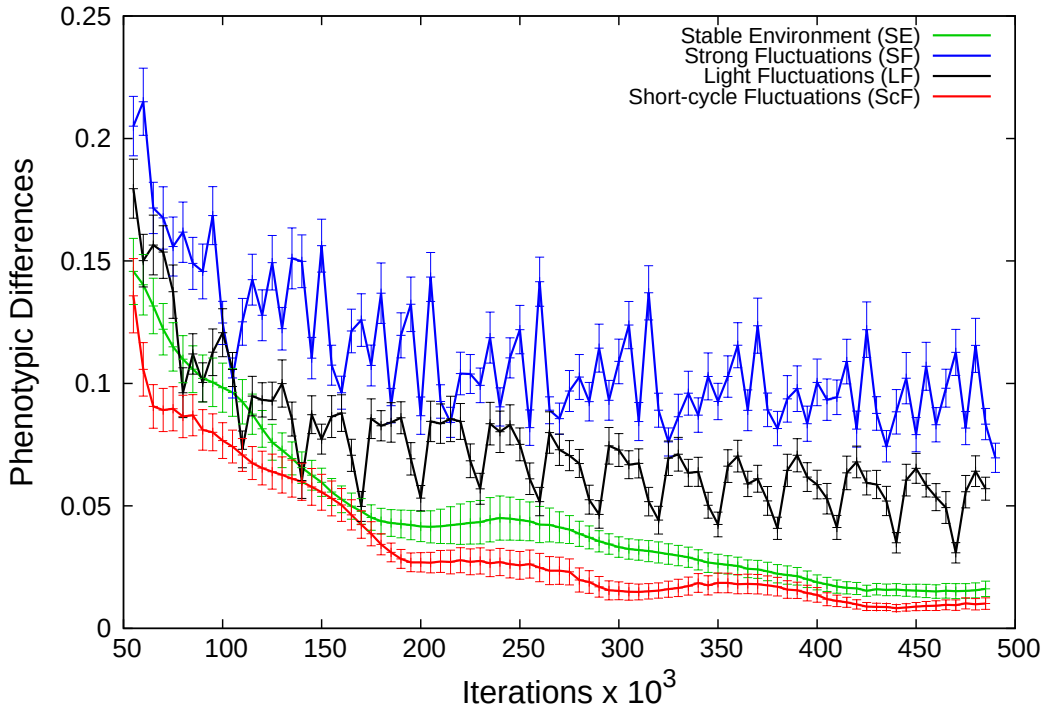


Figure 6.5: Phenotypic comparison average (± SEM) σ: similar environments.

individuals are the only ones that do not reach a 100% survival ratio in a stable environment.

6.4.5 Ending Iteration

Figure 6.11 displays the last iterations reached by living cells of homogeneous runs of genotypes collected at iteration 100,000 and 500,000. Note that ScF genotypes failures are concentrated around iteration 15,000 in LF homogeneous tests and 25,000 in SF homogeneous tests. This corresponds for these two configurations to the first environment for which $K(S_3) = 0$, whereas $K(S_3) = 1$ for all distributions E in ScF. But we also see that some of the genotypes from ScF fail during the early iterations of the homogeneous test regardless of the environmental fluctuations, including SE and ScF. This might imply that the ecosystem resulting

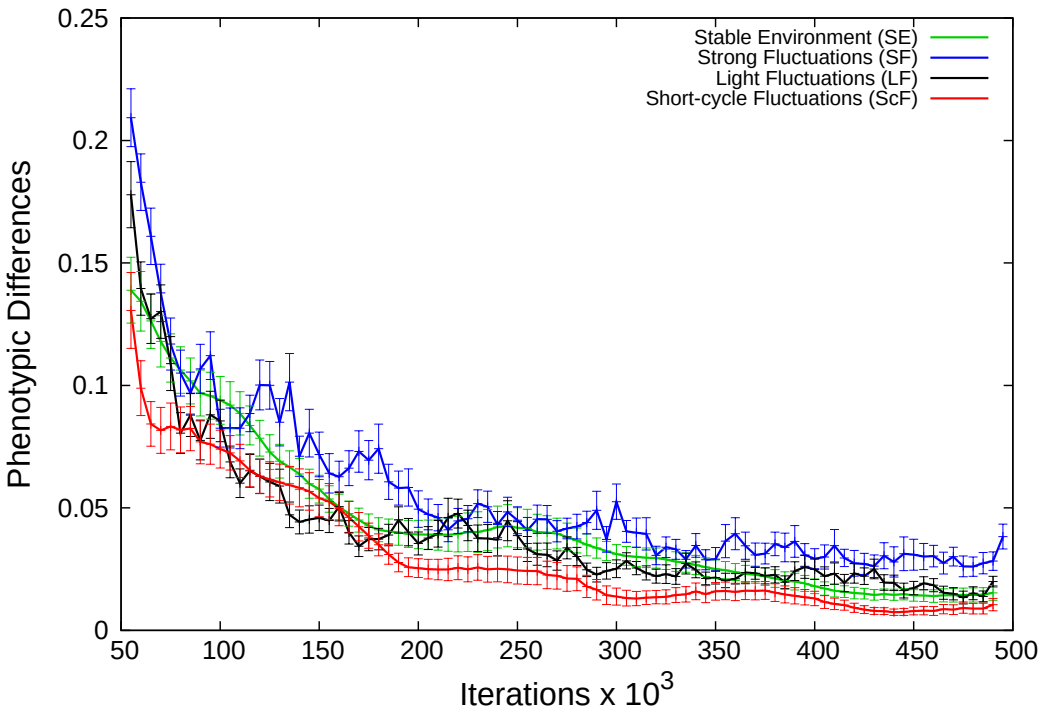


Figure 6.6: Phenotypic comparison average (± SEM) σ: dissimilar environments.

from the evolutionary history of the individuals plays a key role in their ability to survive in ScF.

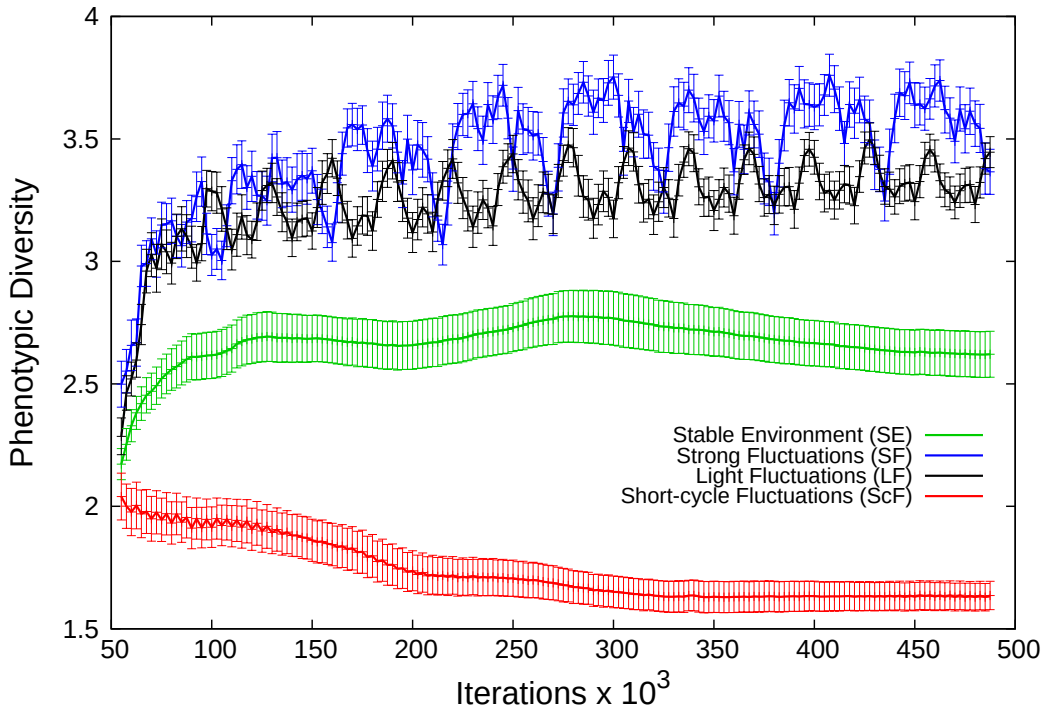


Figure 6.7: Average (± SEM) phenotypic diversity.

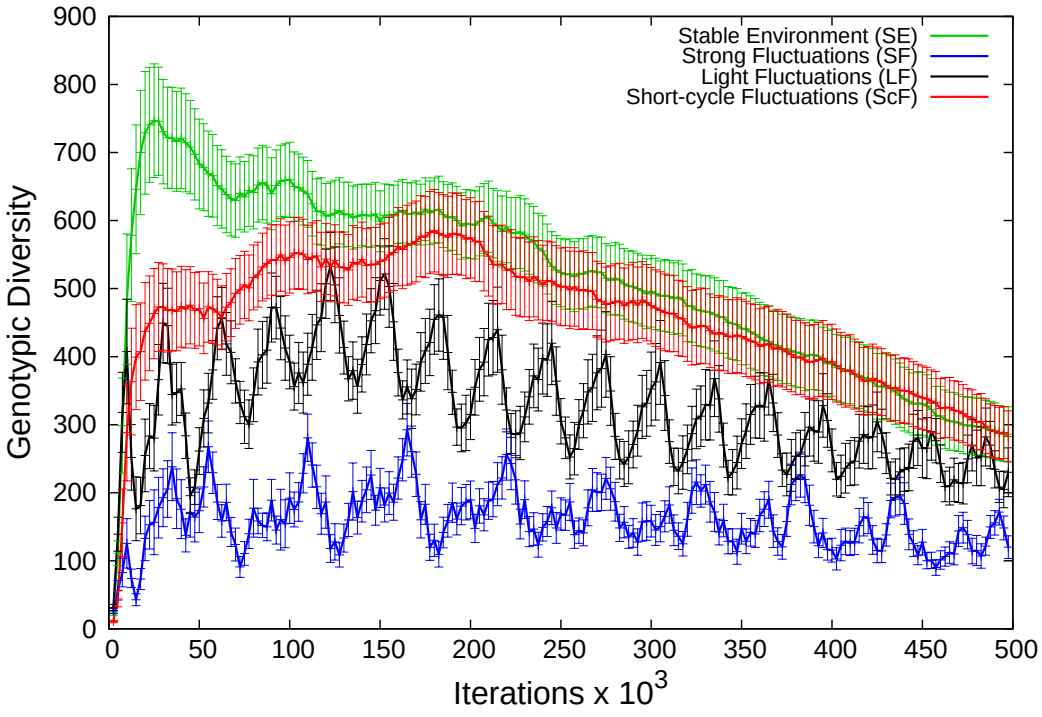


Figure 6.8: Average (± SEM) genotypic diversity.

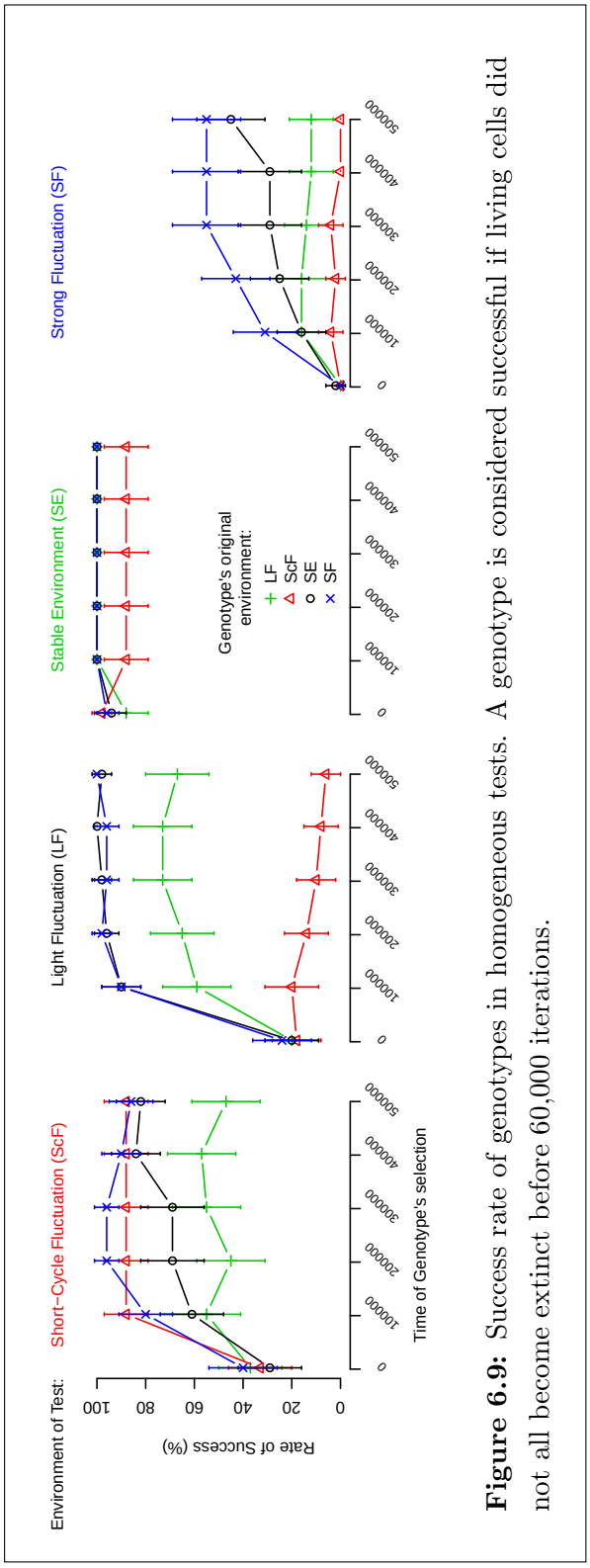
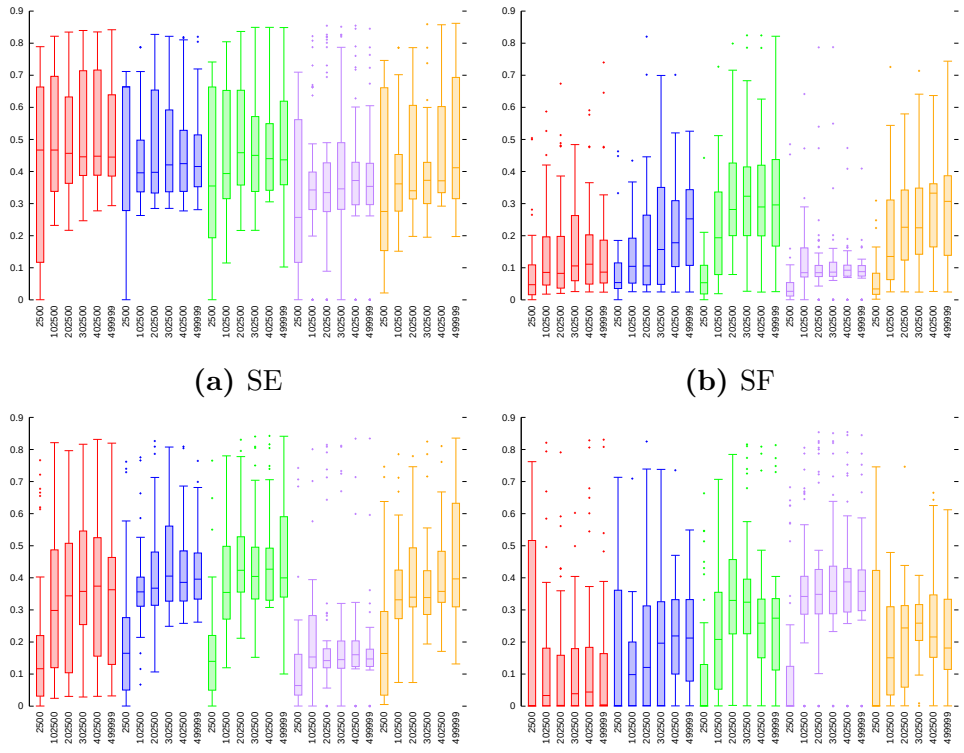


Figure 6.9: Success rate of genotypes in homogeneous tests. A genotype is considered successful if living cells did not all become extinct before 60,000 iterations.

6.4.6 Phenotypic Densities

Figure 6.10 depicts the density of phenotypes, similarly to Chapter 4, as measured in homogeneous tests.



(a) SE (b) SF

(c) LF (d) ScF

Figure 6.10: Densities of Genotype: Each genotype density is processed in four possible different environments.

6.5 Analysis

6.5.1 Environmental Transitions

Figure 6.12 depicts the average transition between environments in different homogeneous runs. To compute this typical transition we averaged the phenotypic disturbance P over iterations $[t-40, t+40]$ for all $t \geq 5,000$ and $t \in T_r$, where T_r is

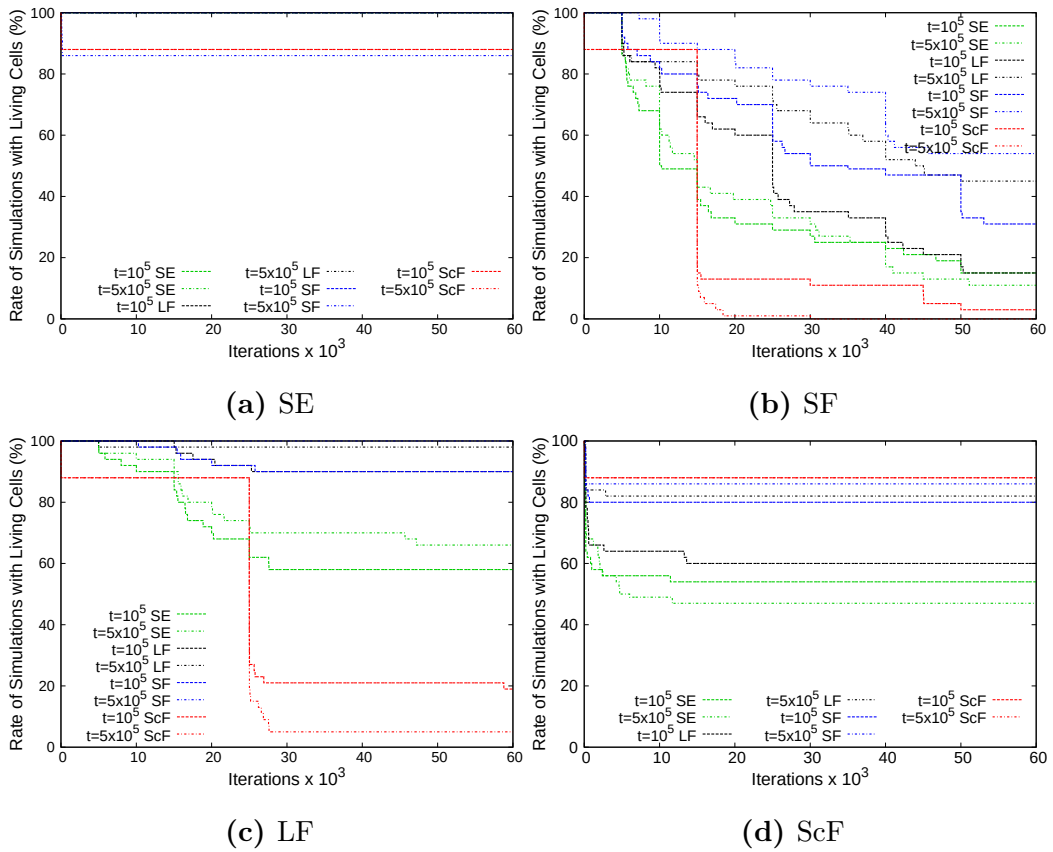
the sequence of iterations of run r at which a transition between environments effectively occurred, i.e. for which $E(t) \neq E(t+1)$. Figure 6.12d shows the average transitions of ScF genotypes collected at the 6 aforementioned time steps {2,500, 102,500, ..., 500,000} and subjected to a homogeneous ScF test. It shows that phenotypes of the genotypes collected later in the evolutionary process are less sensitive to environmental fluctuations. Conversely, as reported in Figure 6.12b, the phenotypes of genotypes from SF keep the same high sensitivity regardless of the iteration at which they were collected. Finally, Figures 6.12a and 6.12c compare the average transitions in homogeneous ScF and SF tests with genotypes collected at iteration 500,000 from the four different configurations. Again, it can be observed here that the phenotype of ScF is much more stable than the others in its original environment. On the contrary, it is very sensitive to transitions in SF.

6.5.2 Phenotypic Diversity

The phenotypic diversity measured in Figure 6.7 can also be observed relatively easily by visual inspection of the CA as displayed in the few screenshots of Figure 6.13. First, LF and SF are visibly different from ScF. These two groups diverge substantially in texture and also clearly differ from SE. Individuals from ScF seem to produce stable and robust phenotypes in any environment encountered within a ScF scheme. Their adaptations appear essentially created by genotypic mutations. They are also very dependent on their original ecosystem, sometimes distinctively as depicted in Figure 6.14, and consequently are not robust in other types of fluctuations, where the effect of mutations is probably enhanced by the reduced size of the genotypes as mentioned previously. By contrast, individuals evolved within LF or SF have high phenotypic diversity, and it seem likely that phenotypic selection occurs despite the fact that their lower genotypic diversity.

6.6 Conclusions

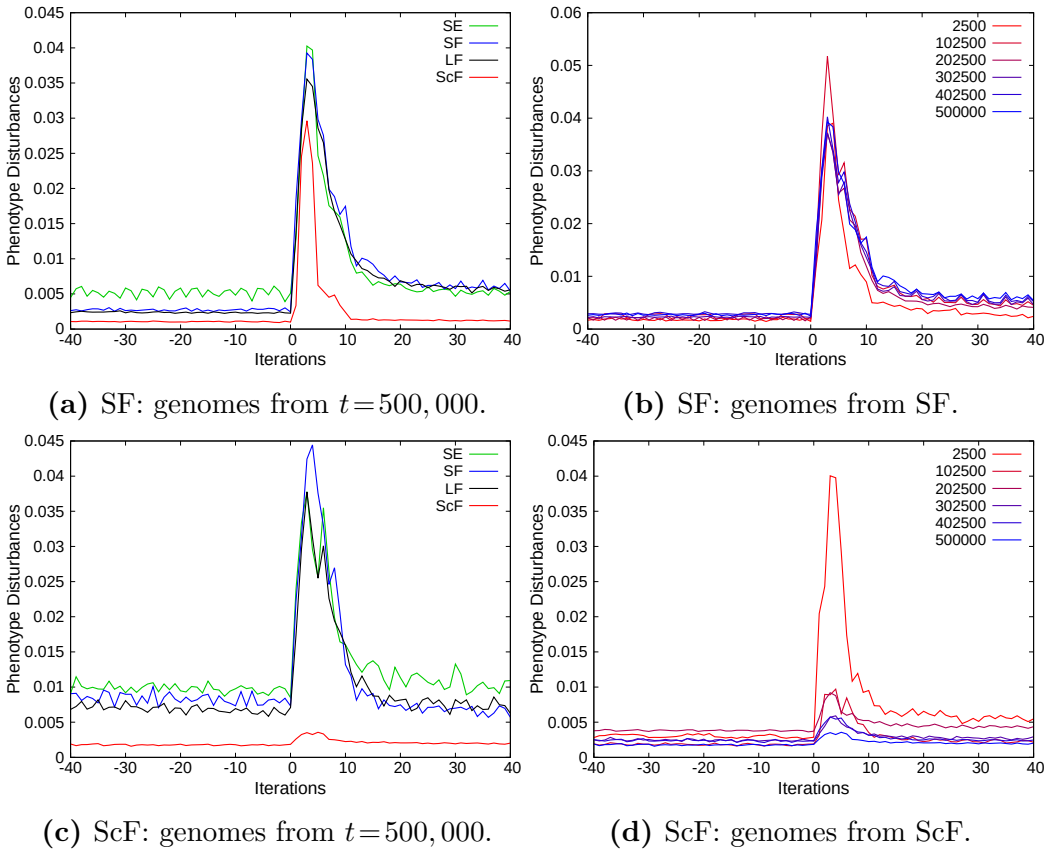
The phenotypic selection found in this series of simulations evoke the multilevel selection model described by Jablonka et al. [2005c]. Among the three tested



(a) SE **(b)** SF **(c)** LF **(d)** ScF

Figure 6.11: Last iterations reached by living cells of homogeneous runs of genotypes from iterations 100,000 and 500,000.

environmental fluctuations, the LF and SF simulations show the most similarities with this model. In ScF, the most successful evolutionary strategy seems to involve small genotypes, which could be favored for their ability to maximize the phenotypic impact of mutations. Furthermore, the inability of most ScF individuals to survive outside of the ecosystem resulting from their evolutionary history is reminiscent of the impossibility of saving species by the unique preservation of their DNA mentioned in Jablonka et al. [2005c]: "You would have to reconstruct the community, and often these communities are very old, with historical memories that are stored in their epigenetic and behavioral systems. These are part of their "identity," part of their stability. You cannot freeze these memories: they



(a) SF: genomes from $t = 500,000$.

(b) SF: genomes from SF.

(c) ScF: genomes from $t = 500,000$.

(d) ScF: genomes from ScF.

Figure 6.12: Phenotypic disturbance: Average transition between environments in different types of homogeneous tests.

have to be maintained and transmitted through use, so you cannot reconstruct the communities from their component parts." (*ibid.*, p. 363). However, these experiments cannot determine whether the main indicator of differences between LF and SF on one side and ScF on the other side is the duration of the environmental cycles or the number of different types of environments, and further investigation is needed.

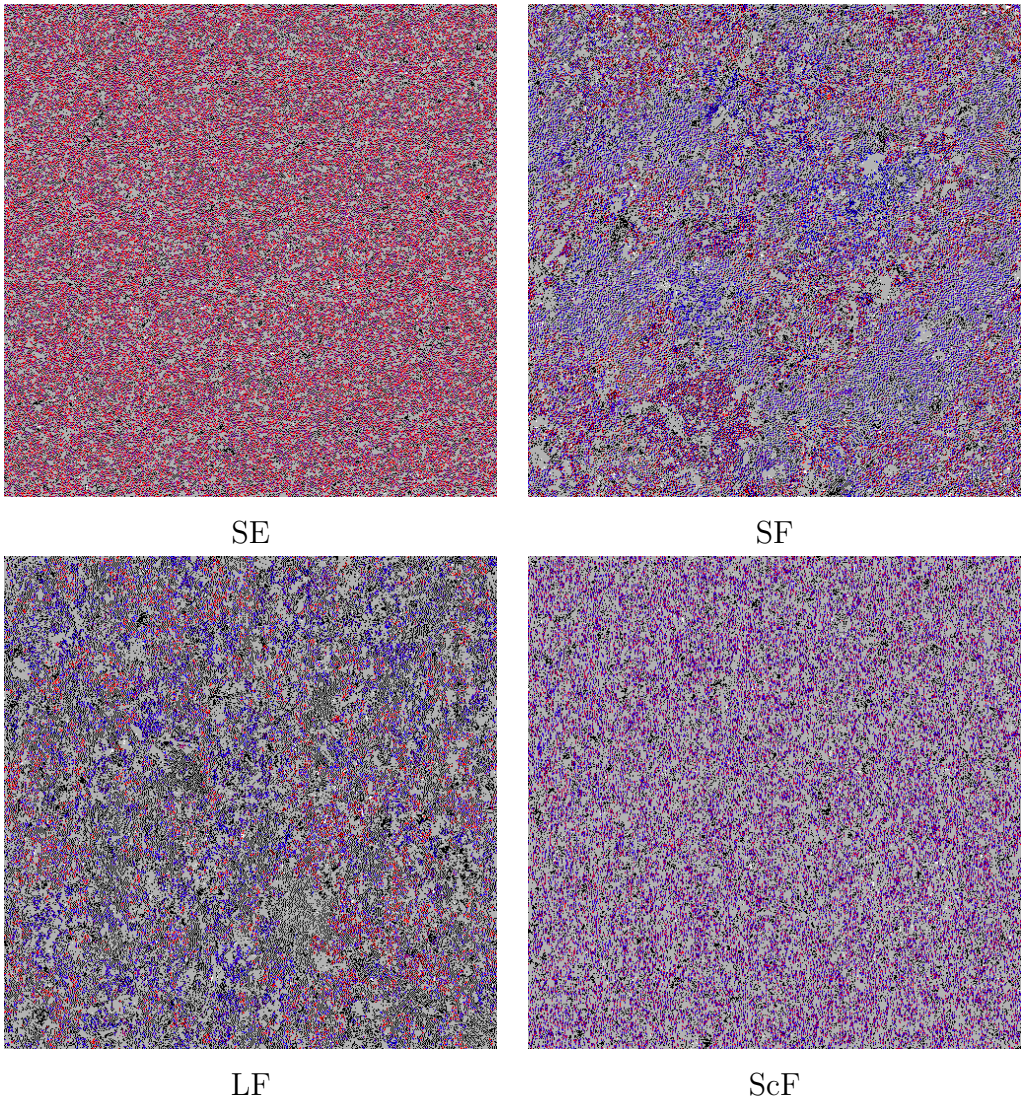
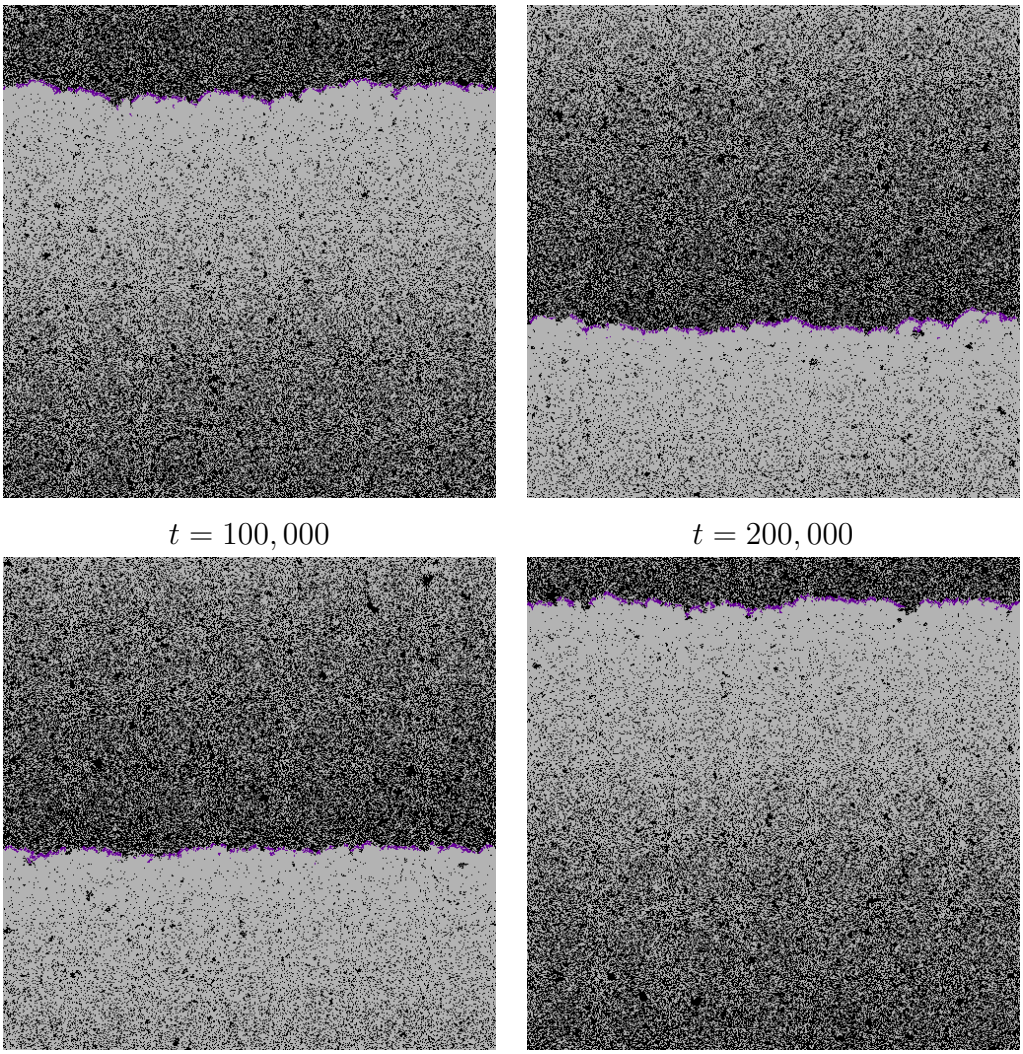


Figure 6.13: Screenshots of the CA. Grid state distribution (phenotype) at iteration 495,000 for the four different configurations. Each cell state is represented by a different color. Black and grey represent cells in the *decay* and *quiescent* states, respectively. Shades of blue, red and purple represent the living states.



$t = 100,000$ $t = 200,000$

$t = 400,000$ $t = 500,000$

Figure 6.14: Original ScF simulation: a distinctive "wavy" phenotype, very stable over time, produces genotypes that fail in the early iterations of the homogeneous test.

6. EVOLUTION OF HETEROGENEOUS CELLULAR AUTOMATA IN FLUCTUATING ENVIRONMENTS

7

A New Wave: A Dynamic Approach to Semantic Genetic Programming by punctuated equilibrium

7.1 Introduction

In this chapter we try to enhance the Wave method proposed initially in Chapter 5. While empirical results necessitate the Wave approach, its design is strongly rooted in lessons from biological evolution. One such idea is *punctuated equilibrium* Gould [1972] which observes that evolution in the living world is not linear, rather it alternates between a succession of periods of *stasis* and rapid change. Wave seeks to emulate this dynamic evolutionary environment by simulating periods of rapid change using mechanisms inspired by evolutionary biology. These mechanisms are *saltationism* Blackburn [1995] where a saltation is defined as non-gradual modifications of the genotype – *macromutations*, and ecological change Milligan [1986]; both saltationism and ecological change may promote rapid speciation. Wave employs saltationism in an EC context by starting each new period with a new population and by modifying the objective function at the beginning of each Wave period to model an environmental change event. The reintialisation of population and a changing objective function across periods in Wave also em-

brace an idea developed by McClintock McClintock [1993] which suggests that environmentally induced stressful conditions may trigger an adaptive response, leading to massive genetic variations in the genomes of plants. In Jablonka et al. [2005c], the authors suggest that these variations will be triggered by epigenetic mechanisms responsive to the stress, and that this trigger response mechanism may be fairly common in biological evolution.

We hypothesise that modifying Wave further such that to bring it closer to those successful themes fom biological evolution which inspired the original wave concept may allow us to simulate a dynamic evolutionary environment which may yield improved performance.

In the punctuated equilibrium model of evolutionary biology, *stasis* is defined as a period of little or no evolutionary change in a species. Effective detection of stasis is critical in Wave so as to conserve the computational effort by stopping a period. Once a period stops, the best individual from that period may be incorporated into the joint solution if it improves the joint solution in some fashion. However, if a period stops *too early*, selection may not have sufficient time to effectively optimise, whereas if the period stops *too late*, it only wastes computation cycles while the population bloats and potentially *over-fits* the training data. Thus, in this chapter, we explore a smart, adaptive stopping criterion for the Wave periods using a *validation set*.

In fact, we take a number of self-adaptive measures in this chapter to extend Wave; for this we propose that Wave *automatically* selects the number of periods for each run, the duration of each period and whether or not to use LS in a given period.

Furthermore, because each period starts with a fresh population (similar to a new run), the previously evolved genetic material gets inaccessible; we hypothesise that it may, therefore, inhibit the establishment of mechanisms analogous to *exaptation* Gould and Vrba [1982]: exaptation occurs when features that now enhance fitness were not originally built by natural selection for their current role, the classic example of exaption being the feathers in some theropod dinosaurs initially selected for their thermal regulator function before being subsequently selected for flight. Exaptation eases the evolution of complexity in the living organisms. To facilitate exaptation, in this chapter, we propose to only partially

renew the population at the beginning of each period, thus retaining some of the evolved genetic material.

The chapter reads as follows: Section 7.2 introduces the original implementation of Wave; Section 7.2 describes the new implementation of Wave; Section 7.4 specify the computational setup; Section 7.5 details the results and discusses their significance; and, finally, Section 7.6 concludes the chapter highlighting the achievements.

7.2 Original Wave

In this section, we briefly describe the original Wave implementation that we already further described in chapter 5, that is, population renewal strategies and the use of a validation data set.

Any period, optimizing over a current target set t of size N, is considered to be *successful* if the best trained individual resulting from the period generates a function f with a RMSE at least better than a neutral individual represented by 0, as below.

$$\sqrt{\sum_{i=1}^{N}(f_i - t_i)^2} < \sqrt{\sum_{i=1}^{N}(t_i - 0)^2} \tag{7.1}$$

When a *successful* period terminates, the new target data set t' for the next period is defined such that $t' = t - f$. However, if the period is not successful, the new period starts with an unchanged target $t' = t$. Therefore, $f_{joint} = \sum_{j=1}^{n}(f_j)$ where f_{joint} is the joint solution of a Wave run, n is the number of successful periods and f_j is the function produced by the jth successful period.

7.3 Proposed Method

Figure 7.1 depicts the logic of both the original and enhanced versions of the Wave system. In this section, we describe various components of the enhanced paradigm. Some of these techniques such as smart stopping of periods apply to all Wave configurations, whereas others are used only in specific setups. Table 7.2 outlines these details.

7.3.1 Partial Population Renewal

Previously, the entire population was renewed at the beginning of a new period. In this study we only renew 80% of the original population[1]; the 20% of individuals that are not replaced are randomly selected. As discussed earlier, the fresh individuals at the start of a new period do not necessarily suffer a disadvantage when pitted against the evolved peers from the previous period: this is because the objective function has changed[2]. Unless there is a strong correlation between the new and old objective functions, the old individuals do not carry advantages or disavantage[3]. Therefore, unlike the approach taken in Pagie and Hogeweg [1997], we do not require specialised mechanisms such as age of the genotype to preserve the disadvantaged individuals. Of course, pre-existing individuals who may have had good fitness in the previous period, may initially be unable to solve the redrafted problem. However, we believe that maintaining this genetic material into the gene pool offers the possibility of the emergence of phenomena such as exaptation or reuse of existing modules, while at the same time allowing the new genetic material to flourish.

7.3.2 Smart Stopping of Wave Periods

Previously, in Chapter 5, a period was terminated when the Wave system determined that the period was not significantly improving the fitness – if the improvement over the two most recent generations was less than 0.5% of the improvement over the three generations previous to the two most recent generations. The two following conditions 7.2 and 7.3 were formalising the period stopping criteria:

$$g_c > g_m \tag{7.2}$$

$$(B^F(g_c) - B^F(g_c - 2)) \leq (B^F(g_c - 2) - B^F(g_c - 5))/200 \tag{7.3}$$

[1]First, we explored replacing only 50% of the population; however, the population converged very quickly without improving the results.

[2]Renewing the target set as $t' = t - f$ akins here a stressful ecological change.

[3]A possible extension of this work is to measure this correlation and adjust the relative proportion of new and old population members accordingly.

where g_c is the current generation, g_m is a minimum number of generations before a period stops and $B^F(g_c)$ is the best training fitness at generation g_c.

In this work we investigate a new strategy whereby, in every generation, we also compute the fitness of the best trained individual on a validation set. Suppose O(f,d) represents the objective fitness (or fitness value) of an individual function f on a data set d, then $O(f_p(g), v_p)$ is the objective fitness of the individual $f_p(g)$ evaluated over the validation data set v_p; note $f_p(g)$ is the best trained individual at generation g of the period p. Considering we are minimizing the fitness, we then terminate a period if $O(f_p(g), v_p) > O(f_p(g-1), v_p)$. In other words, we stop a period *whenever* the validation fitness degrades. Granted, that the validation fitness may oscillate in the future, if the period is allowed to progress; however, our preliminary investigations revealed that waiting for that to happen did not improve scores on the unseen test data sets.

Often, in our experience, especially when LS is enabled, the validation fitness does not change over many generations, that is:

$$O(f_p(g), v_p) = O(f_p(g-1), v_p); \tag{7.4}$$

this represents stasis. We decide to halt the current period if stasis continues for more than 5 consecutive generations; however, if at any time, an individual that is *eligible* to be a part of the joint solution is produced, we allow the current period to have another 25 generations[1] of stasis[2]. The eligible individual is the one that produces the best validation fitness thus far, across all the periods. Formally,

$$O(f_p(g), v_p) < \min_{i=0\ldots p-1} O(f_i, v_i). \tag{7.5}$$

The same criteria is reapplied, as below, at the end of the period to check whether this period has produced *any* solution that is eligible to be a part of the joint solution:

$$O(f_p, v_p) < \min_{i=0\ldots p-1} O(f_i, v_i) \tag{7.6}$$

where f_p is the best trained individual at any generation during the current period p that also has the best validation fitness. Due to the stopping conditions

[1]These choices appear adhoc but they are based on some empirical success; however, they are not necessarily optimal.

[2]Note, we still stop the period at any time the validation fitness degrades.

described above, this individual is always present in the penultimate generation of the period. Note, the eligibility criteria to get into the joint solution described here is different from that in the previous implementation of Wave described in Section 7.2.

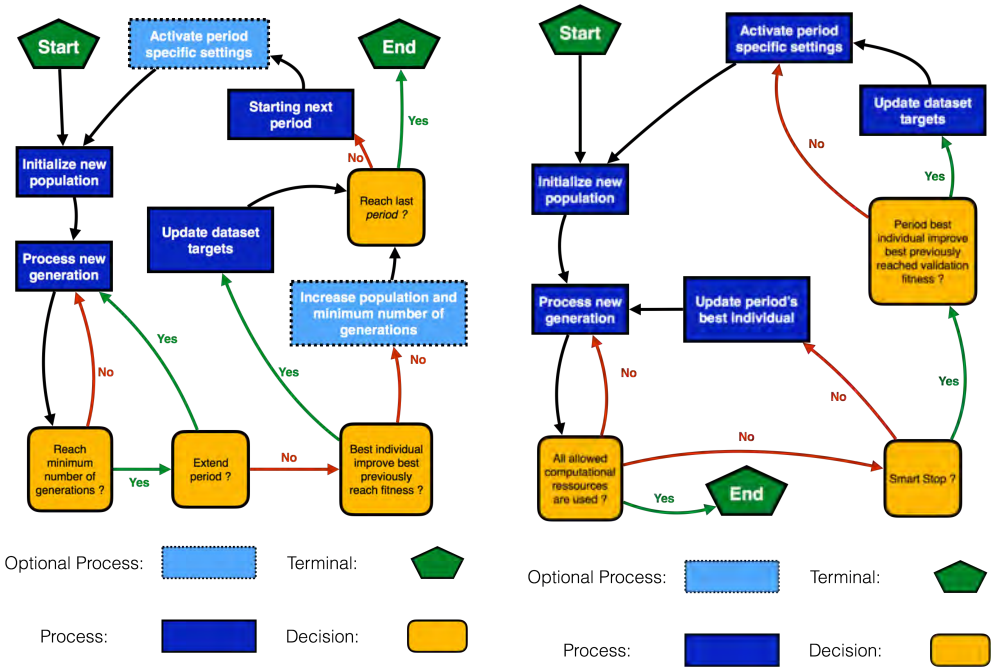
7.3.3 Adaptive Linear Scaling

Previously, the most effective Wave configuration involved alternating between periods with and without LS. However, our exploratory runs indicate that the success rate of periods with and without LS is not uniform. We also noticed that the success rate of LS is proportionally much higher in the first period than in the last period. Moreover, all this is highly dependent on the problem under consideration. So, in this work, we have chosen not to alternate between periods with and without LS. Instead, at the beginning of each period, the system chooses randomly between these two modes based on their success rate in preceding periods. To achieve this the probability of activating LS for a given period is $P = \frac{r_{LS}}{r_{LS}+r_{\neg LS}}$ where $r_{LS} = \frac{LS_{suc}+1}{LS_{tot}+1}$ is the approximate success rate of periods using LS; LS_{tot} is the total number of periods that used LS *thus far* and LS_{suc} is the number of successful periods that used LS; $r_{\neg LS}$ is the approximate success rate of period not using LS computed the same way. It is important to note that these counts of various kinds of periods stabilise as the number of periods grow thus giving a more reliable measure of the probability of activating LS.

7.4 Experiments

7.4.1 Benchmarks

We compare Wave with both EC-based and non-EC-based machine learning methods. We will use standard GP, both with and without LS to benchmark both the computational cost of evolution and accuracy of the resulting models. While we use GSGP Moraglio et al. [2012] and Multiple Linear Regression (MLR) solely to benchmark the accuracy of the achieved results. This is because GSGP caches evaluations from the first generation and does not evaluate new material



(a) Original Wave

(b) Enhanced Wave

Figure 7.1: Flow of Wave algorithm. The period stop criterion is no longer based on a minimum number of generations.

afresh; instead, this work does not cache evaluations yet and therefore cannot compare with GSGP. Also, MLR is well known to be far cheaper than GP yet has recently been proposed as a tough benchmark for GP to beat Arnaldo et al. [2014].

7.4.2 Problem Suite

For this study we have used three multi-dimensional data sets from the UCI Machine learning repository Bache and Lichman [2013] together with two mathematical functions. The following data sets from the UCI repository are used:

- **Concrete Strength** where the objective is to predict the compressive strength of concrete and data set includes 1030 instances each having 8 inputs.
- **Yacht** where the objective is to predict the hydrodynamic performances of a yacht, and data set includes 308 instances each with 7 inputs.
- **Powerplant** where the task is to predict the net hourly electrical energy output of a power plant and data set includes 9568 instances and 4 inputs.

The two mathematical functions chosen are:

- **Poly-10 Poli [2003]** $y = x1*x2+x3*x4+x5*x6+x1*x7*x9+x3*x6*x10$ and
- **Div-5 Vladislavleva et al. [2009b]** $y = \frac{10}{5+\sum_{i=1}^{5}(x_i-3)^2}$.

For each of the two mathematical functions, 500 data points in the range $[0; 1]$ are randomly generated. For each run we randomly split the given data set into three subsets of equal size for training, validation and testing purposes, that is, $N_{training} = N_{validation} = N_{testing}$. For the benchmark algorithms, because they do not use periods and therefore cannot use a validation set the way we do here[1], the instances in the validation set are added to the training set, the numbers of testing instances remains of course the same between the benchmarks and Wave.

[1]Exploratory experiments also indicated that standard GP does not benefit from a validation data set using one third of the total available datas on the data sets used in this chapter.

Previously, in Medernach et al. [2015a] the Wave system did worse than standard GP on the Yacht problem. As Wave had done better on all of the other data sets we are interested in understanding the cause of the difference in performance. We suspect that the relatively small size of the Yacht data set (308 data points) may have contributed to higher over-fitting. In order to examine this hypotheis we select the Concrete Strength data set on which Wave performed well and create a smaller version of this data set to use as a surrogate and to see if, when using the same training and testing size that for Yacht, the performances degraded and become similar to that observed on Yacht. Thus, we conduct separate runs on smaller training sets from the **Concrete Strength** problem where the $N_{training} = N_{validation} = 103 \approx 308/3$. The remaining points make the test set such that $N_{testing} = 824 \approx 1030 - 206$. We will refer to this particular configuration as the **Small Concrete** configuration from now on.

7.4.3 Common Parameters

The parameters described in Table 7.1 have been chosen to be the same as in Medernach et al. [2015a] and are consistently applied across all Wave and other GP benchmark experiments. We process 45 runs for each EC configuration for each data set. Because of its low computational cost we performed 100 MLR runs on each data set.

Parameter	Value
Population	500 individuals
Replacement Strategy	Generational
Operator Probabilities	Xover: 0.9; Point mutation: 0.1
Tournament Size	10[a]
Max. depth	17
Max. size	100
Functions set	$+,-,\times,\div$(Protected division)
Terminal set	Inputs and constants -1.0, -0.5, 0.0, 0.5 & 1.0
Fitness	RMSE
Initialisation	Ramped half & half
Max. initial depth	8
Mutation Step (GSGP Only)	0.1[b]

[a] A relatively high tournament size but recently successfully used in Goncalves and Silva [2013] and Muhammad Atif Azad et al. [2014]; we kept the default tournament size of 3 on GSGP.
[b] This is the default mutation step in the implementation proposed in Castelli et al. [2014].

Table 7.1: GP Parameters for Wave, GP benchmarks and GSGP.

7.4.4 Experimental Configurations

In this chapter we study two Wave configurations using both adaptive LS and smart stopping: the first configuration does not use any optional settings; the second one uses partial population renewal. All of these configurations and bench-

marks are described in Table 7.2.

Name	Wave	Renew [a]	LS
GP with linear scaling:			
GP:LS	Off	NA	On
Standard GP:			
GP	Off	NA	Off
Multiple linear regression:			
MLR	NA	NA	NA
Geometric Semantic GP[c]:			
GSGP	NA	NA	NA
Wave:			
Wave	On	100	Alt[b]
Wave with partial population renewal:			
Wave:PPR	On	80	Alt[b]

[a] Proportion of population renewed at the beginning of each period.
[b] Alternation between LS and non-LS periods, based on LS and non LS success rate.
[c] As per Castelli et al. [2014].

Table 7.2: Different Wave and benchmarks configurations.

7.4.5 Performance Metrics

It is not relevant to compare the Wave approach with conventional GP simply by measuring the performance of each on the same number of generations. This is because the population is regularly renewed in Wave so that individuals do not have the same opportunity to bloat as they do in conventional GP. Consequently, each generation is generally less computationally expensive to process with Wave. Therefore, we believe that it would not be equitable to take an approach of terminating runs after the same number of generations for each method, nor to compare statistics measured every generation.

The ratio between testing fitness and nodes processed is a measure of the performance of a GP used in particular when the material processed during each iteration is not the same as in a normal fitness evaluation as in [Azad and Ryan, 2010a]. That's why we choose to use the total cost C of computed nodes as a measure of computational cost of a run. Typically, the computational cost of a

Wave run will be:

$$C = \sum_{g=1}^{g_{tot}} (\sum_{i=1}^{pop} N_{training} \times S_{I_{gi}}) + N_{validation} \times S_{BI_g} \tag{7.7}$$

where *pop* (population size) is 500 individuals, $S_{I_{gi}}$ is the size of the ith individual I_{gi} of the generation g and $S_{BI_{gi}}$ is the size of the best individual at generation g instead, for classic GP:

$$C = \sum_{g=1}^{g_{tot}} \sum_{i=1}^{pop} N \times S_{I_{gi}} \tag{7.8}$$

where N is the size of the full data set. To ensure equity of resources allocated to different systems, we stop a run when its computational cost reaches a maximum value C_{max}. We chose[1]:

$$C_{max} = k \times (N_{validation} + N_{training}) \tag{7.9}$$

where k is a factor chosen so that a run of classic GP with a population of 500 individuals without LS continues for approximately 100 generations. Here we used $k = 4000000$ for **Concrete** and **Small Concrete** and $k = 2000000$ for **Yacht**, **powerplant**, **Div-5** and **Poly-10**. Those values have been chosen to roughly bring standard GP close to 100 generations, except on Powerplant where we choose a smaller value to cater for its high computational cost due to the size of the data set (9568 instances). On all data sets and with all parameters we report median testing fitness on a 95% confidence interval in Figure 7.2.

We observe the statistics at intervals of $C_{max}/100$ and we call these intevals *steps* from now on. Nevertheless we only report a single testing value at every step for MLR and GSGP. Of course MLR is not an EC method, therefore it was not possible to report gradual progress. Also, because, as stated in Castelli et al. [2014], GSGP cached node evaluations, we could not come up with an identical budget for GSGP. However, so as not to put it to a disadvantage, we choose a tougher benchmark from GSGP: we benchmark the Wave results against the *best* test fitness *ever* achieved by GSGP during 2000 generations. Notice, this guards

[1] $N_{validation} + N_{training}$ is constant on the same data set for all possible configurations since the benchmarks that do not use validation data set, append instances of the validation data set to the training data set.

against any over-fitting that GSGP might have experienced and therefore, avoids the pitfall of choosing an inopportune time to report the testing fitness. In fact, as the results show, GSGP mostly achieved its best test fitness well before reaching the end of run; while there is no guarantee that testing fitness cannot improve again, conventional wisdom suggests that overfitting is likelier if we extend the GSGP runs any further.

7.5 Results and discussion

7.5.1 Population Renewal

To study the effects of our partial population renewal setting, Wave:PPR, we report the earliest period at which the ancestors of the last eligible individual were created. As shown in Table 7.3, a majority of the last eligible individuals added as part of the joint solution have an ancestor created in the first period. Considering the number of periods reported in Table 7.4 and that we renew 80% of the individuals each period, this result was not obvious. We suspect that this is possibly partly due the relatively large tournaments used, but it also indicates that the genetic material of individuals evolved to address the first objective function still plays a role later on when the objective function has been modified. This may indicates successful events of exaptation.

Concrete Strengh	**Small Concrete**	**Yacht**	**Powerplant**	**Div-5**	**poly-10**
73%	76%	62%	73%	77%	58%

Table 7.3: Proportion of last eligible individuals included in the joint solution who have an ancestor that was created in the first period.

7.5.2 Self-adaptation

Looking at Table 7.4, we see that the number of periods executed and the proportion of them which use LS varies greatly depending on the problem. The use of smart stopping of periods and adaptive LS allows Wave to automatically

configure itself depending on the problem.

Problem	Total Periods [a]	With LS [a]	Without LS [a]
Concrete Strength	52 (57%)	24 (55%)	29 (59%)
Small Concrete[b]	62 (35%)	29 (33%)	33 (36%)
Powerplant	12 (80%)	06 (75%)	06 (91%)
Yacht	40 (39%)	22 (39%)	18 (38%)
Div-5	26 (45%)	22 (56%)	04 (00%)
Poly-10	74 (25%)	49 (29%)	25 (14%)

[a] Number of periods followed by their success rate in brackets.
[b] Same data set as Concrete Strength but with $N_{training} = N_{validation} = 103$ and $N_{testing} = 824$

Table 7.4: Average number and success rate of periods calculated with and without LS for Wave:PPR.

7.5.3 Overfitting

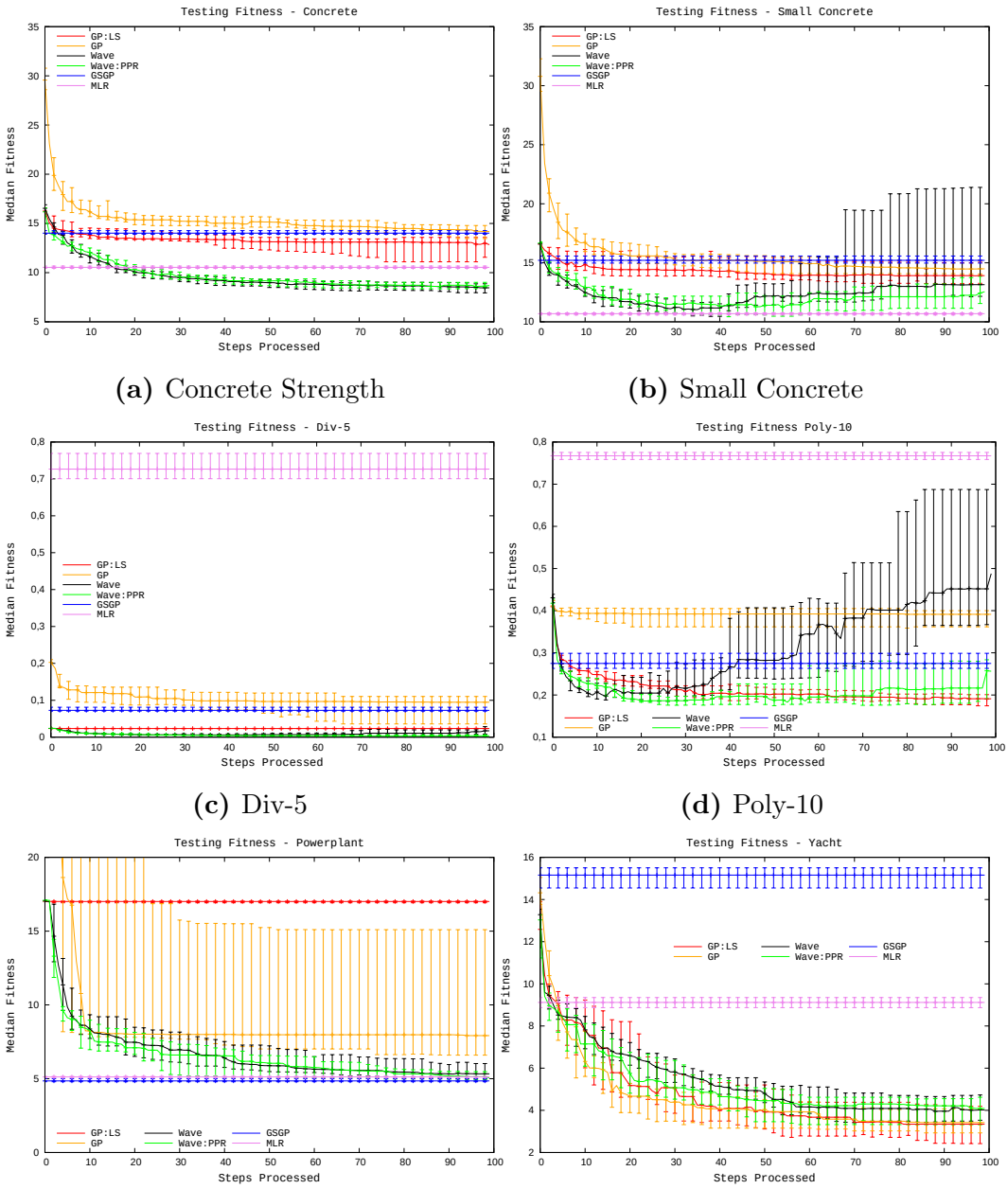
As it can be seen by comparing Figures 7.2a and 7.2b, the size of the training data set influences over-fitting. This assumption is reinforced by the absence of over-fitting in Figure 7.2e where the size of the data set used is relatively large at 9568 data-points. However, Figures 7.2c and 7.2d show that factors other than just the data sizes also affect overfitting; these figures correspond to two data sets, Div-5 and Poly-10, that are identical in size and result from sampling mathematical functions without adding noise and only one them, Poly-10, generate over-fitting.

7.5.4 Computational Cost

Wave:PPR is the only method tested whose performance either exceeds or equals that of classic GP methods across all data sets at every step. Therefore, Wave:PPR outperforms standard GP both with and without LS. Wave:PPR also produces the smallest joint solutions among Wave settings as depicted in Figure 7.3.

7.5.5 Performances

Because of the nature of GSGP and MLR it was not possible to do a step-by-step comparison with Wave. From step 40 onwards Wave:PPR performs as well as MLR on the Powerplant problem and outperforms MLR on all the other data sets except Small Concrete. The best median testing fitness for GSGP individuals



(a) Concrete Strength **(b)** Small Concrete

(c) Div-5 **(d)** Poly-10

(e) Powerplant **(f)** Yacht

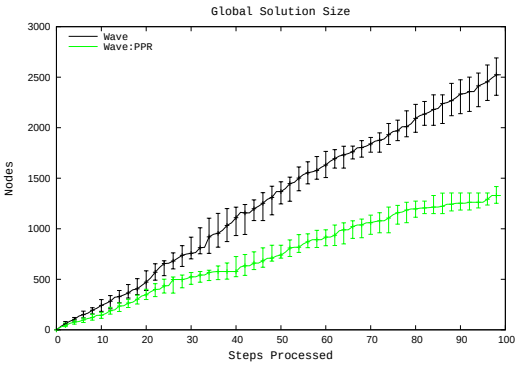
Figure 7.2: Median Testing Fitness of joint solution / best individual - 95% confidence interval.

is reached well before the latest generation on 4 out of 6 data sets (Yacht: gen 1; Small Concrete: gen 1112; Div-5: gen 600; Poly-10: gen 472), so it is very unlikely that GSGP could improve its result with more computational resources (higher number of generations) on those data sets.

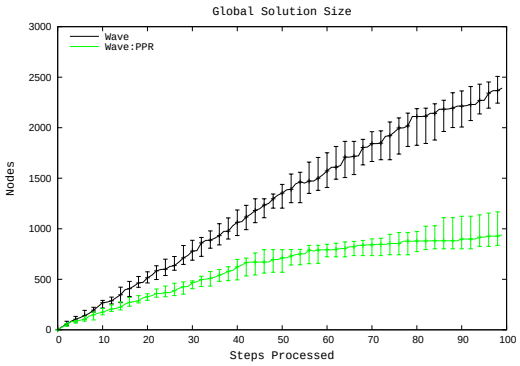
From step 10 onwards Wave:PPR performs as well as or better than GSGP on those four problems. It is hard to conclude on the two remaining problems because both Wave:PPR and GSGP reached their lowest median testing fitness at the end of the simulation which means both of them could probably improve their results with more computational resources. However we note that Wave:PPR outperforms GSGP from step 10 onwards, while on powerplant the final Wave:PPR testing fitness is statistically equivalent to GSGP reported results. It is also worth noting, that the final testing fitness with Wave:PPR is better than or equal to the best results available (the lowest point of the fitness curve - therefore non-potentially damaged by over-fitting) in a previous investigation by Medernach et al. Medernach et al. [2015a]; although, the previous study has some experimental differences in terms of computational cost with the work reported here.

7.6 Conclusions

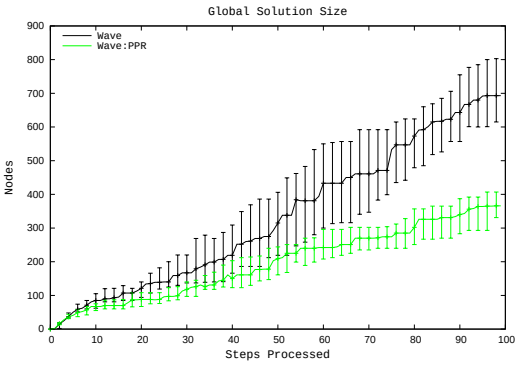
In this chapter, we presented several improvements to Wave GP previously proposed in Chapter 5. Based on the results obtained, the proposed changes make Wave more efficient on the given problems and allow it to adapt to their characteristics. In particular, Wave seems to use mechanisms analogous to *exaptation* by utilising the older individuals evolved during the previous periods to train against a new objective function; this approach, that we termed as Wave:PPR, seems robust as only multiple linear regression (MLR) performs better than Wave:PPR and that too on one of the problems only. Notice, MLR has been flagged as a tough benchmark for standard GP in recent literature. Additionally, the use of a validation set has improved Wave of ability detect periods of stasis.



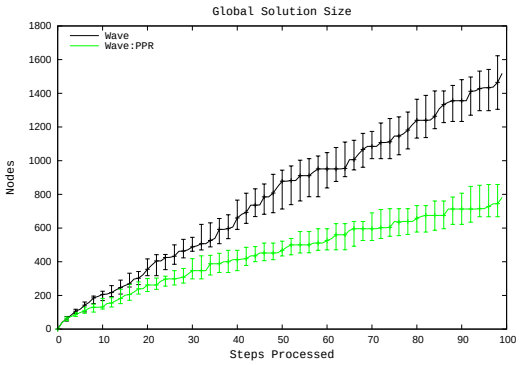
(a) Concrete Strength



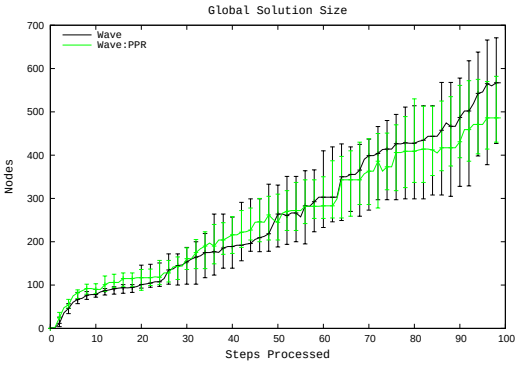
(b) Small Concrete



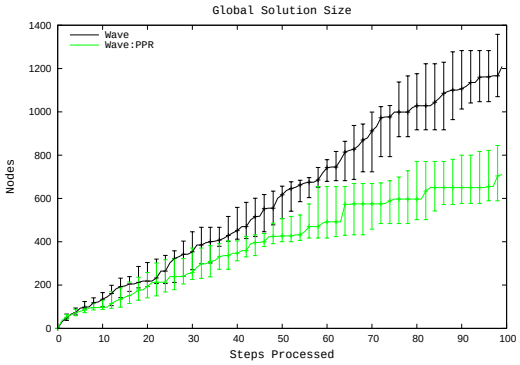
(c) Div-5



(d) Poly-10



(e) Powerplant



(f) Yacht

Figure 7.3: Median Size of Overall solution - 95% confidence interval.

7. A NEW WAVE: A DYNAMIC APPROACH TO SEMANTIC GENETIC PROGRAMMING BY PUNCTUATED EQUILIBRIUM

Part III

Conclusions and Future Directions

8

Conclusions and Future Directions

The aim of this thesis was to study the effects of environmental fluctuations in evolutionary programming. We have accomplished this goal by proposing two new evolutionary simulations, Wave: a Genetic Programming approach and HetCA: a heterogeneous Cellular Automaton aimed at generating an open evolution. The remainder of this chapter is organised as follows: Section 8.1 summarizes the work in this thesis while Section 8.2 presents the potential opportunities offered by the findings of this thesis in terms of future research.

8.1 Summary

To study the effects of environmental fluctuations in the context of evolutionary simulations, in Chapter 1 we posed seven research questions which spawned the following ten objectives:

1. Conduct a survey into the state-of-the-art evolutionary computing and more particularly in GP.

2. Summarize the role played by environmental fluctuations in the development of evolutionary biology.

3. Propose a brief summary of the issues and the history of Alife.

4. Summarize the existing work in terms of evolutionary cellular automata.

5. Creation of an Alife simulation capable of exploring the importance of environmental fluctuations.

6. Test the benefits of changes in the fitness function in Evolutionary Algorithms such as GP.

7. Specify the notion of evolutionary progress and test it in the context of a simulation that does not use an explicit fitness function.

8. Take inspiration from environmental change in biology to build a GP system solving a problem in successive steps using different fitness functions.

9. Test the effect of environmental fluctuations on selection levels in an open-ended simulation.

10. Extend the Random interleaved sampling to test other forms of interleaved.

Background Survey

In order to accomplish the first 4 objectives defined in Chapter 1, a summary of the history of natural selection, focusing on the role that environmental fluctuations play in evolutionary biology, is presented in Chapter 2. This chapter also deals with evolutionary computation and in particular with the different methods of GP as well as with the development of Alife. As in evolutionary biology, a particular interest is accorded to experiments related to environmental variations. In Alife we have focused more specifically on cellular automata as well as on GP which are two methods used during this thesis.

Wave and Interleaved GP

The benefits of environmental fluctuations on GPs are studied through two systems: Wave and Interleaved GP.

The classic Random Interleaved Sampling is extended in Chapter 3, as proposed in objective 10, through two alternate experiments: Interleaved-Size and

Interleaved-Random. This series of experiments show that alternating all generation between two functions of fitness is useful in GP, as proposed in objective 6 of Chapter 1. The research shows that, on the problems studied, the Interleaved-Size approach supports alternation of the two desirable objectives of finding a good solution to the problem and discovering a small solution, while at the same time offering a reduction in over-fitting as least as good as that provided by the Interleaved-Random strategy.

A new GP algorithm, Wave, is proposed in Chapters 5 and 7, fulfilling the 8th objective defined in the introduction. In Wave, a problem is solved by a succession of small GP runs called periods, each of which optimizes the residual errors of the best individual of the previous run. Each period potentially uses different parameters and a different fitness function. Improvements to the original method have been proposed in Chapter 7, in particular a method, based on the use of a validation set, to detect stasis where individuals no longer improve or overfit. Wave has advantages in terms of speed because only part of the global solution is evaluated during each period. Moreover, on the problems studied it proves to be a competitive method with respect to other methods of GP such as GSGP or non-EC based machine learning methods such as MLR.

HetCA

HetCA is an artificial life simulation that addresses objective 5 using a heterogeneous CA with local rules to simulate the open ended evolution. In HetCA the environment of a cell is exclusively constituted by the states of neighboring cells and consequently maintain long-term evolutionary changes, a form of open-ended evolution, also maintains a perpetually changing environment from the point of view of the cells.

Objective 7 is tackled in Chapter 3 and the simulation succeeded in producing long-term evolutionary dynamics as well as evolutionary progress, according to three criteria: the density of the phenotype, the size of the genotype and the robustness of the individuals. The definition of evolutionary progress used is that

proposed by [Shanahan, 2012]. Moreover strategies[1] used by genotypes in those simulations may indicate the emergence of altruistic behavior in this simulation.

Objective 9, external environmental fluctuations, were introduced and studied in Chapter 6. These fluctuations are obtained by giving a disadvantage to the cells being at certain states to propagate their genotypes to the neighboring cells, and by cyclically varying these disadvantaged states. We observe, through the study of phenotypic and genotypic diversity, that depending on the types of cycles used, the selection seems to take place at the genotypic level (the transition rules) or at the phenotypic level (the cell states). These results are in line with ideas on the levels of selection developed by [Jablonka et al., 2005c].

8.2 Future Recommandations

More generally, these experiments showed the impact of in-depth studies of environmental fluctuations in the context of EC simulations. We describe some extensions to the works in this thesis that seem to us have particular potential.

Spatiality in HetCA

As shown, for example, by circadian cycles, environmental fluctuations have had an impact on the way life has evolved. But these fluctuations are not only temporal, they are also spatial. For example, through spatial and progressive environmental changes, Stephen Jay Gould explains the apparent gradualism of evolution. We have not been interested here in spatial variations, but it would be quite possible and certainly interesting to add this dimension to HetCA. For example, rather than switching from one environmental condition to another in a discrete way it would be possible to gradually make the transition from an initial location to the whole CA. One can also consider defining several fixed zones on the grid of the cellular automaton where to alternate the different possible environmental conditions.

[1] Uses of decay to build wall between cells using different genotypes as well as the release of space by the genetically induced change of state towards quiescent.

Amplitude and frequency of fluctuations in HetCA

From the experiment proposed in Chapter 6, it is difficult to determine the precise conditions leading to phenotypic selection or genotypic selection. Additional experiments would make it possible to decide, for example, by testing other configurations of rapid and slow fluctuations. On the other hand, if the frequency of the fluctuations is the determining factor, it would be interesting to determine the threshold from which the type of selection is reversed. Finally, it may be possible to consider whether this possible threshold is dependent on parameters such as, for example, the number of successive living iterations before decay of a cell.

New types of fluctuations in HetCA

In the work presented here we have chosen to vary the capacity of the cells to transmit their genotype according to their state. However, HetCA uses different parameters and it would have been possible to use them instead. In particular, it would have been possible to vary the age of the cells or their decay time. It would be very interesting to see whether the phenotypic and genotypic differences are affected by such fluctuations to see whether they are affected by the environmental fluctuations in general or the types of fluctuation that we have used here.

Wave: a hybrid methodology?

Wave allows us to combine different forms of GP. Nevertheless, for the time being, the only method used beyond classical GP is linear scaling GP. It would be interesting to exploit this feature by combining other GP methods. Moreover, Wave could easily reuse non-EC based machine learning methods in some periods if the method in question is able to solve problems of symbolic regression. For example it would be quite possible to combine Wave with MLR or deep neural networks.

Multiobjective Interleaved GP

With Interleaved-Size we showed that it was potentially possible to use the Interleaved GP to accomplish several objectives simultaneously. However, this possibility has not been exploited to the maximum and it might be possible to exploit it within the framework of other forms of multi-objective GP. For example, in the context of symbolic regression problems such as those used here, one can think of separating the set of data into several subsets of data used alternately, one can also think of testing it on classic multi-objective problems such as, for example, cones problem.

Solve other problems with Wave

This thesis restricted the application Wave to symbolic regression problems; a clear next step would be to use Wave to solve other types of problems, such as classification. It should be noted that, as explained in [Loveard and Ciesielski, 2001], there are different approaches to classification in GP and that if it is theoretically possible to use Wave to solve problems of this type, several different methods can be envisaged.

Evolution in four dimensions

In such works as [Jablonka et al., 2005c], a model of evolution in four dimensions that we mentioned in Chapter 2 is proposed, here we have sought inspiration in these new models of evolutionary biology, but if we consider a phenotypic selection and a genoypic selection in HetCA we do not propose simulations where four levels of selections are present simultaneously. To propose such models in EC would be both a means of exploring their usefulness in machine learning (for example to simultaneously evolve the behavior and the morphology of a robot more efficiently) but also to some extent to test the predictions made in evolutionary biology concerning these models.

9

Glossary

Altruism

In evolutionary biology, an organism is considered *altruistic* when its behaviour benefits other organisms, at a cost to itself. Costs and benefits are measured in terms of reproductive fitness [Okasha, 2008].

Evolutionary Biology

Evolutionary biology is a subfield of biology that focuses on the evolutionary mechanisms that led to produce the diversity of life that can be seen on earth.

Epigenetics

Epigenetics mechanisms are a set of molecular mechanisms that modulate gene expression, such as for example the methylation of DNA.

Exaptation

Exaptation [Gould and Vrba, 1982] is a shift in the function of a trait during evolution.

Fixism

Fixism is the assumption that all species of living things have always existed, which was a commonly held belief in the past. It is generally opposed to *Trans-*

mutation of species theory.

Functional Biology

Functional Biology is used here as the study of proximal causes of living organisms functioning mechanisms.

Group Selection

Group Selection is the assumption that natural selection might operate not only at the level of individuals but also between groups of individuals of the same species.

Kin Selection

To explain altruism, Kin Selection develops the idea that increasing the chances of reproduction of related individuals, even at the cost of reducing its own chances of reproduction, is a valid evolutionary strategy.

Memes

Memes are cultural replicators, copied by imitation, as proposed by Richard Dawkins in [Dawkins et al., 1989a].

Phyletic Gradualism

This theory theorizes that evolution and most speciations are slow uniform and gradual.

Punctuated Equilibrium

Punctuated Equilibrium was proposed in [Gould and Eldredge, 1977]. They argue that the study of fossils shows that evolution is not linear but a succession of stasis and rapid rare significant evolutionary changes

Saltationism

This theory describes how important mutations can move an individual far away from its parents on the fitness landscape. This theory is not without controversy [Mayr, 1942] since in modern evolutionary theory, saltationism it is often considered to be fundamentally incompatible with phyletic gradualism and therefore is controversial. Nevertheless more recent works [Pigliucci, 2007] shows that it plays a role in evolution by natural selection [Blackburn, 1995].

Spontaneous Generation

Spontaneous Generation is an obsolete theory that develops the idea that simple living organisms appear spontaneously from inanimate matter

Transmission of acquired characteristics

The transmission of acquired characteristics is the ability for living beings to transmit to their offspring characteristics acquired during their lives. It was long accepted as an obvious mechanism, and then she left the scientic consensus with the discovery of Mendelian genetics, it is again considered in the light of the development of epigenetics.

Transmutation of species

Transmutation of species is the theory that the species of living beings have changed over time, that some of them extinct while others have appeared. This term was first used in Lamarck [1809] and is generally opposed to *fixism*.

10

Very long-term HetCA simulation

We have chosen to present here two examples of HetCA runs that are particularly long, one classic *Stable* simulation without environmental fluctuation and the other with the fluctuations previously described as *Light fluctuations*.

10.1 Stable Simulation

10.1.1 Major Diversification Phase

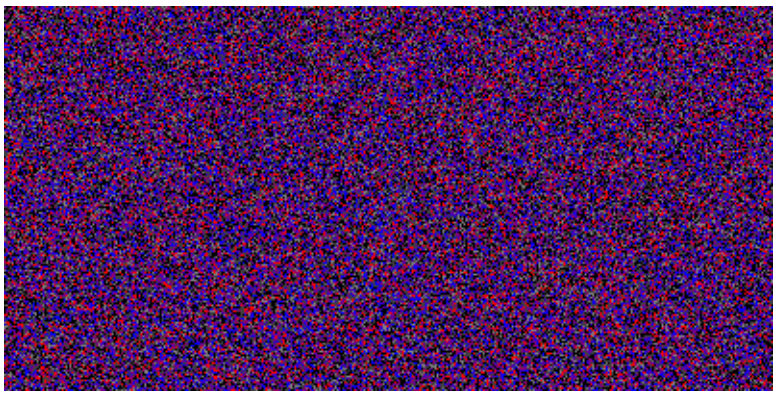


Figure 10.1: Iteration 2.

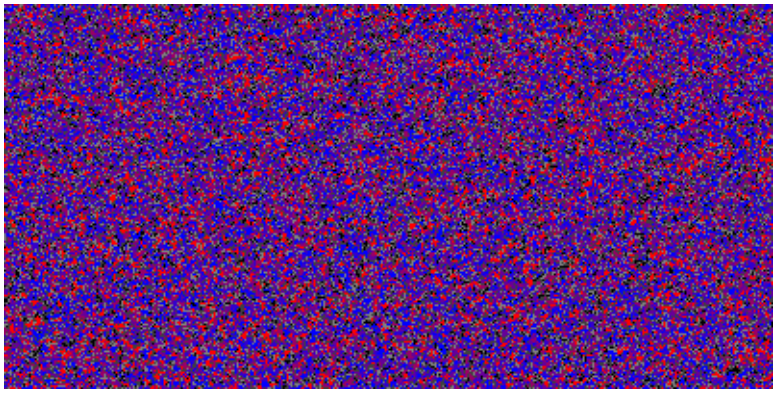


Figure 10.2: Iteration 5.

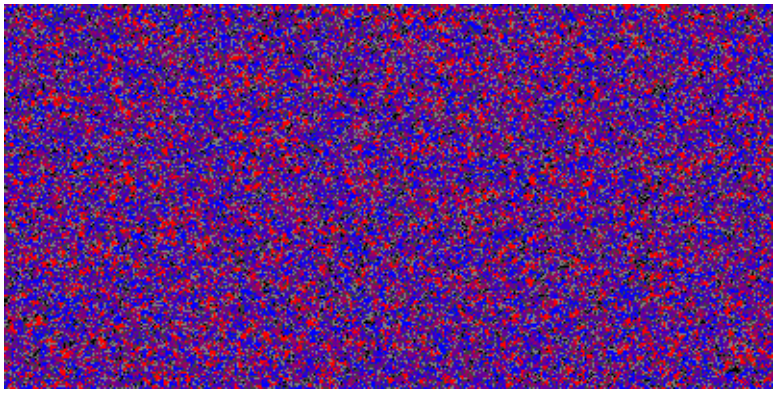


Figure 10.3: Iteration 3.

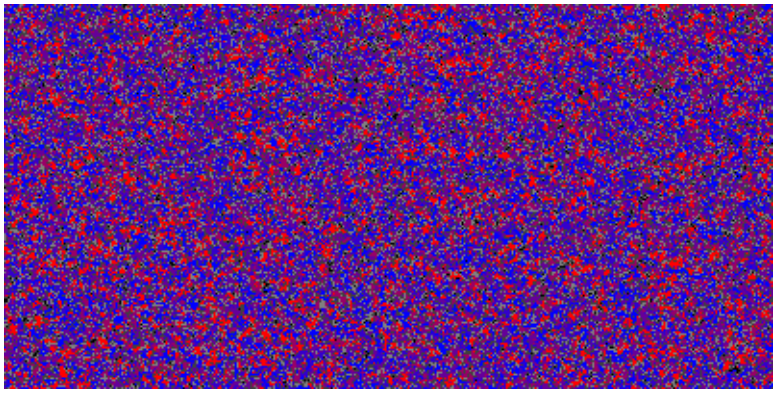


Figure 10.4: Iteration 5.

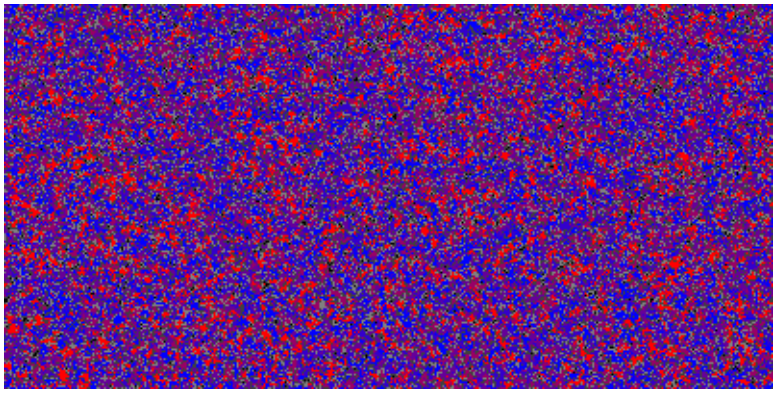


Figure 10.5: Iteration 7.

10.1.2 Massive Extinction Phase

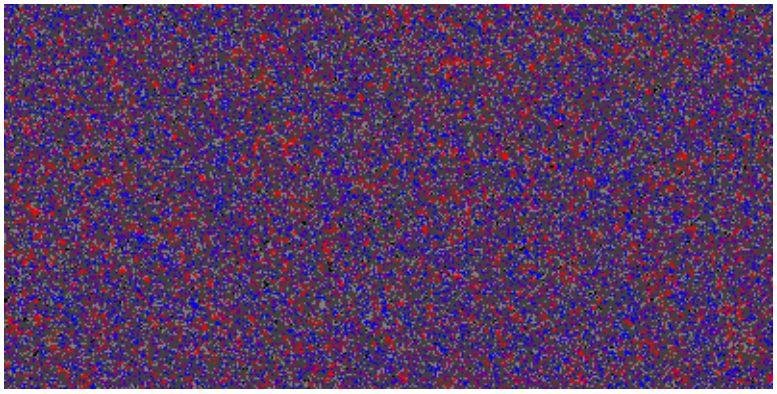


Figure 10.6: Iteration 8.

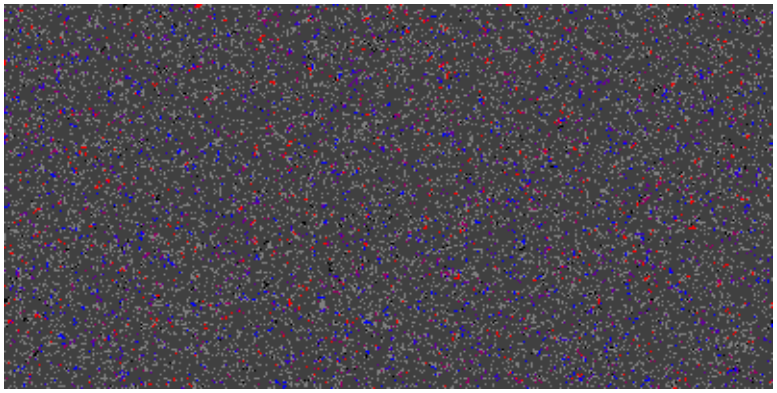


Figure 10.7: Iteration 10.

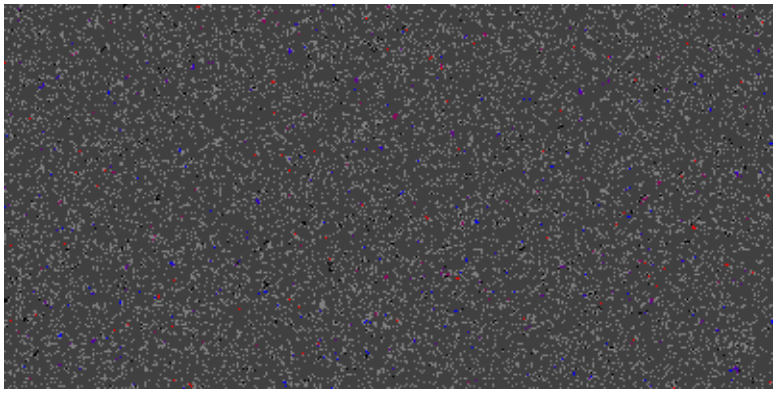


Figure 10.8: Iteration 13.

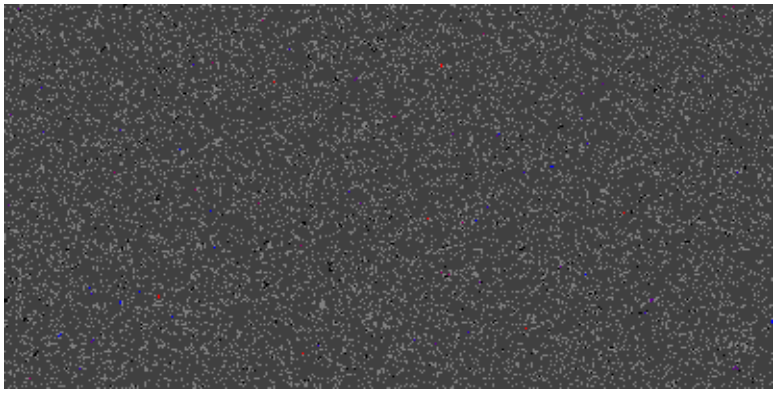


Figure 10.9: Iteration 16.

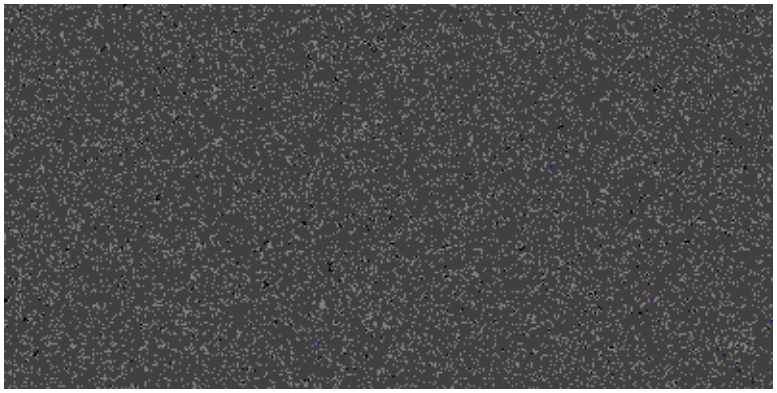


Figure 10.10: Iteration 19.

10.1.3 Colonization of free space by survivors

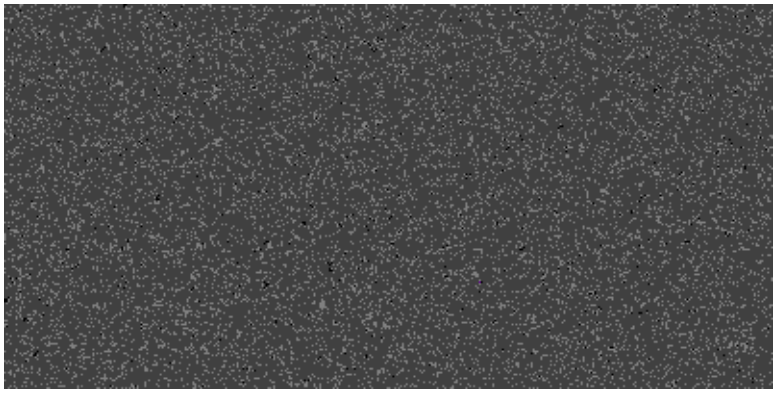


Figure 10.11: Iteration 100.

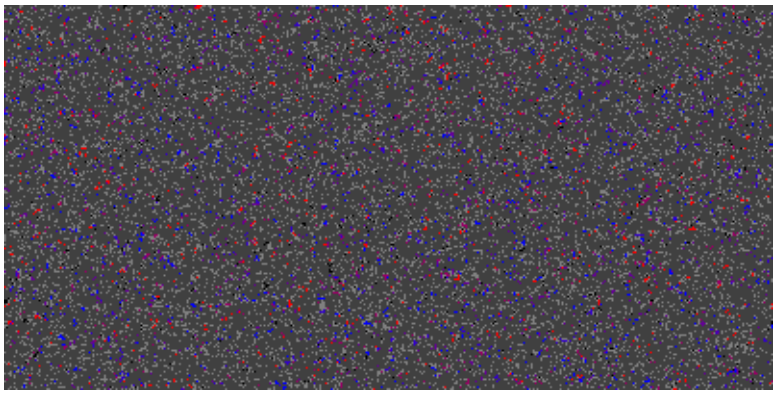


Figure 10.12: Iteration 2500.

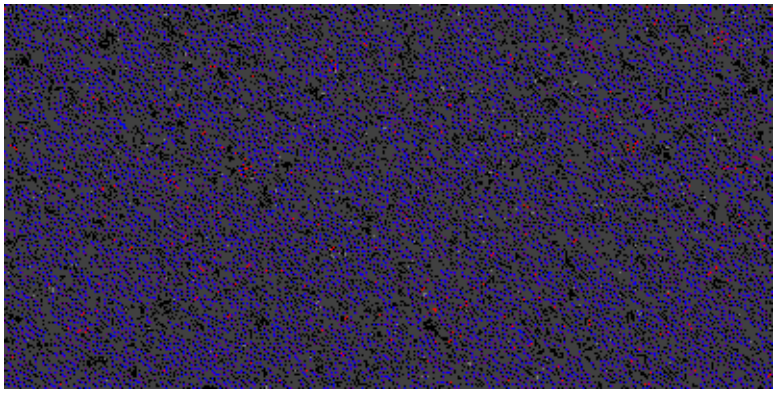


Figure 10.13: Iteration 5000.

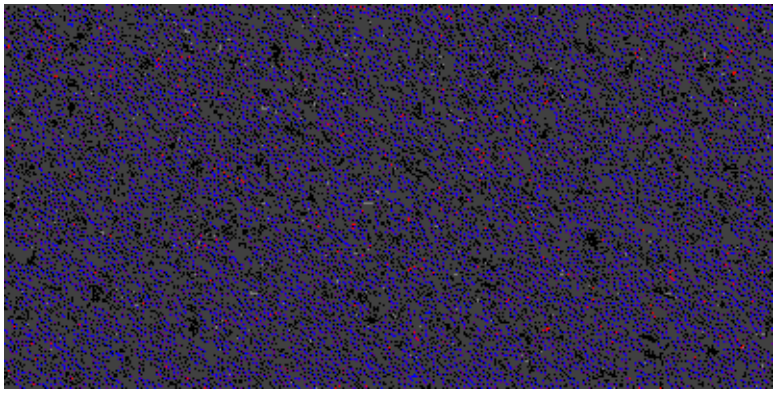


Figure 10.14: Iteration 7500.

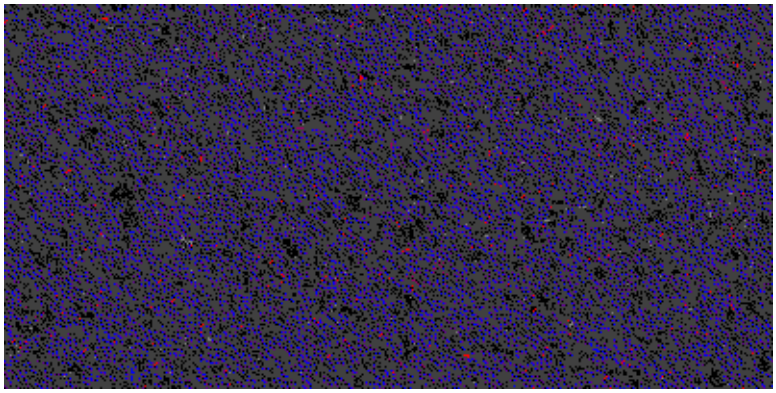


Figure 10.15: Iteration 10000.

10.1.4 Long-Term Evolution

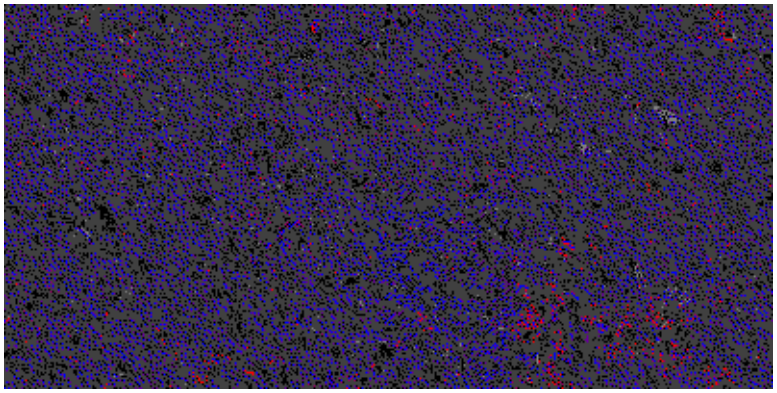


Figure 10.16: Iteration 50000.

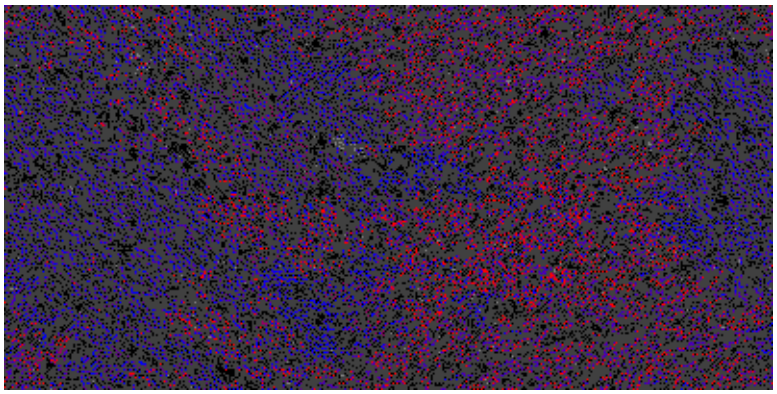


Figure 10.17: Iteration 100000.

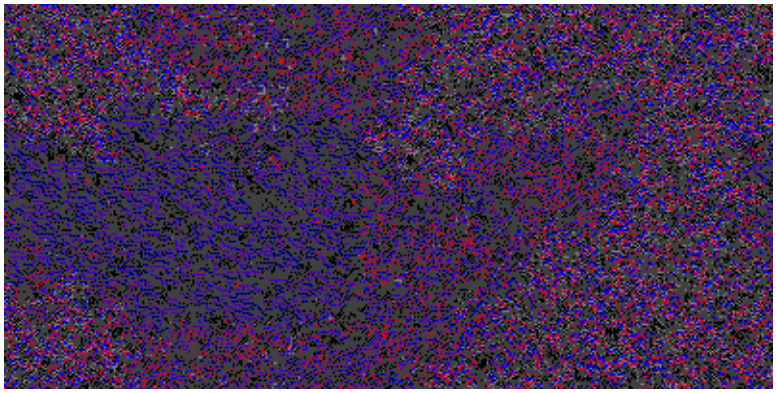


Figure 10.18: Iteration 200000.

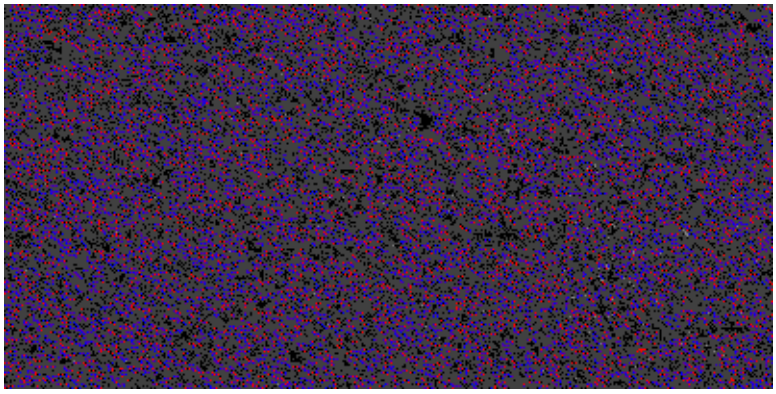


Figure 10.19: Iteration 300000.

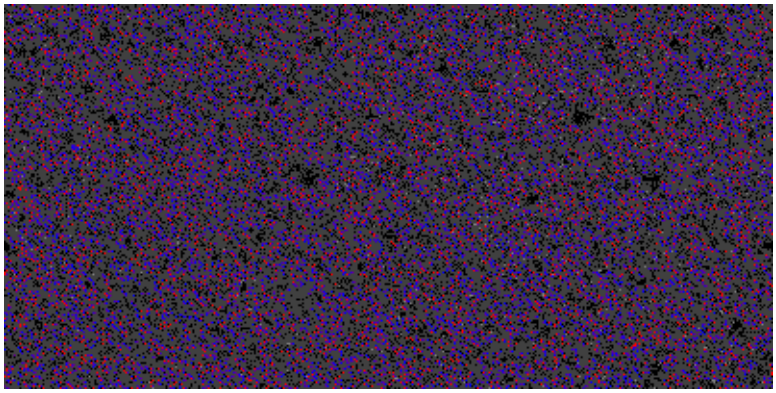


Figure 10.20: Iteration 400000.

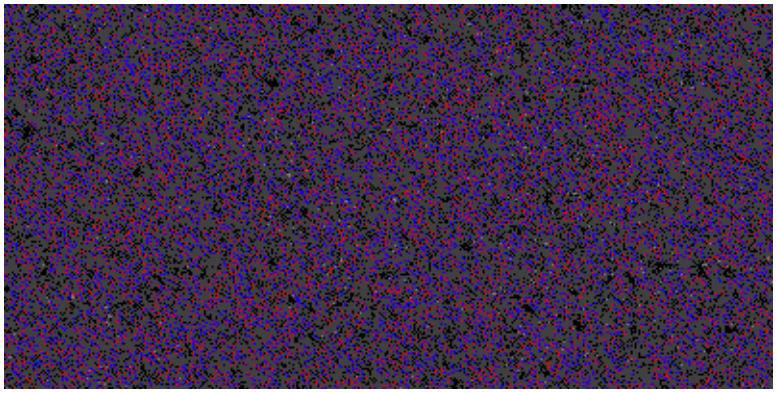


Figure 10.21: Iteration 500000.

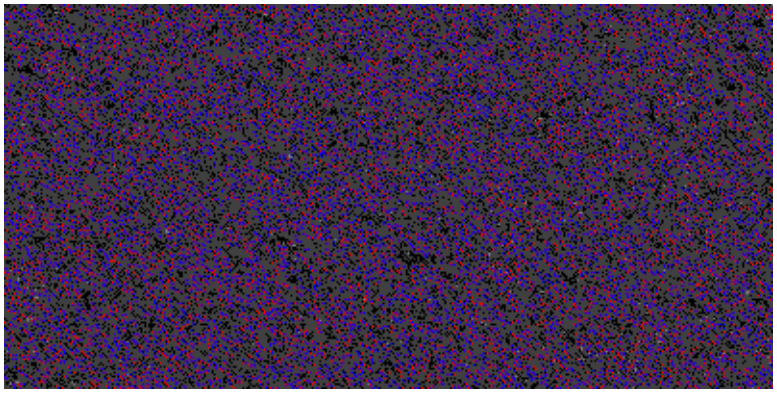


Figure 10.22: Iteration 600000.

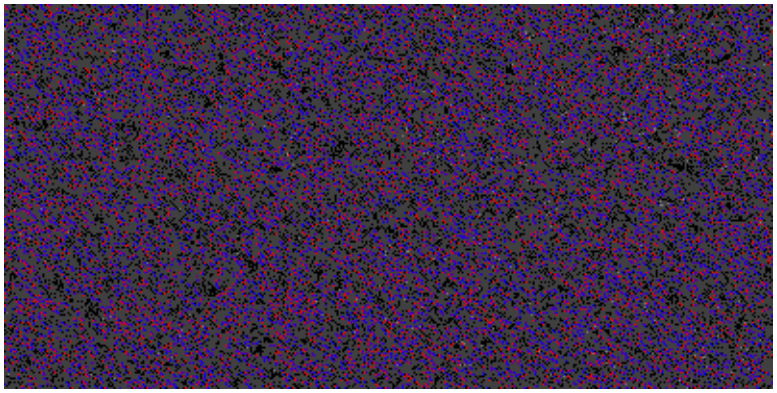


Figure 10.23: Iteration 700000.

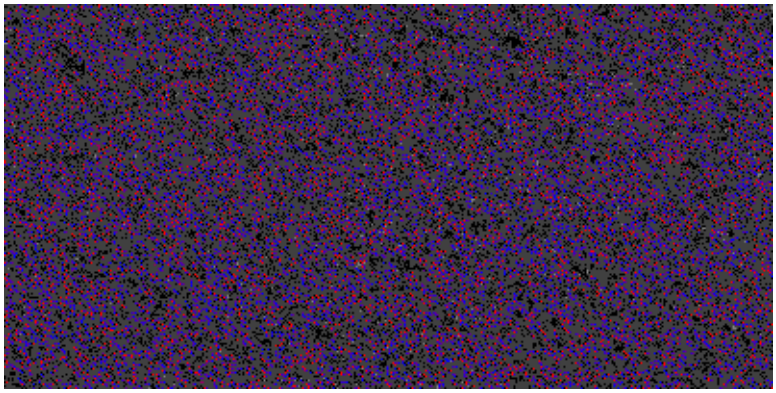


Figure 10.24: Iteration 800000.

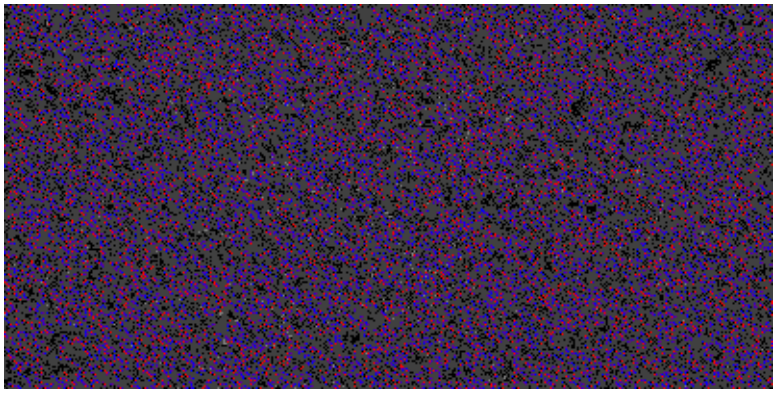


Figure 10.25: Iteration 900000.

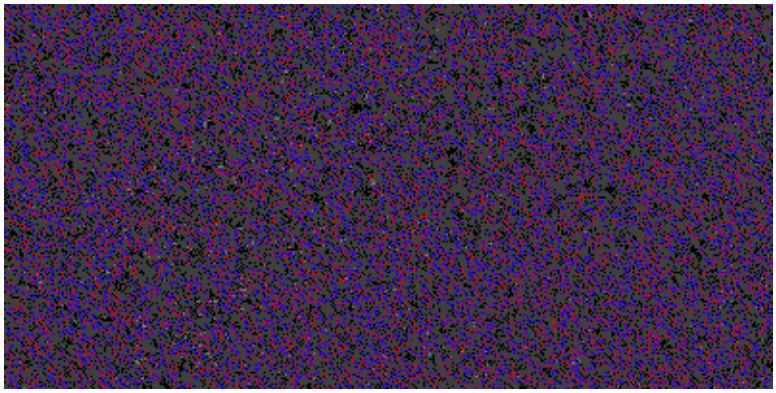


Figure 10.26: Iteration 1000000.

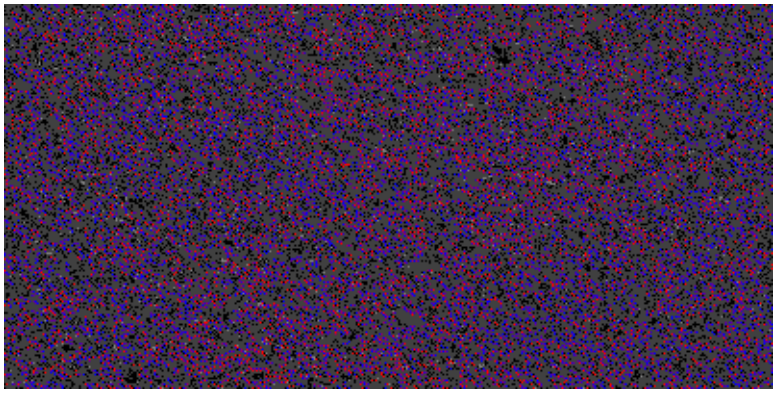


Figure 10.27: Iteration 1100000.

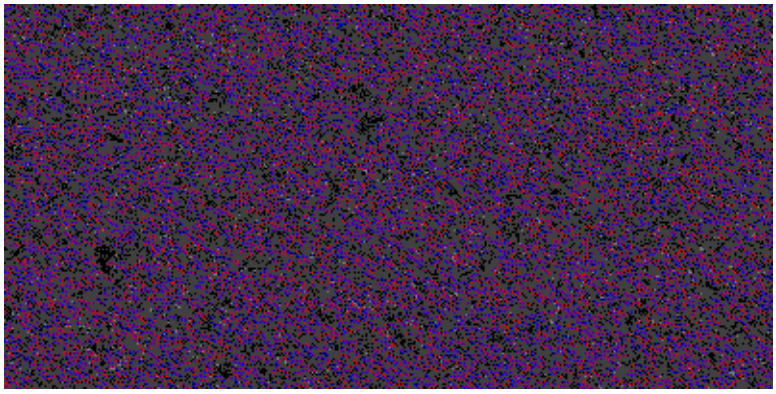


Figure 10.28: Iteration 1200000.

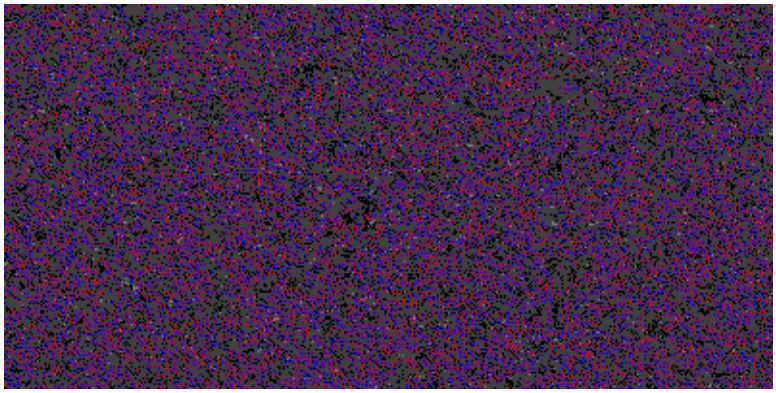


Figure 10.29: Iteration 1300000.

10.2 Light Fluctuation Simulation

10.2.1 Major Diversification Phase

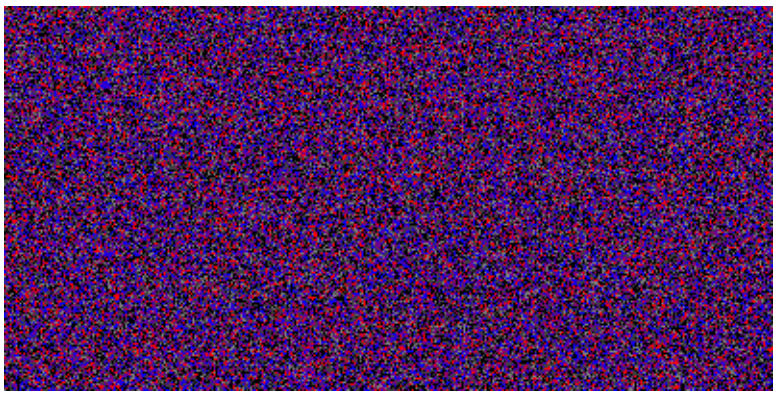


Figure 10.30: Iteration 2.

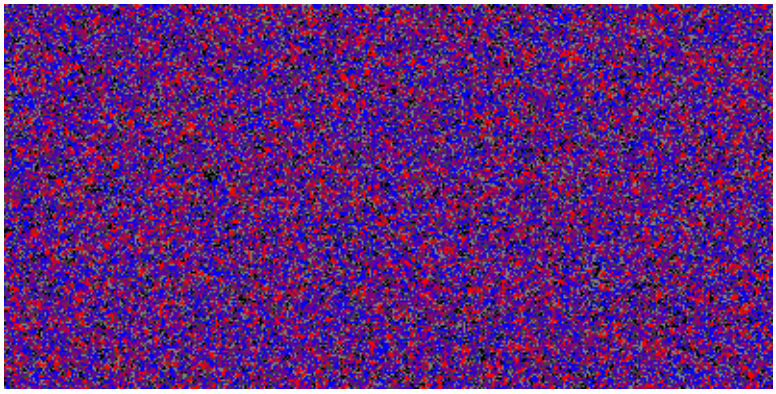


Figure 10.31: Iteration 5.

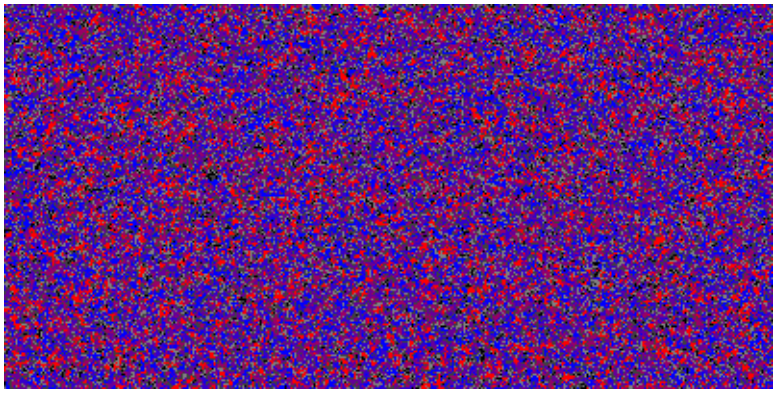


Figure 10.32: Iteration 3.

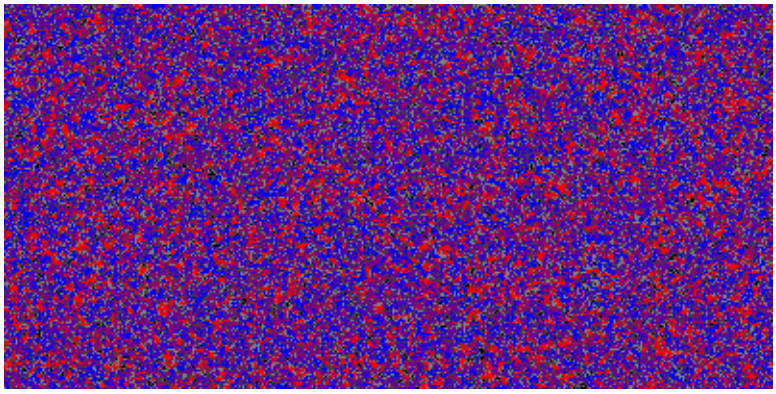


Figure 10.33: Iteration 5.

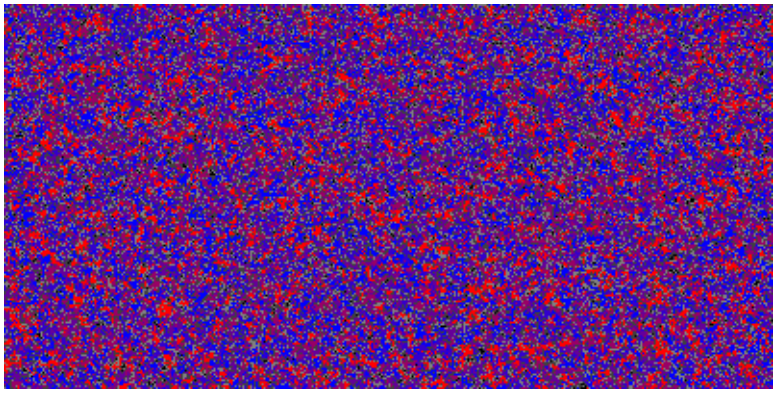


Figure 10.34: Iteration 7.

10.2.2 Massive Extinction Phase

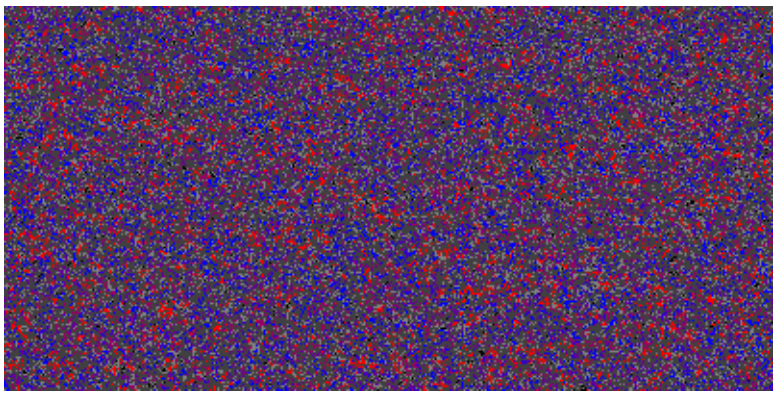


Figure 10.35: Iteration 8.

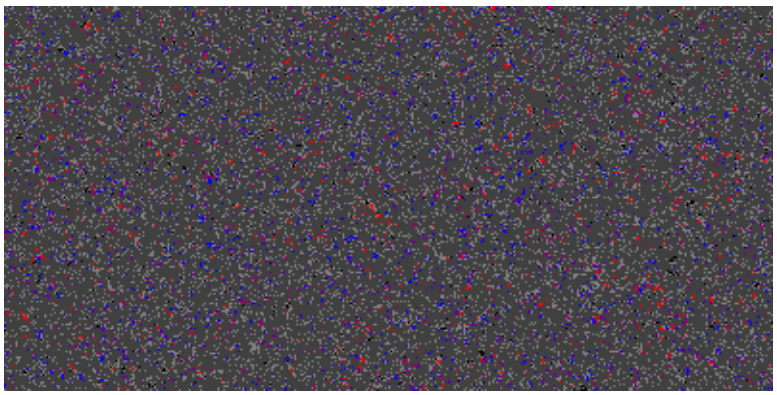


Figure 10.36: Iteration 10.

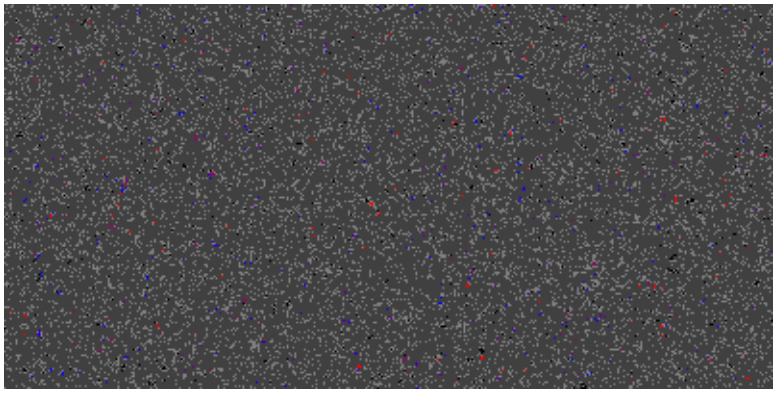


Figure 10.37: Iteration 13.

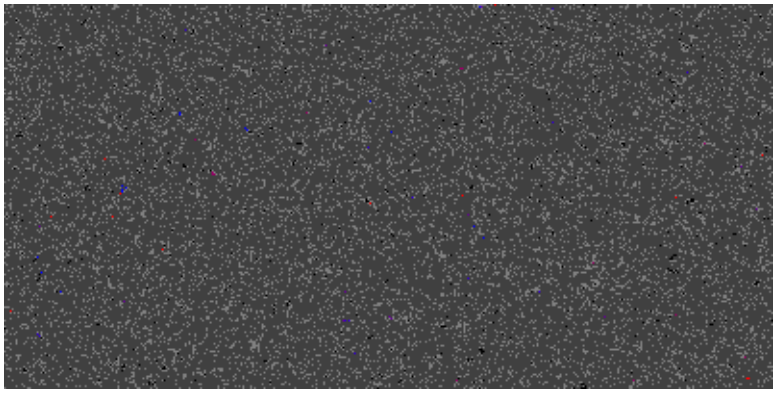


Figure 10.38: Iteration 16.

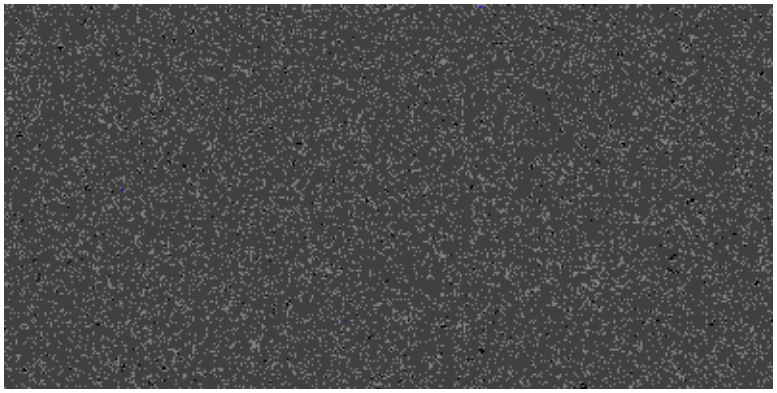


Figure 10.39: Iteration 19.

10.2.3 Colonization of free space by survivors

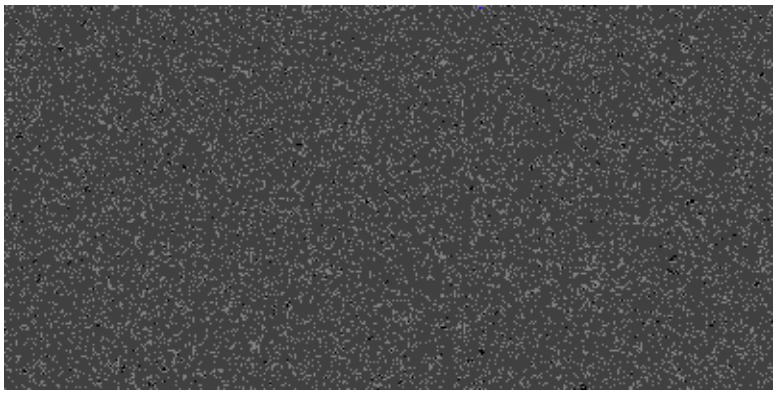


Figure 10.40: Iteration 100.

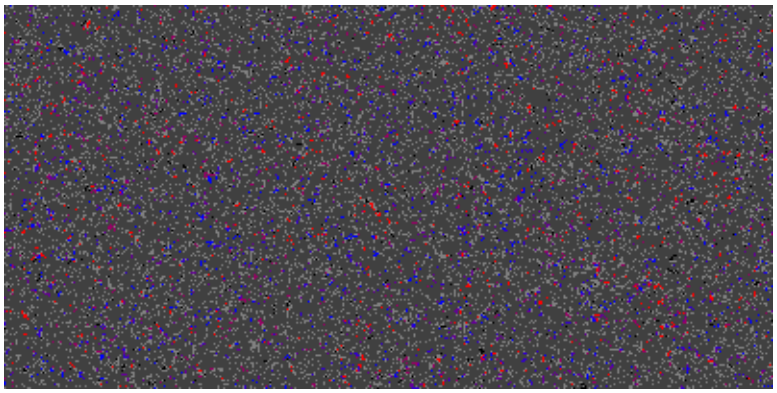


Figure 10.41: Iteration 2500.

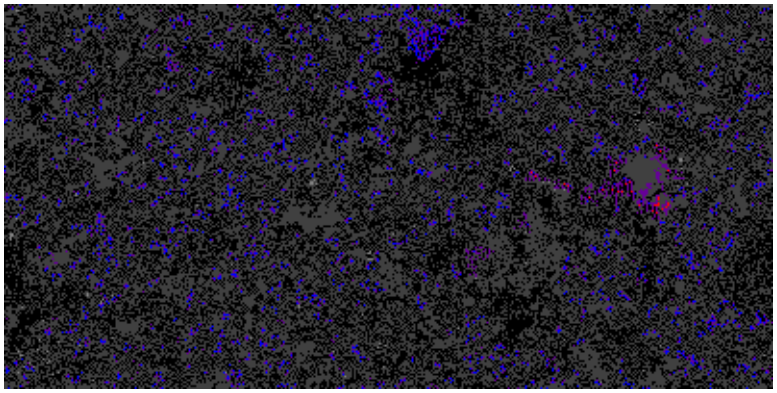


Figure 10.42: Iteration 5000.

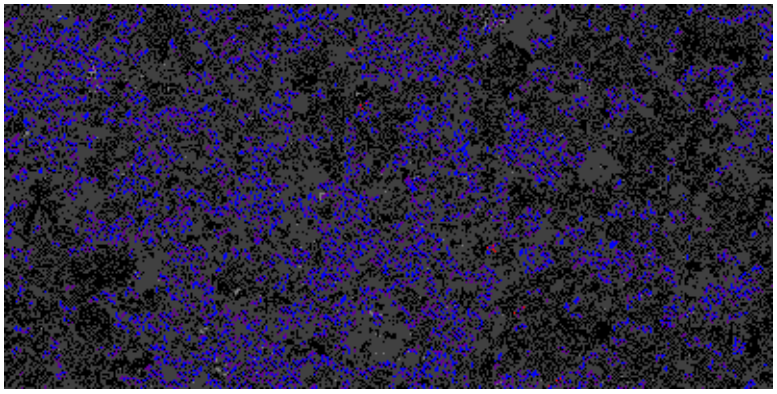


Figure 10.43: Iteration 7500.

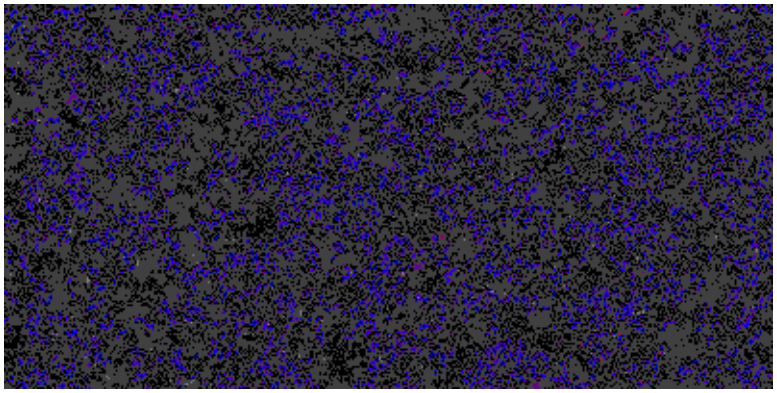


Figure 10.44: Iteration 10000.

10.2.4 Long-Term Evolution

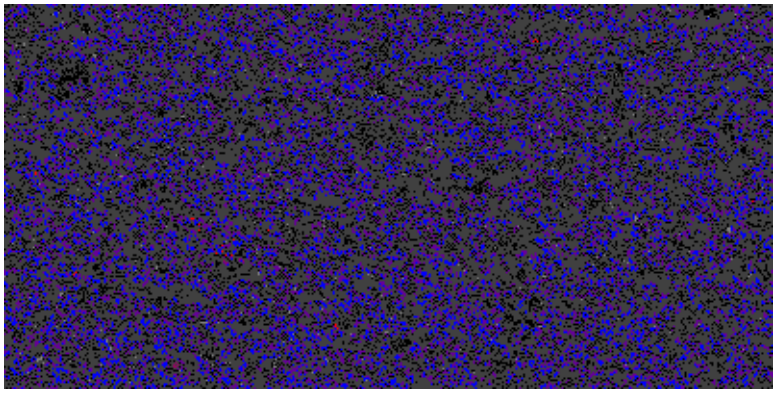


Figure 10.45: Iteration 50000.

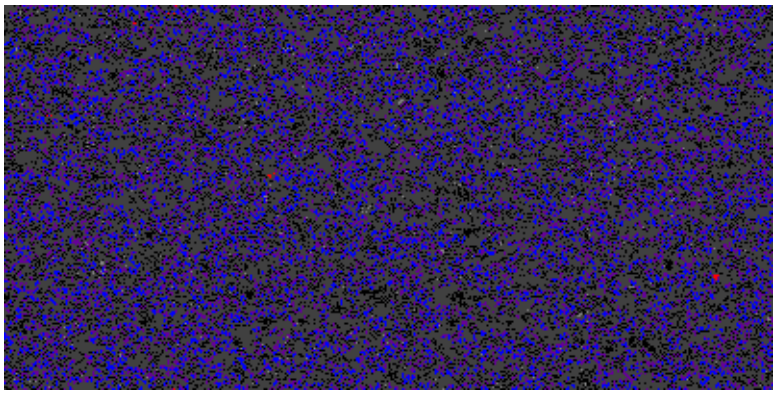


Figure 10.46: Iteration 100000.

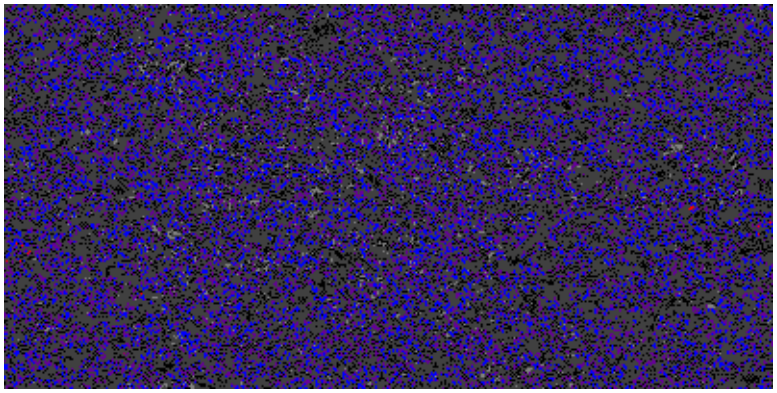


Figure 10.47: Iteration 200000.

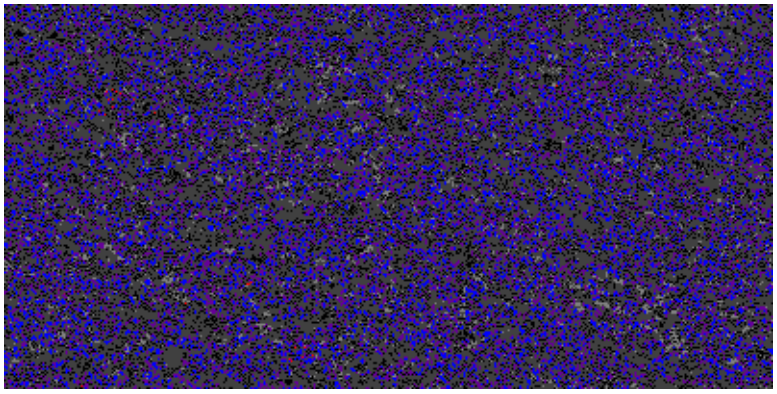


Figure 10.48: Iteration 300000.

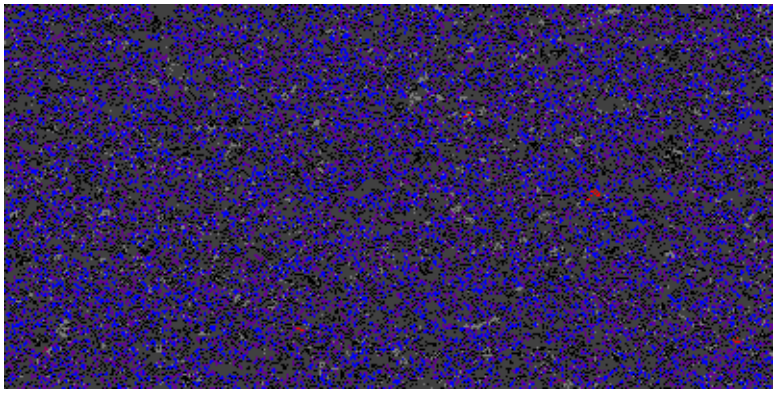


Figure 10.49: Iteration 400000.

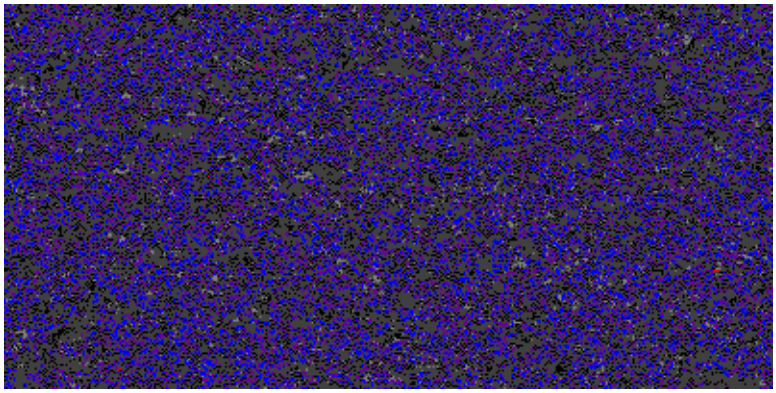


Figure 10.50: Iteration 500000.

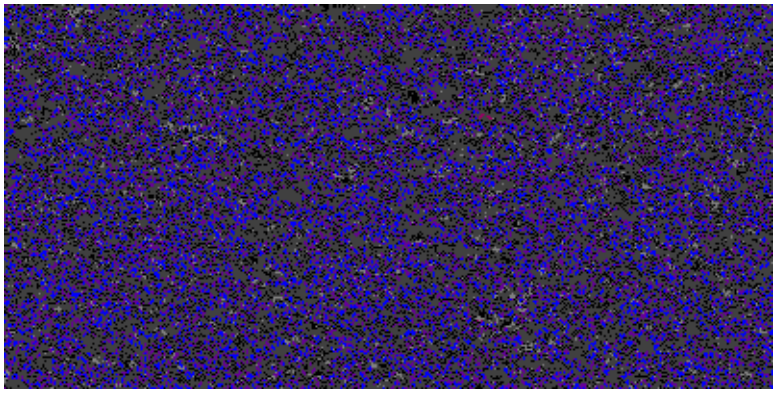


Figure 10.51: Iteration 600000.

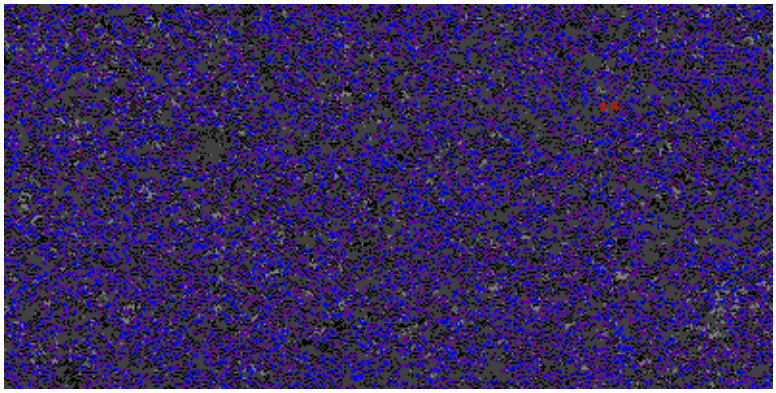


Figure 10.52: Iteration 700000.

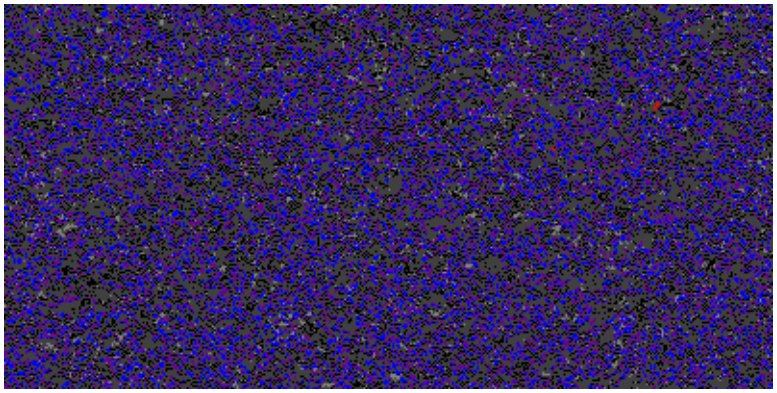


Figure 10.53: Iteration 800000.

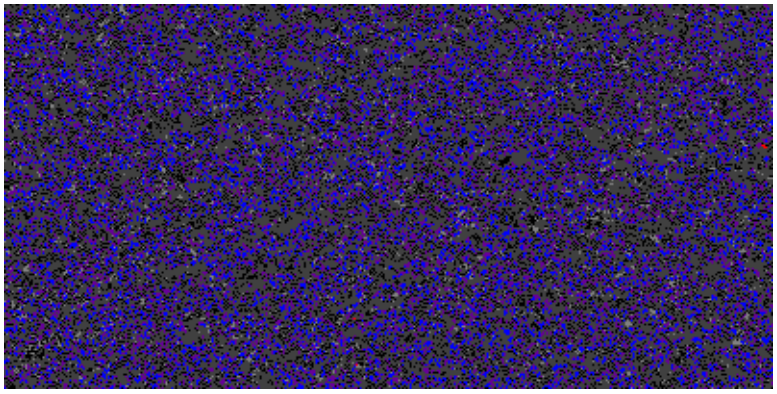


Figure 10.54: Iteration 900000.

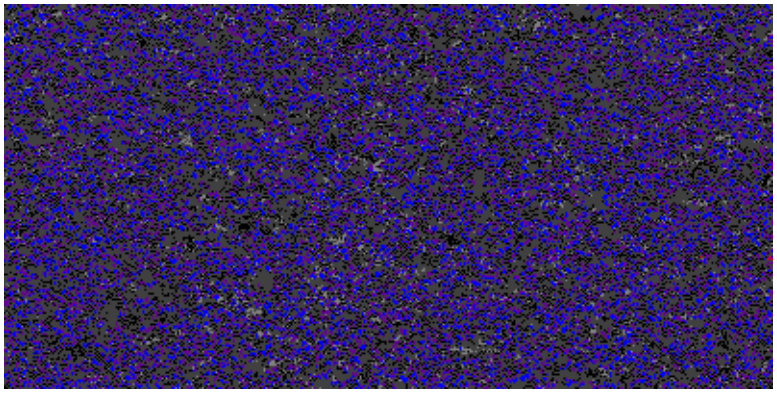


Figure 10.55: Iteration 1000000.

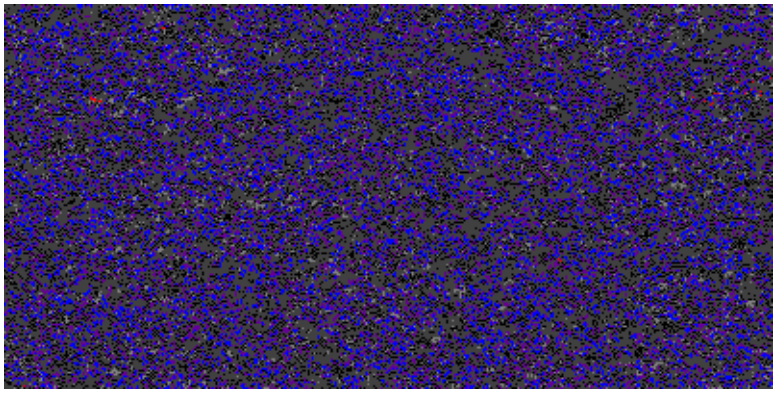


Figure 10.56: Iteration 1100000.

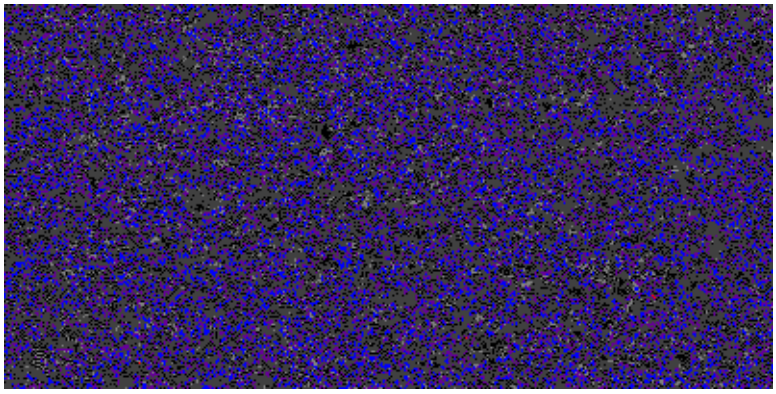


Figure 10.57: Iteration 1200000.

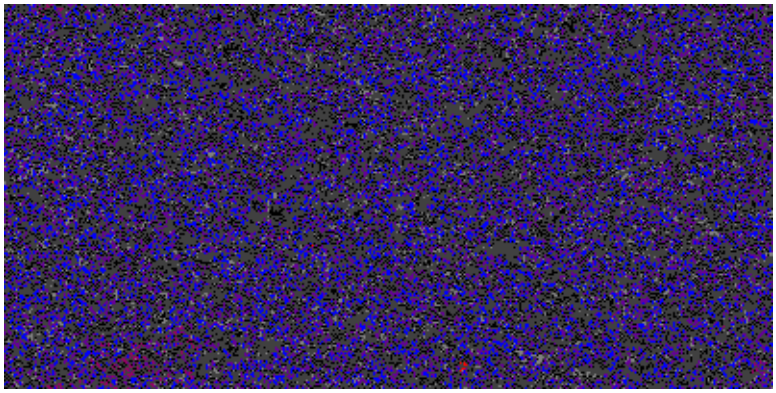


Figure 10.58: Iteration 1300000.

10.3 Small Fluctuation Simulation

10.3.1 Major Diversification Phase

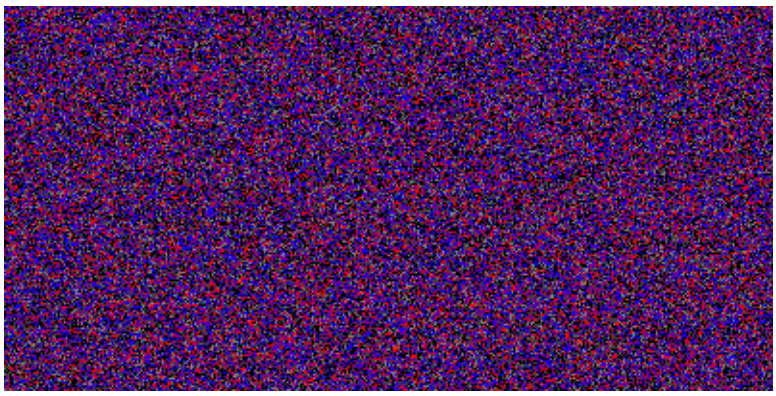


Figure 10.59: Iteration 2.

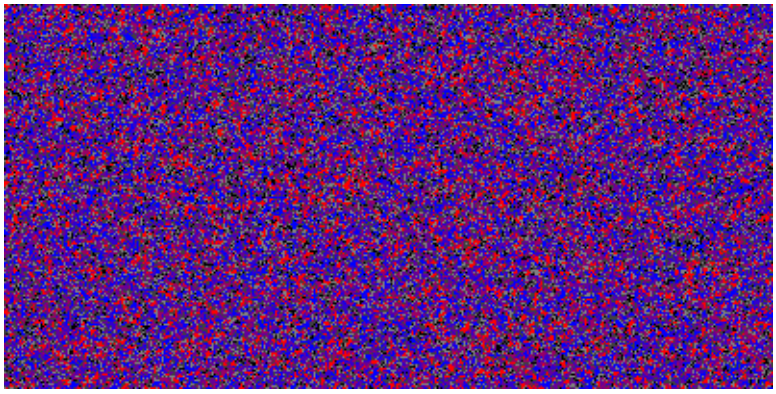


Figure 10.60: Iteration 5.

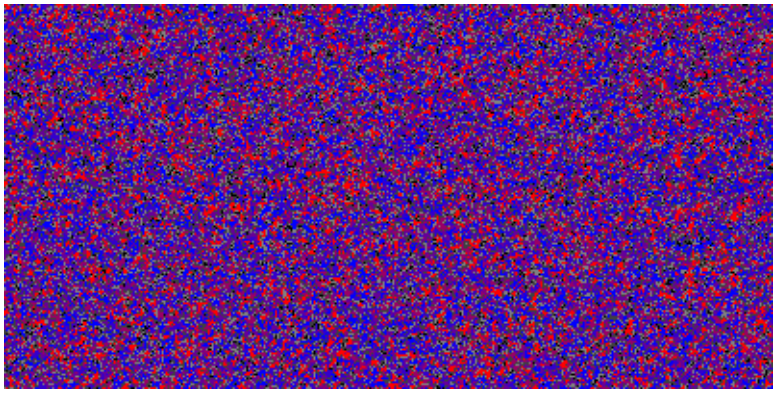


Figure 10.61: Iteration 3.

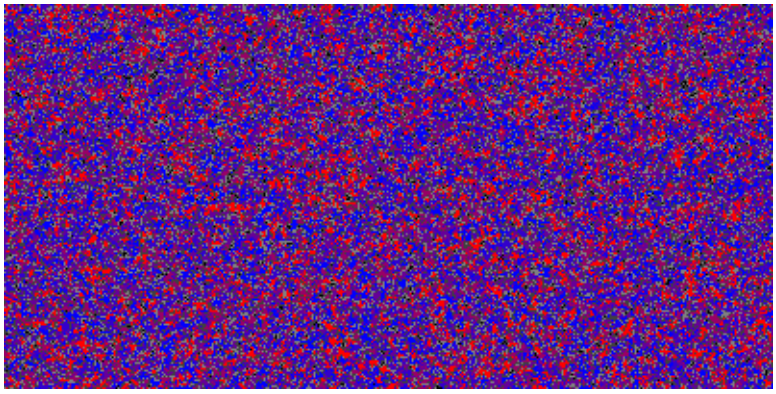


Figure 10.62: Iteration 5.

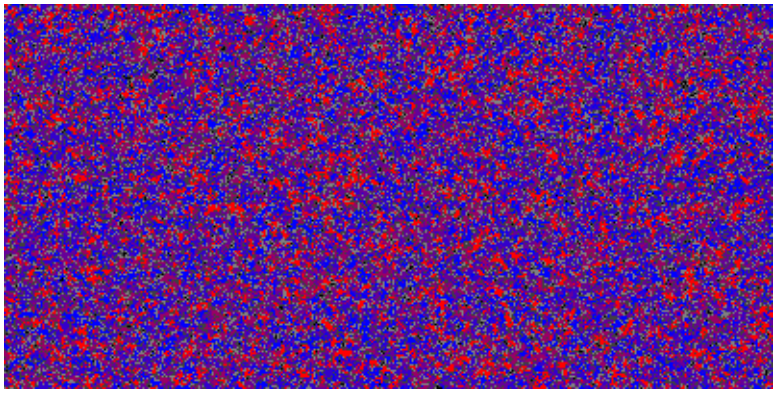


Figure 10.63: Iteration 7.

10.3.2 Massive Extinction Phase

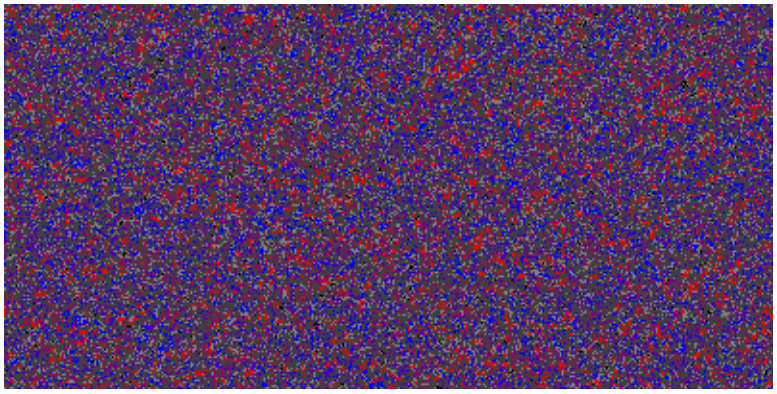


Figure 10.64: Iteration 8.

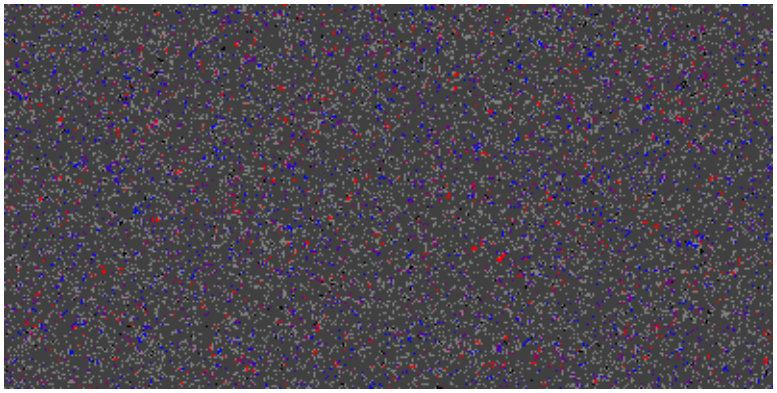


Figure 10.65: Iteration 10.

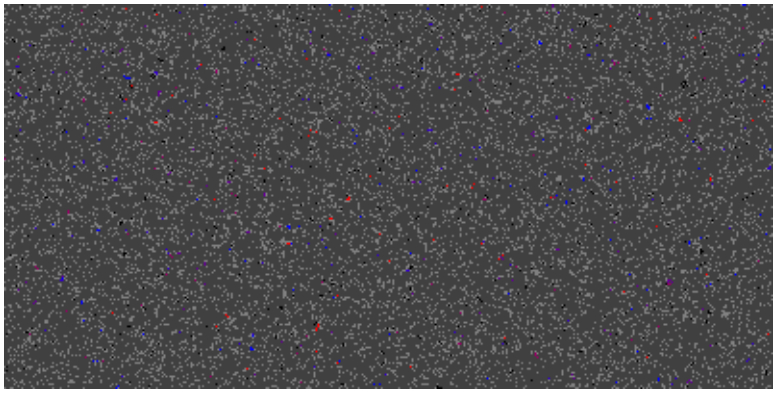


Figure 10.66: Iteration 13.

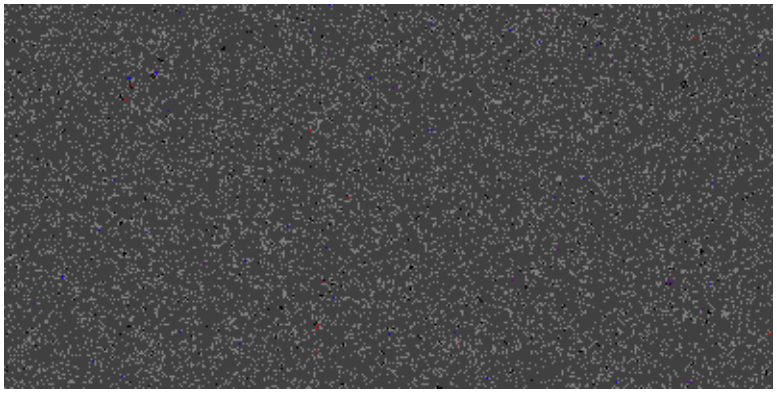


Figure 10.67: Iteration 16.

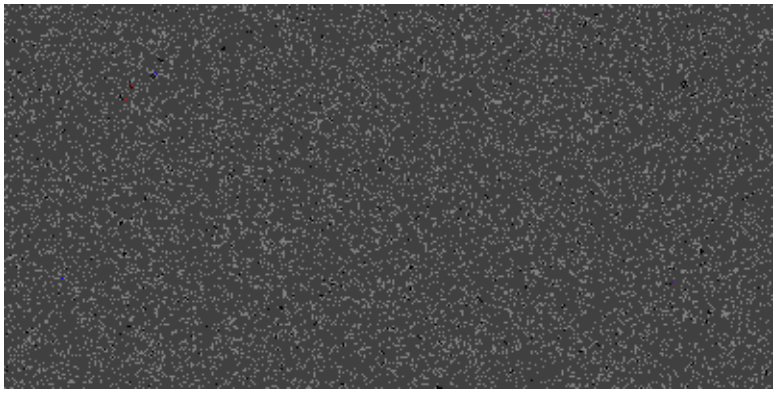


Figure 10.68: Iteration 19.

10.3.3 Colonization of free space by survivors

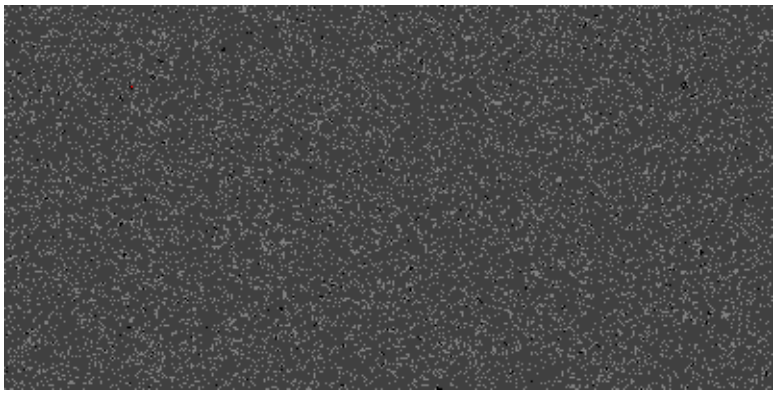


Figure 10.69: Iteration 100.

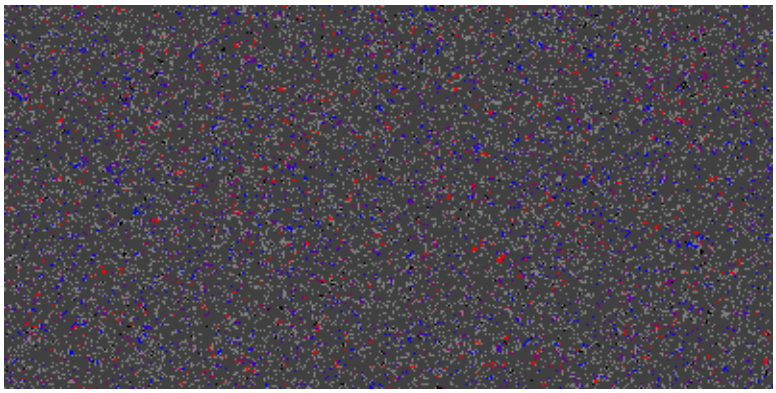


Figure 10.70: Iteration 2500.

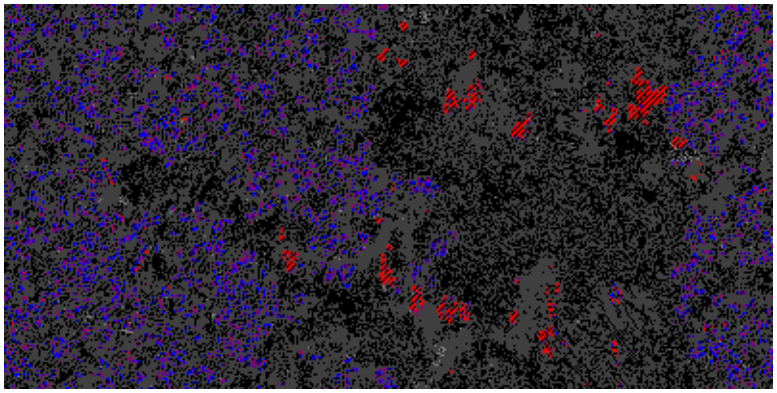


Figure 10.71: Iteration 5000.

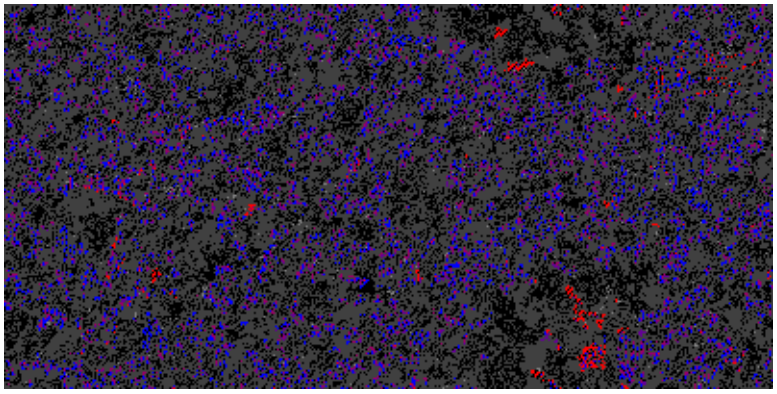


Figure 10.72: Iteration 7500.

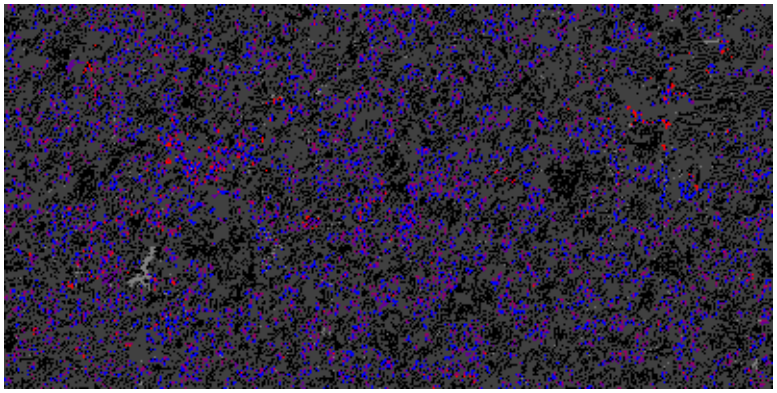


Figure 10.73: Iteration 10000.

10.3.4 Long-Term Evolution

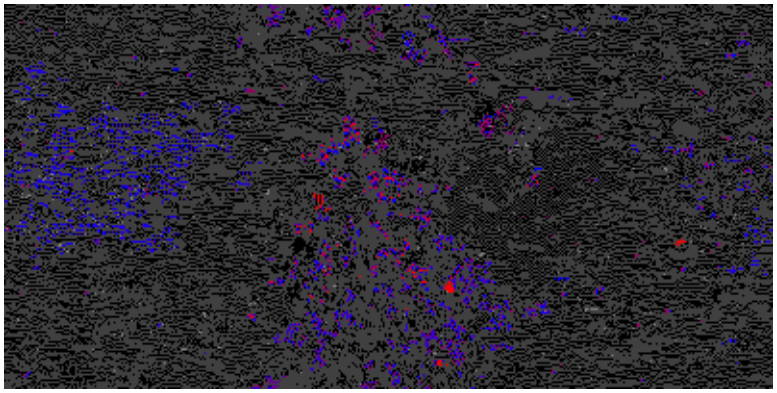


Figure 10.74: Iteration 50000.

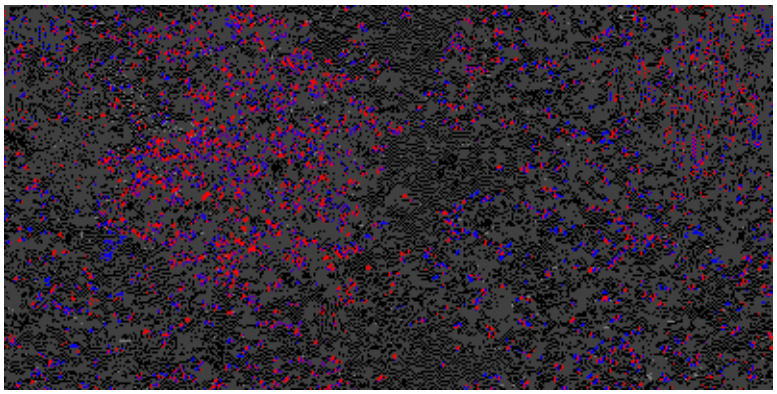


Figure 10.75: Iteration 100000.

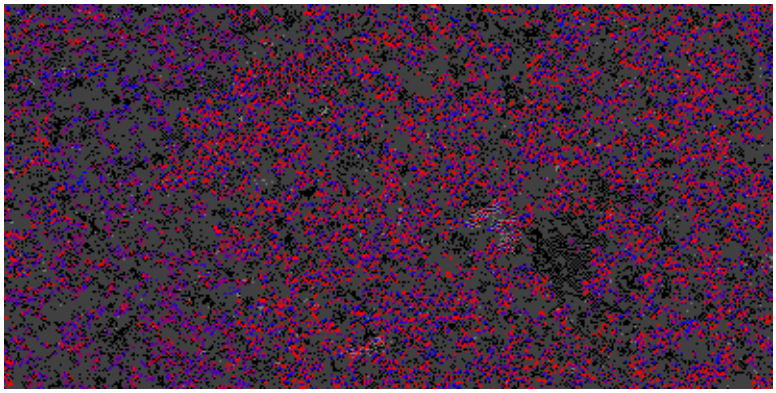


Figure 10.76: Iteration 200000.

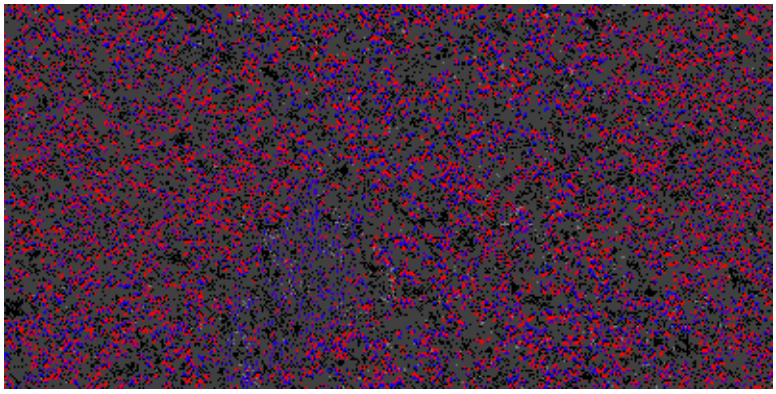


Figure 10.77: Iteration 300000.

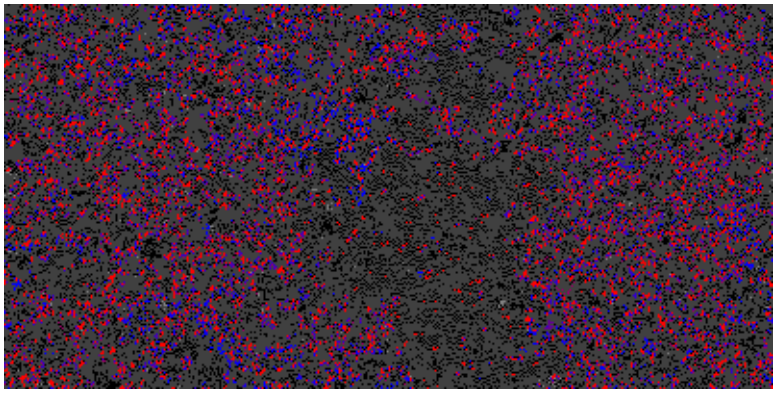


Figure 10.78: Iteration 400000.

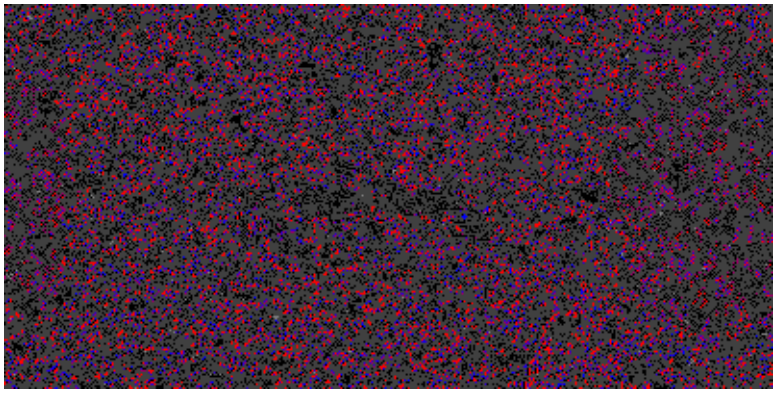


Figure 10.79: Iteration 500000.

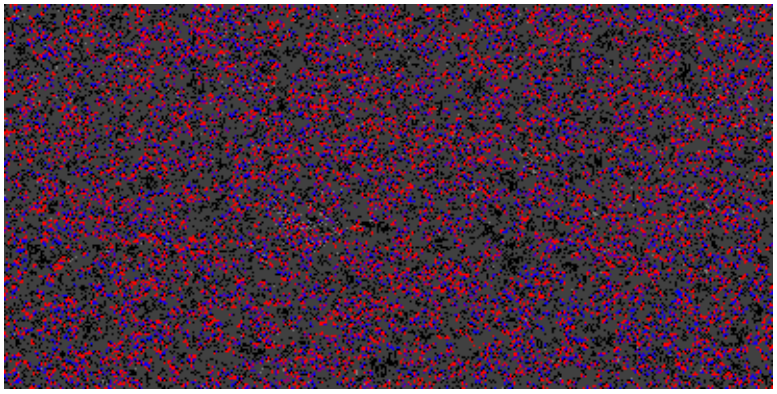


Figure 10.80: Iteration 600000.

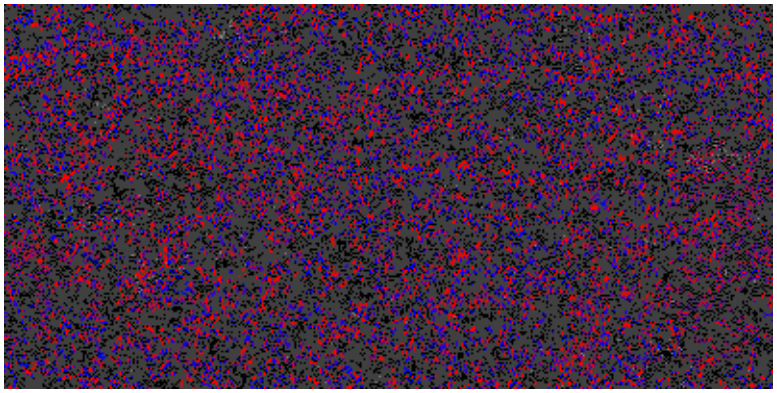


Figure 10.81: Iteration 700000.

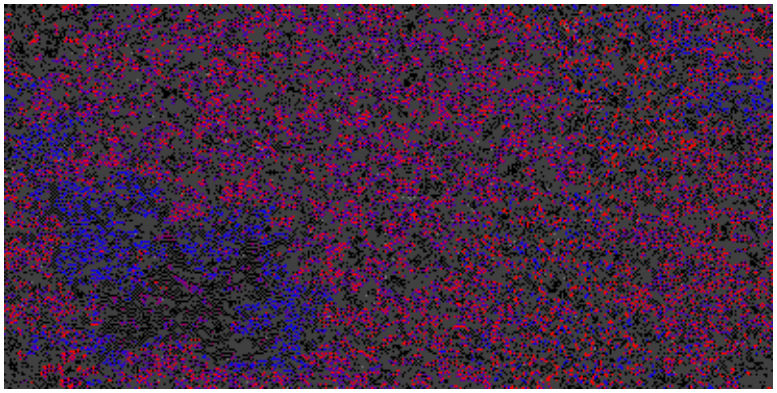


Figure 10.82: Iteration 800000.

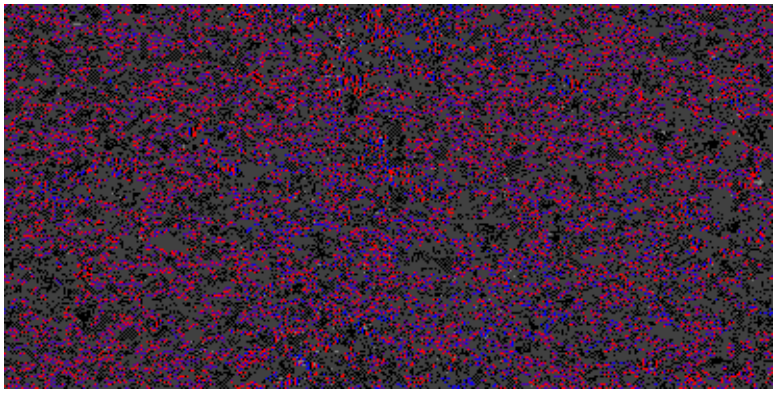


Figure 10.83: Iteration 900000.

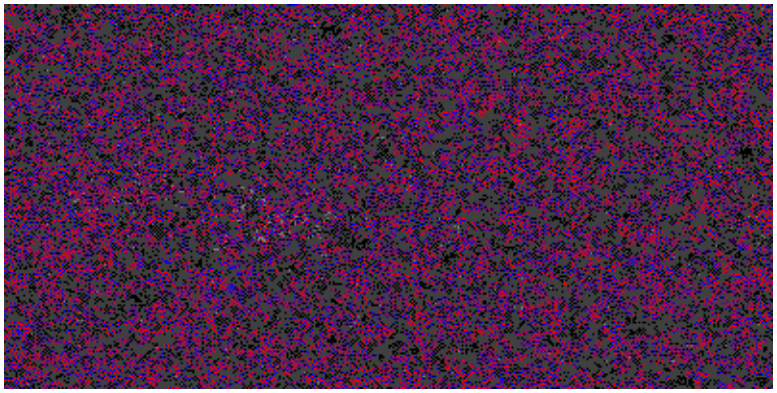


Figure 10.84: Iteration 1000000.

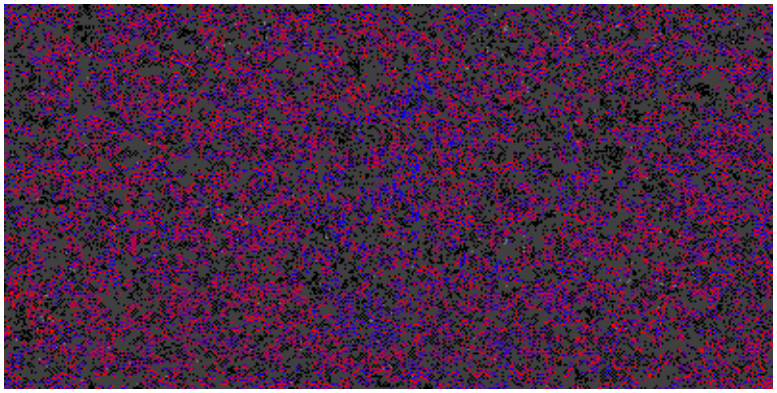


Figure 10.85: Iteration 1100000.

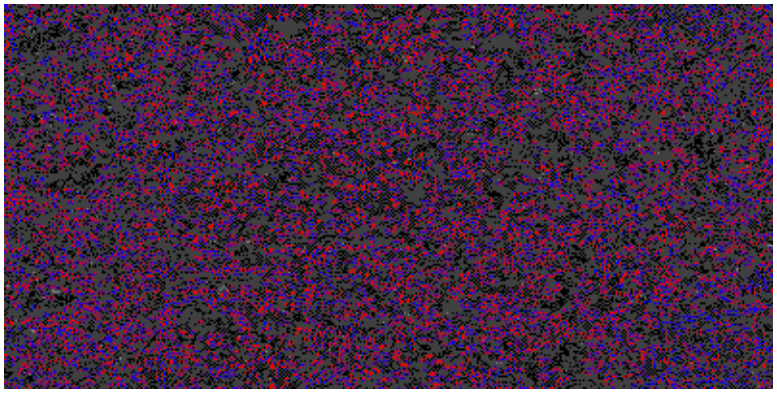


Figure 10.86: Iteration 1200000.

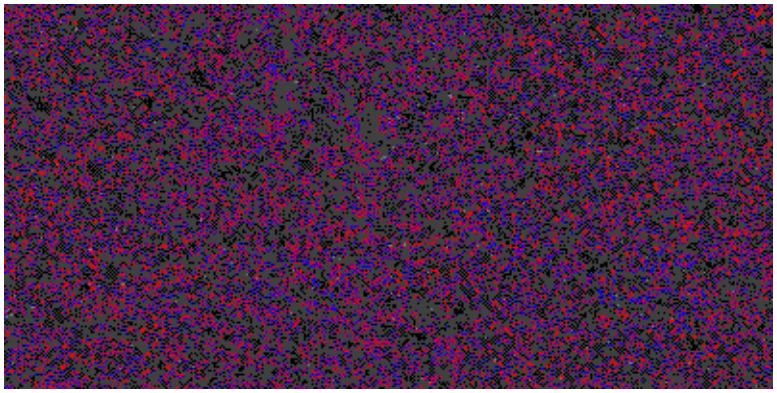


Figure 10.87: Iteration 1300000.

References

Abbot, P., Abe, J., Alcock, J., Alizon, S., Alpedrinha, J. A., Andersson, M., Andre, J.-B., Van Baalen, M., Balloux, F., Balshine, S., et al. (2011). Inclusive fitness theory and eusociality. *Nature*, 471(7339):E1–E4. 20

Adami, C. and Brown, C. T. (1994). Evolutionary learning in the 2d artificial life system avida. In *Artificial life IV*, volume 1194, pages 377–381. Cambridge, MA: MIT Press. 8, 50

Adami, C., Ofria, C., and Collier, T. C. (2000). Evolution of biological complexity. *Proceedings of the National Academy of Sciences*, 97(9):4463–4468. 50

Archetti, F., Lanzeni, S., Messina, E., and Vanneschi, L. (2006). Genetic programming for human oral bioavailability of drugs. In Keijzer, M., Cattolico, M., Arnold, D., Babovic, V., Blum, C., Bosman, P., Butz, M. V., Coello Coello, C., Dasgupta, D., Ficici, S. G., Foster, J., Hernandez-Aguirre, A., Hornby, G., Lipson, H., McMinn, P., Moore, J., Raidl, G., Rothlauf, F., Ryan, C., and Thierens, D., editors, *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, volume 1, pages 255–262, Seattle, Washington, USA. ACM Press. 100

Arnaldo, I., Krawiec, K., and O'Reilly, U.-M. (2014). Multiple regression genetic programming. In *Proceedings of the 2014 conference on Genetic and evolutionary computation*, pages 879–886. ACM. 123, 166

Azad, R., Medernach, D., and Ryan, C. (2014). Efficient interleaved sampling of training data in genetic programming. In *Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion*, pages 127–128. ACM. 122

Azad, R. and Ryan, C. (2010a). Abstract functions and lifetime learning in genetic programming for symbolic regression. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 893–900. ACM. 169

Azad, R. M. A. and Ryan, C. (2010b). Abstract functions and lifetime learning in genetic programming for symbolic regression. In Branke, J., Pelikan, M., Alba, E., Arnold, D. V., Bongard, J., Brabazon, A., Branke, J., Butz, M. V., Clune, J., Cohen, M., Deb, K., Engelbrecht, A. P., Krasnogor, N., Miller, J. F., O'Neill, M., Sastry, K., Thierens, D., van Hemert, J., Vanneschi, L., and Witt, C., editors, *GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 893–900, Portland, Oregon, USA. ACM. 41

Azad, R. M. A. and Ryan, C. (2011). Variance based selection to improve test set performance in genetic programming. In Krasnogor, N., Lanzi, P. L., Engelbrecht, A., Pelta, D., Gershenson, C., Squillero, G., Freitas, A., Ritchie, M., Preuss, M., Gagne, C., Ong, Y. S., Raidl, G., Gallager, M., Lozano, J., Coello-Coello, C., Silva, D. L., Hansen, N., Meyer-Nieberg, S., Smith, J., Eiben, G., Bernado-Mansilla, E., Browne, W., Spector, L., Yu, T., Clune, J., Hornby, G., Wong, M.-L., Collet, P., Gustafson, S., Watson, J.-P., Sipper, M., Poulding, S., Ochoa, G., Schoenauer, M., Witt, C., and Auger, A., editors, *GECCO '11: Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1315–1322, Dublin, Ireland. ACM. 41

Bache, K. and Lichman, M. (2013). Uci machine learning repository. 122, 166

Banzhaf, W. (1993). Genetic programming for pedestrians. In *ICGA*. 36

Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D. (1998). *Genetic programming: an introduction*. Morgan Kaufmann Publishers San Francisco. 36

Barricelli, N. A. et al. (1954). Esempi numerici di processi di evoluzione. *Methodos*, 6(21-22):45–68. 31, 49

Beadle, L. and Johnson, C. G. (2008). Semantically driven crossover in genetic programming. In *IEEE Congress on Evolutionary Computation*, pages 111–116. 38

Beckmann, B. E., McKinley, P. K., and Ofria, C. (2007). Evolution of an adaptive sleep response in digital organisms. In *European Conference on Artificial Life*, pages 233–242. Springer. 52

Bedau, M. A. (2000). Artificial life. In *Blackwell Guide to the Philosophy of Computing and Information*. Citeseer. 49

Bedau, M. A. (2003). Artificial life: organization, adaptation and complexity from the bottom up. *Trends in cognitive sciences*, 7(11):505–512. 51

Bedau, M. A., Snyder, E., Brown, C. T., Packard, N. H., et al. (1997). A comparison of evolutionary activity in artificial evolving systems and in the biosphere. In *Proceedings of the fourth European Conference on Artificial life*, pages 125–134. Citeseer. 50

Bidlo, M. and Vašíček, Z. (2008). Cellular automata-based development of combinational and polymorphic circuits: A comparative study. In Hornby, G., Sekanina, L., and Haddow, P., editors, *Evolvable Systems: From Biology to Hardware*, pages 106–117. Springer. 46

Bidlo, M. and Vašíček, Z. (2012). Evolution of cellular automata using instruction-based approach. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. 47

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer. 40

Blackburn, D. G. (1995). Saltationist and punctuated equilibrium models for the evolution of viviparity and placentation. *Journal of Theoretical Biology*, 174(2):199–216. 9, 114, 159, 187

Blackmore, S. (2000). *The meme machine*, volume 25. Oxford Paperbacks. 21

Blickle, T. and Thiele, L. (1994). Genetic programming and redundancy. In Hopf, J., editor, *Genetic Algorithms within the Framework of Evolutionary Computation (Workshop at KI-94, Saarbrücken)*, pages 33–38, Im Stadtwald, Building 44, D-66123 Saarbrücken, Germany. Max-Planck-Institut für Informatik (MPI-I-94-241). 41

Bonduriansky, R. (2012). Rethinking heredity, again. *Trends in ecology & evolution*, 27(6):330–336. 17

Borrello, M. E. (2005). The rise, fall and resurrection of group selection. *Endeavour*, 29(1):43–47. 23

Brameier, M. and Banzhaf, W. (2006). *Linear Genetic Programming*. Springer. 63

Brameier, M. F. and Banzhaf, W. (2007). *Linear genetic programming*. Springer Science & Business Media. 36

Bredeche, N. and Montanier, J.-M. (2012). Environment-driven Open-ended Evolution with a Population of Autonomous Robots. In *Evolving Physical Systems Workshop*, East Lansing, États-Unis. 52

Castelli, M., Silva, S., and Vanneschi, L. (2014). A c++ framework for geometric semantic genetic programming. *Genetic Programming and Evolvable Machines*, 16(1):73–81. 168, 169, 170

Castelli, M., Vanneschi, L., and Silva, S. (2013). Prediction of high performance concrete strength using genetic programming with geometric semantic genetic operators. *Expert Systems with Applications*, 40(17):6856–6862. 38

Ciliberti, S. et al. (1999). In real or artificial life, is evolutionary progress in a closed system possible? what'snew— lenski et al. *Proceedings of the Genetic and Evolutionary Computation Conference*. 51

Cleland, C. E. (2002). Methodological and epistemic differences between historical science and experimental science. *Philosophy of Science*, 69(3):447–451. 48

Conway, J., Guy, R., and Berlekamp, E. (2003). *Winning Ways for Your Mathematical Plays, Vol. 2*. CRC Press. 46

Crow, J. F. and Kimura, M. (1965). Evolution in sexual and asexual populations. *American Naturalist*, pages 439–450. 19

Darwin, C. (1859). On the origin of the species by natural selection. 18

Darwin, C. (1868). *The variation of animals and plants under domestication*, volume 2. O. Judd. 18

Darwin, C. (1888). *The descent of man, and selection in relation to sex*, volume 1. Murray. 20

Darwin, C. and Wallace, A. (1858). On the tendency of species to form varieties; and on the perpetuation of varieties and species by natural means of selection. *Journal of the proceedings of the Linnean Society of London. Zoology*, 3(9):45–62. 7, 18

David, B., Cohen, G., and Dessaint, J.-P. (2009). Génétique environnementale et épigénétique ou la réhabilitation de jb lamarck. In *XXXIIème Journée du GAICRM, Suze-La-Rousse (Drôme, France), 21 mars 2009*. 23

Dawkins, R. (1982). *The extended phenotype*. Oxford Paperbacks. 22

Dawkins, R. (1997). Human chauvinism. 78

Dawkins, R. (2004). Extended phenotype–but not too extended. a reply to laland, turner and jablonka. *Biology and Philosophy*, 19(3):377–396. 23

Dawkins, R. et al. (1989a). *The selfish gene*. Oxford university press. 21, 27, 186

Dawkins, R. et al. (1989b). *The selfish gene*. Oxford university press. 21

de Beer, G. (1964). Mendel, darwin, and fisher (1865-1965). *Notes and Records of the Royal Society of London*, 19(2):192–226. 18

de Buffon, G.-L. L. C. (1753). *Histoire naturelle générale et particulière*, volume 4. Dufart. 16

de Maupertuis, P. L. M. (1751). *Vénus physique*. 17

de Maupertuis, P.-L. M. and Trublet, N. C. J. (1754). *Essai sur la formation des corps organisés*. 17

Deutsch, A. and Dormann, S. (2005). *Cellular Automaton Modelling of Biological Pattern Formation: Characterization, Applications and Analysis*. Birkhauser. 46

Doucette, J. (2010). Novelty-based fitness measures in genetic programming. *Master of science in computer science, Dalhouise University*. 37

Fisher, R. A. (1930). *The genetical theory of natural selection: a complete variorum edition*. Oxford University Press. 18

Fitzgerald, J. and Ryan, C. (2014). On size, complexity and generalisation error in gp. In *Proceedings of the 2014 conference on Genetic and evolutionary computation*, pages 903–910. ACM. 82

Floreano, D. and Urzelai, J. (2000). Evolutionary robots with on-line self-organization and behavioral fitness. *Neural Networks*, 13(4):431–443. 52

Fogel, D. B. (2006). Nils barricelli-artificial life, coevolution, self-adaptation. *IEEE Computational Intelligence Magazine*, 1(1):41–45. 31, 49

Frank, A. and Asuncion, A. (2010). UCI Machine Learning Repository. 100

Fraser, A. S. (1957). Simulation of genetic systems by automatic digital computers i. introduction. *Australian Journal of Biological Sciences*, 10(2):484491. 48

Freund, Y. (1990). Boosting a weak learning algorithm by majority. In *COLT*, volume 90, pages 202–216. 37

Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *Icml*, volume 96, pages 148–156. 37

Ganguly, N., Sikdar, B., Deutsch, A., Canright, G., and Chaudhuri, P. (2003). A survey on cellular automata. Technical Report 9, Centre for High Performance Computing,Dresden University of Technology. 47

Godfrey-Smith, P. (2007). Conditions for evolution by natural selection. *The Journal of Philosophy*, 104(10):489–516. 18

Goncalves, I. and Silva, S. (2013). Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In Krawiec, K., Moraglio, A., Hu, T., Uyar, A. S., and Hu, B., editors, *Proceedings of the 16th European Conference on Genetic Programming, EuroGP 2013*, volume 7831 of *LNCS*, pages 73–84, Vienna, Austria. Springer Verlag. 95, 97, 98, 99, 100, 108, 122, 168

Gonçalves, I., Silva, S., Melo, J. B., and Carreiras, J. M. (2012). Random sampling technique for overfitting control in genetic programming. In *Genetic Programming*, pages 218–229. Springer. 95

Gould, N. E.-S. J. (1972). Punctuated equilibria: an alternative to phyletic gradualism. 8, 114, 159

Gould, S. J. (1988). *On replacing the idea of progress with an operational notion of directionality*. na. 30

Gould, S. J. and Eldredge, N. (1977). Punctuated equilibria: the tempo and mode of evolution reconsidered. *Paleobiology*, 3(02):115–151. 9, 27, 76, 186

Gould, S. J. and Vrba, E. S. (1982). Exaptation-a missing term in the science of form. *Paleobiology*, pages 4–15. 28, 160, 185

Gross, M. R. (1991). Salmon breeding behavior and life history evolution in changing environments. *Ecology*, 72(4):1180–1186. 27

Haasdijk, E., Bredeche, N., and Eiben, A. (2014). Combining environment-driven adaptation and task-driven optimisation in evolutionary robotics. *PloS one*, 9(6):e98466. 41

Hamilton, W. D. (1964). The genetical evolution of social behaviour. ii. *Journal of theoretical biology*, 7(1):17–52. 20

Heard, E. and Martienssen, R. A. (2014). Transgenerational epigenetic inheritance: myths and mechanisms. *Cell*, 157(1):95–109. 29

Hebb, D. O. (1949). The organization of behavior. 51

Hillis, W. D. (1990). Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D: Nonlinear Phenomena*, 42(1):228–234. 41

Hoaglin, D. C., Mosteller, F., and Tukey, J. W., editors (1983). *Understanding robust and exploratory data analysis*. Wiley series in probability and mathematical statistics. Wiley-Interscience. 124

Holland, J. H. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press. 33

Hornby, G. S. (2006). Alps: the age-layered population structure for reducing the problem of premature convergence. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 815–822. ACM. 41

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417. 71

Huxley, J. (1944). Evolution: the modern synthesis. 18

Iba, H. (1999). Bagging, boosting, and bloating in genetic programming. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2*, pages 1053–1060. Morgan Kaufmann Publishers Inc. 37

Ilachinski, A. (2001). *Cellular automata: a discrete universe*. World Scientific. 44

Jablonka, E. and Lamb, M. J. (2002). The changing concept of epigenetics. *Annals of the New York Academy of Sciences*, 981(1):82–96. 23

Jablonka, E., Lamb, M. J., and Zeligowski, A. (2005a). *Evolution in Four Dimensions, revised edition: Genetic, Epigenetic, Behavioral, and Symbolic Variation in the History of Life*. MIT press. ix, 25

Jablonka, E., Lamb, M. J., and Zeligowski, A. (2005b). *Evolution in four dimensions, revised edition: Genetic, epigenetic, behavioral, and symbolic variation in the history of life*. MIT press. 22

Jablonka, E., Lamb, M. J., and Zeligowski, A. (2005c). *Evolution in Four Dimensions, revised edition: Genetic, Epigenetic, Behavioral, and Symbolic Variation in the History of Life*. MIT press. 24, 29, 140, 153, 154, 160, 182, 184

Jablonka, E., Oborny, B., Molnar, I., Kisdi, E., Hofbauer, J., and Czaran, T. (1995). The adaptive advantage of phenotypic memory in changing environments. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 350(1332):133–141. 10, 28

Jayat, D. (2010). *Les animaux ont-ils une culture?* EDP sciences. 24

Jepson, G. L., Mayr, E., Simpson, G. G., et al. (1949). Genetics, paleontology, and evolution. *Genetics, paleontology, and evolution.* 9

Johnson, N. A., Lahti, D. C., and Blumstein, D. T. (2012). Combating the assumption of evolutionary progress: lessons from the decay and loss of traits. *Evolution: Education and Outreach*, 5(1):128–138. 30

Jost, L. (2006). Entropy and diversity. *Oikos*, 113(2):363–375. 141

Keijzer, M. (2003). Improving symbolic regression with interval arithmetic and linear scaling. In *Genetic programming*, pages 70–82. Springer. 39

Koonin, E. and Aravind, L. (2002). Origin and evolution of eukaryotic apoptosis: the bacterial connection. *Cell death and differentiation*, 9(4):394–404. 68

Kowaliw, T. and Banzhaf, W. (2009). Augmenting artificial development with local fitness. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 316–323. 47

Kowaliw, T., Grogono, P., and Kharma, N. (2004). Bluenome: A novel developmental model of artificial morphogenesis. In et al., K. D., editor, *6th Conference on Genetic and Evolutionary Computation (GECCO)*. 46, 47

Kowaliw, T., Grogono, P., and Kharma, N. (2007). The evolution of structural form through artificial embryogeny. In *IEEE Symposium on Artificial Life (IEEE-ALIFE)*, pages 425–432. IEEE. 46

Koza, J. R. (1990). *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*. Stanford University, Department of Computer Science. 34

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA. 95

Koza, J. R. (1994). Genetic programming ii: Automatic discovery of reusable subprograms. *Cambridge, MA, USA*. xiii, 8, 32, 35

Krawiec, K. (2016). Semantic genetic programming. In *Behavioral Program Synthesis with Genetic Programming*, pages 55–66. Springer. 38

La Cava, W., Helmuth, T., Spector, L., and Danai, K. (2015). Genetic programming with epigenetic local search. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1055–1062. ACM. 40

La Cava, W. and Spector, L. (2015). Inheritable epigenetics in genetic programming. In *Genetic Programming Theory and Practice XII*, pages 37–51. Springer. 40

Lachmann, M. and Jablonka, E. (1996). The inheritance of phenotypes: an adaptation to fluctuating environments. *Journal of theoretical biology*, 181(1):1–9. 28

Laland, K., Matthews, B., and Feldman, M. W. (2016). An introduction to niche construction theory. *Evolutionary Ecology*, pages 1–12. 29

Lamarck, J.-B.-P. (1809). *Philosophie zoologique*. 16, 17, 187

Langdon, W. B. (2000). Quadratic bloat in genetic programming. In Whitley, D., Goldberg, D., Cantu-Paz, E., Spector, L., Parmee, I., and Beyer, H.-G., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 451–458, Las Vegas, Nevada, USA. Morgan Kaufmann. 41

Langdon, W. B. and Poli, R. (1998). *Fitness causes bloat*. Springer. 113

Langton, C. G. (1984). Self-reproduction in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1):135–144. 47

Langton, C. G. (1986). Studying artificial life with cellular automata. *Physica D: Nonlinear Phenomena*, 22(1-3):120–149. 48

Langton, C. G. et al. (1989). *Artificial life*. Addison-Wesley Publishing Company Redwood City, CA. 8, 47

Larkin, F. (2010). *Artificial Evolution Approaches to Address the Data Challenges Encountered During Financial Forecasting*. PhD thesis, University of Limerick. 41

Latour, B. (1989). Pasteur et pouchet: hétérogenèse de lhistoire des sciences. *M. Serres, Éléments dhistoire des sciences, Bordas, Paris*. 18

Lehman, J. and Stanley, K. O. (2008). Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336. 51

Lehman, J. and Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223. 37

Leigh, E. G. (1981). The average lifetime of a population in a varying environment. *Journal of Theoretical Biology*, 90(2):213–239. 27

Lenski, R. E., Ofria, C., Pennock, R. T., and Adami, C. (2003). The evolutionary origin of complex features. *Nature*, 423(6936):139–144. 50

Lessin, D., Fussell, D., and Miikkulainen, R. (2013). Open-ended behavioral complexity for evolved virtual creatures. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 335–342. ACM. 41

Levins, R. (1968). *Evolution in changing environments: some theoretical explorations*. Number 2. Princeton University Press. 26, 53

Levins, R. and Lewontin, R. C. (1985). *The dialectical biologist*. Harvard University Press. 18

Lipson, H., Pollack, J. B., and Suh, N. P. (2002). On the origin of modular variation. *Evolution*, 56(8):1549–1556. 52, 139

Lloyd, E. (2012). Units and Levels of Selection. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Spring 2012 edition. 136

Loveard, T. and Ciesielski, V. (2001). Representing classification problems in genetic programming. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 2, pages 1070–1077. IEEE. 184

Lynch, M. and Conery, J. S. (2003). The origins of genome complexity. *science*, 302(5649):1401–1404. 82

Maclin, R. and Opitz, D. (1997). An empirical evaluation of bagging and boosting. *AAAI/IAAI*, 1997:546–551. 37

Maley, C. (1999). Four steps toward open-ended evolution. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2*, pages 1336–1343. Morgan Kaufmann Publishers Inc. 8

Mayr, E. (1942). *Systematics and the origin of species, from the viewpoint of a zoologist*. Harvard University Press. 9, 187

Mayr, E. (1982). Speciation and macroevolution. *Evolution*, pages 1119–1132. 9, 114

McClintock, B. (1993). *The significance of responses of the genome to challenge*. Singapore: World Scientific Pub. Co. 160

McPhee, N. F., Ohs, B., and Hutchison, T. (2008). Semantic building blocks in genetic programming. In *European Conference on Genetic Programming*, pages 134–145. Springer. 38

McShea, D. W. (1991). Complexity and evolution: what everybody knows. *Biology and Philosophy*, 6(3):303–324. 30

Medernach, D., Fitzgerald, J., Azad, R., and Ryan, C. (2015a). Wave: Incremental erosion of residual error. In *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference*, pages 1285–1292. ACM. 167, 168, 174

Medernach, D., Fitzgerald, J., Carrignon, S., and Ryan, C. (2015b). Evolutionary progress in heterogenous cellular automata (hetca). In *Proceedings of the European Conference on Artificial Life 2015 (ECAL 2015)*, volume 13, pages 512–519. 135, 142

Medernach, D., Kowaliw, T., Ryan, C., and Doursat, R. (2013). Long-term evolutionary dynamics in heterogeneous cellular automata. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 231–238. ACM.

Mendel, G. (1865). *Experiments in plant hybridisation*. 18

Miller, J. (2004). Evolving a self-repairing, self-regulating, french flag organism. In et al., K. D., editor, *6th Conference on Genetic and Evolutionary Computation (GECCO)*, pages 129–139. Springer-Verlag. 46

Milligan, B. G. (1986). Punctuated evolution induced by ecological change. *American Naturalist*, pages 522–532. 9, 114, 159

Misevic, D., Ofria, C., and Lenski, R. E. (2010). Experiments with digital organisms on the origin and maintenance of sex in changing environments. *Journal of heredity*, 101(suppl 1):S46–S54. 52

Moraglio, A., Krawiec, K., and Johnson, C. G. (2012). Geometric semantic genetic programming. In *International Conference on Parallel Problem Solving from Nature*, pages 21–31. Springer. 38, 39, 164

Muhammad Atif Azad, R., Medernach, D., and Ryan, C. (2014). Efficient approaches to interleaved sampling of training data for symbolic regression. In *Nature and Biologically Inspired Computing (NaBIC), 2014 Sixth World Congress on*, pages 176–183. IEEE. 168

Muller, G. B. (2007). Evo-devo: extending the evolutionary synthesis. *Nat Rev Genet*, 8(12):943–949. 10.1038/nrg2219. 29

Ofria, C. and Wilke, C. O. (2004). Avida: A software platform for research in computational evolutionary biology. *Artificial life*, 10(2):191–229. 50

Okada, N., Sasaki, T., Shimogori, T., and Nishihara, H. (2010). Emergence of mammals by emergency: exaptation. *Genes to Cells*, 15(8):801–812. 28

Okasha, S. (2006a). *Evolution and the levels of selection*, volume 16. Clarendon Press Oxford. 136

Okasha, S. (2006b). The levels of selection debate: philosophical issues. *Philosophy Compass*, 1(1):74–85. 22

Okasha, S. (2008). Biological altruism. *Stanford encyclopedia of philosophy*. 185

Oliveira, L. O., Otero, F. E., Pappa, G. L., and Albinati, J. (2014). Sequential symbolic regression with genetic programming. 39, 114

Olson, E. T. (1997). The ontological basis of strong artificial life. *Artificial Life*, 3(1):29–39. 48

O'Neill, M., Vanneschi, L., Gustafson, S., and Banzhaf, W. (2010). Open issues in genetic programming. *Genetic Programming and Evolvable Machines*, 11(3/4):339–363. Tenth Anniversary Issue: Progress in Genetic Programming and Evolvable Machines. 41

Ostoya, P. (1954). Maupertuis et la biologie. *Revue d'histoire des sciences et de leurs applications*, 7(1):60–78. 17

ONeill, M., Vanneschi, L., Gustafson, S., and Banzhaf, W. (2010). Open issues in genetic programming. *Genetic Programming and Evolvable Machines*, 11(3-4):339–363. 43

Pagie, L. and Hogeweg, P. (1997). Evolutionary consequences of coevolving targets. *Evolutionary computation*, 5(4):401–418. 162

Pan, Z. and Reggia, J. (2010). Computational discovery of instructionless self-replicating structures in cellular automata. *Artificial Life*, 16:1064–5462. 46, 47

Paris, G., Robilliard, D., and Fonlupt, C. (2001). Applying boosting techniques to genetic programming. In *International Conference on Artificial Evolution (Evolution Artificielle)*, pages 267–278. Springer. 37

Paris, G., Robilliard, D., and Fonlupt, C. (2003). Genetic programming with boosting for ambiguities in regression problems. In *European Conference on Genetic Programming*, pages 183–193. Springer. 37

Pennell, M. W., Harmon, L. J., and Uyeda, J. C. (2014). Is there room for punctuated equilibrium in macroevolution? *Trends in ecology & evolution*, 29(1):23–32. 27

Perry, S. F. and Oliveira, E. S. (2010). Respiration in a changing environment. *Respiratory physiology & neurobiology*, 173:S20–S25. 28

Pigliucci, M. (2007). Do we need an extended evolutionary synthesis? *Evolution*, 61(12):2743–2749. 9, 187

Piszcz, A. and Soule, T. (2006). Genetic programming: Optimal population sizes for varying complexity problems. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 953–954. ACM. 119

Poli, R. (2003). A simple but theoretically-motivated method to control bloat in genetic programming. In *Genetic programming*, pages 204–217. Springer. 122, 166

Poli, R., Langdon, W., and McPhee, N. (2008). *A Field Guide To Genetic Programming*. Lulu Enterprises. 63

Poli, R. and McPhee, N. (2008). Parsimony pressure made easy. In Keijzer, M., Antoniol, G., Congdon, C. B., Deb, K., Doerr, B., Hansen, N., Holmes, J. H., Hornby, G. S., Howard, D., Kennedy, J., Kumar, S., Lobo, F. G., Miller, J. F., Moore, J., Neumann, F., Pelikan, M., Pollack, J., Sastry, K., Stanley, K., Stoica, A., Talbi, E.-G., and Wegener, I., editors, *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1267–1274, Atlanta, GA, USA. ACM. 41

Price, G. R. (1972). Extension of covariance selection mathematics. *Annals of human genetics*, 35(4):485–490. 20

Queller, D. C., Rong, S., and Liao, X. (2015). Some agreement on kin selection and eusociality? *PLoS Biol*, 13(4):e1002133. 20

Ray, T. S. (1992). Evolution, ecology and optimization of digital organisms. Technical report, Technical Report 92-08-042, Santa Fe Institute, Santa Fe, NM. 8, 49

Rechenberg, I. (1964). Kybernetische lösungsansteuerung einer experimentellen forschungsaufgabe. In *Annual Conference of the WGLR at Berlin in September*, volume 35. 33

Ruberto, S., Vanneschi, L., Castelli, M., and Silva, S. (2014). Esagp–a semantic gp framework based on alignment in the error space. In *Genetic Programming*, pages 150–161. Springer. 38

Ruse, M. (1980). Charles darwin and group selection. *Annals of science*, 37(6):615–630. 20

Ryan, C., Collins, J., and Neill, M. O. (1998). Grammatical evolution: Evolving programs for an arbitrary language. In *European Conference on Genetic Programming*, pages 83–96. Springer. 36

Sayama, H. (1998). Spontaneous evolution of self reproducing loops in cellular automata. *InterJournal complex systems*, 236:363–374. 47

Sayama, H. (2004). Self-protection and diversity in self-replicating cellular automata. *Artificial Life*, 10(1):83–98. 46

Schaffer, W. M. (1974). Optimal reproductive effort in fluctuating environments. *American Naturalist*, pages 783–790. 27

Shanahan, T. (2000). Evolutionary progress? *BioScience*, 50(5):451–459. 29

Shanahan, T. (2012). Evolutionary progress: conceptual issues. *eLS*. 30, 91, 135, 182

Shanken, E. A. (1998). Life as we know it and or life as it could be: Epistemology and the ontology ontogeny of artificial life. *Leonardo*, pages 383–388. 8

Sheehan, L. (2000). *Ecoga: a novel self-adapting ecologically inspired genetic algorithm.* 43

Sipper, M. (1998). Computing with cellular automata: Three cases for nonuniformity. *Phys. Rev. E*, 57:3589–3592. 46

Sipper, M. and Tomassini, M. (1999). Computation in artificially evolved, non-uniform cellular automata. *Theoretical Computer Science*, 217(1):81–98. 46

Smith, J. M. (1990). Models of a dual inheritance system. *Journal of Theoretical Biology*, 143(1):41–53. 28

Smith, J. M. and Szathmary, E. (1997). *The major transitions in evolution.* Oxford University Press. 68

Soule, T. and Foster, J. A. (1998). Effects of code growth and parsimony pressure on populations in genetic programming. *Evolutionary Computation*, 6(4):293–309. 41

Tanev, I. and Yuta, K. (2003). Epigenetic programming: an approach of embedding epigenetic learning via modification of histones in genetic programming. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 4, pages 2580–2587. IEEE. 7, 39

Taylor, T., Bedau, M., Channon, A., Ackley, D., Banzhaf, W., Beslon, G., Dolson, E., Froese, T., Hickinbotham, S., Ikegami, T., et al. (2016). Open-ended evolution: Perspectives from the oee1 workshop in york. *Artificial Life*. 49, 51

Thalmann, D. (2005). Autonomy. In *ACM SIGGRAPH 2005 Courses*, page 5. ACM. 48

Tufte, G. (2006). Gene regulation mechanisms introduced in the evaluation criteria for a hardware cellular development system. In *NASA/ESA Conference on Adaptive Hardware and Systems*, pages 137–144. 46

Tuomisto, H. (2010a). A consistent terminology for quantifying species diversity? yes, it does exist. *Oecologia*, 164(4):853–860. 141

Tuomisto, H. (2010b). A diversity of beta diversities: straightening up a concept gone awry. part 1. defining beta diversity as a function of alpha and gamma diversity. *Ecography*, 33(1):2–22. 141

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59(236):433–460. 31

Vanneschi, L., Castelli, M., and Silva, S. (2010). Measuring bloat, overfitting and functional complexity in genetic programming. In Branke, J., Pelikan, M., Alba, E., Arnold, D. V., Bongard, J., Brabazon, A., Branke, J., Butz, M. V., Clune, J., Cohen, M., Deb, K., Engelbrecht, A. P., Krasnogor, N., Miller, J. F., O'Neill, M., Sastry, K., Thierens, D., van Hemert, J., Vanneschi, L., and Witt, C., editors, *GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 877–884, Portland, Oregon, USA. ACM. 41

Vichniac, G., Tamayo, P., and Hartman, H. (1986). Annealed and quenched inhomogeneous cellular automata (inca). *Journal of Statistical Physics*, 45:875–883. 46

Visser, J., Hermisson, J., Wagner, G. P., Meyers, L. A., Bagheri-Chaichian, H., Blanchard, J. L., Chao, L., Cheverud, J. M., Elena, S. F., Fontana, W., et al. (2003). Perspective: evolution and detection of genetic robustness. *Evolution*, 57(9):1959–1972. 29

Vladislavleva, E. J., Smits, G. F., and den Hertog, D. (2009a). Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. *IEEE Transactions on Evolutionary Computation*, 13(2):333–349. 41

Vladislavleva, E. J., Smits, G. F., and Den Hertog, D. (2009b). Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. *Evolutionary Computation, IEEE Transactions on*, 13(2):333–349. 122, 166

Von Neumann, J., Burks, A. W., et al. (1966). Theory of self-reproducing automata. *IEEE Transactions on Neural Networks*, 5(1):3–14. 44

Vostinar, A. E., Dolson, E. L., Wiser, M. J., and Ofria, C. (2016). Identifying necessary components for open-ended evolution. 51

West-Eberhard, M. J. (2005). Developmental plasticity and the origin of species differences. *Proceedings of the National Academy of Sciences*, 102(suppl 1):6543–6549. 29

Wolfram, S. (1984). Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1-2):1–35. 44

Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media. 44, 46, 75

Yaeger, L. (1994). Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or poly world: Life in a new context. In *SANTA FE INSTITUTE STUDIES IN THE SCIENCES OF COMPLEXITY-PROCEEDINGS VOLUME-*, volume 17, pages 263–263. ADDISON-WESLEY PUBLISHING CO. 50

Yeh, I. C. (1998). Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12):1797 – 1808. 100

Yinusa, A. and Nehaniv, C. (2011). Study of inheritable mutations in von neumann self-reproducing automata using the golly simulator. In *2011 IEEE Symposium on Artificial Life (ALIFE)*, pages 211–217. 46

Yu, T. (2007). Program evolvability under environmental variations and neutrality. In *Proceedings of the 9th annual conference companion on Genetic and evolutionary computation*, pages 2973–2978. ACM. 52, 139

Zhang, B.-T. and Mühlenbein, H. (1995). Balancing accuracy and parsimony in genetic programming. *Evolutionary Computation*, 3(1):17–38. 41