LUCA MANZONI

DYNAMICS OF BIOINSPIRED COMPUTATION

Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano-Bicocca
Dottorato di Ricerca in Informatica – Ciclo XXV

DYNAMICS *of* BIOINSPIRED COMPUTATION

LUCA MANZONI

Advisor: Leonardo Vanneschi
Tutor: Gabriella Pasi
Coordinator: Stefania Bandini

Real life is, to most men,
a long second-best, a perpetual compromise
between the ideal and the possible;

but the world of pure reason knows no compromise,
no practical limitations, no barrier to the creative activity
embodying in splendid edifices
the passionate aspiration after the perfect
from which all great work springs.

Remote from human passions,
remote even from the pitiful facts of nature,
the generations have gradually created an ordered cosmos,
where pure thought can dwell as in its natural home,
and where one, at least, of our nobler impulses
can escape from the dreary exile of the actual world.

– Bertrand Russell (1872-1967)

ABSTRACT

Since its birth, Computer Science has been inspired by natural phenomena. Two main approaches where developed through the years. The first one is concerned with the study of the properties nature-inspired models that can also be used to model nature. Examples in this fields are Cellular Automata and Reaction Systems. The second one uses nature-inspired model to perform optimization tasks where exact algorithms are not applicable. Evolutionary algorithms, like Genetic Algorithms and Genetic Programming are some of the most prominent examples.

The main aim of this thesis is the study of the dynamics of four different nature-inspired models with the goal of providing both an improvement in the single areas and a cross-pollination of methods and techniques. The four chosen models are:

- *Genetic Algorithms*. A traditional optimization techniques that is inspired by the Darwinian theory of evolution.
- *Genetic Programming*. A more recent technique, similar to classical Genetic Algorithms, that uses programs instead of fixed length binary strings as a representation method.
- *Reaction Systems*. A recently developed formalism inspired by chemical reactions.
- *Cellular Automata*. Am extensively studied model made of a lattice of identical automata that can exchange information only locally.

As for Genetic Algorithm, one of their main operators, the crossover, was studied. In particular, the minimum number of iteration of the crossover operator needed to produce certain individuals was investigated.

As for Genetic Programming, two new measures to quantify the quality of the solution learned were developed. To better understand the dynamics of Genetic Programming a new benchmark, inspired by a similar benchmark in Genetic Algorithms, was introduced. Finally, a method to reduce the worst case space requirements of Semantic Genetic Programming from exponential to polynomial was devised.

Some combinatorial properties of Reaction Systems were studied. Furthermore, an evolutionary version of Reaction Systems to be used for optimization was introduced. This new algorithm proved to have performances comparable to current state-of-the-art machine learning algorithms.

The dynamical and computational properties of a variation of classical Cellular Automata, fully-Asynchronous Cellular Automata, have been studied. Furthermore, a first step in providing a more general framework to study asynchronicity in Cellular Automata has been made with the introduction of $\mathfrak{m}$-Asynchronous Cellular Automata.

This thesis provides both improvements in these four different areas and a first step toward an exchange of methods and techniques between those areas.

PUBLICATIONS

Some ideas and figures have appeared previously in the following publications:

1. Alberto Dennunzio, Enrico Formenti, and Luca Manzoni.
   Computing Issues of Asynchronous CA.
   Accepted in *Fundamenta Informaticae*.

2. Alberto Dennunzio, Enrico Formenti, Luca Manzoni, and Giancarlo Mauri.
   m-asynchronous cellular automata.
   In Georgios Ch. Sirakoulis and Stefania Bandini, editors, *Cellular Automata - 10th International Conference on Cellular Automata for Research and Industry, ACRI 2012*, volume 7495 of *Lecture Notes in Computer Science*, pages 653–662, Santorini, Greece, September 2012. Springer.
   Second International Workshop on Asynchronous Cellular Automata.

3. Mauro Castelli, Luca Manzoni, and Leonardo Vanneschi.
   Parameter tuning of evolutionary reactions systems.
   In *Genetic and Evolutionary Computation Conference, GECCO 2012*, pages 727–734, Philadelphia, USA, July 2012. ACM.

4. Luca Manzoni, Leonardo Vanneschi, and Giancarlo Mauri.
   A distance between populations for one-point crossover in genetic algorithms.
   Theoretical Computer Science, 429:213–222, 2012.

5. Luca Manzoni.
   Asynchronous cellular automata and dynamical properties.
   Natural Computing, 11(2):269–276, 2012.

6. Luca Manzoni, Mauro Castelli, and Leonardo Vanneschi.
   Evolutionary reaction systems.
   In Mario Giacobini, Leonardo Vanneschi, and William S. Bush, editors, *Evolutionary Computation, Machine Learning and Data Mining in Computational Biology, EvoBIO 2012*, volume 7246 of *Lecture Notes in Computer Science*, pages 13–25, Màlaga, Spain, April 2012. Springer.

7. Jérôme Chandesris, Alberto Dennunzio, Enrico Formenti, and Luca Manzoni.
   Computational aspects of asynchronous cellular automata.
   In Mauri Giancarlo and Leporati Alberto, editors, *Developments in Language Theory DLT 2011*, volume 6795 of *Lecture Notes in Computer Science*, pages 466–468, Milano, Italy, July 2011. Springer.

8. Leonardo Vanneschi, Mauro Castelli, and Luca Manzoni.
   The k landscapes: a tunably difficult benchmark for genetic programming.
   In *Genetic and Evolutionary Computation Conference, GECCO 2011*, pages 1467–1474, Dublin, Ireland, July 2011. ACM.

9. Mauro Castelli, Luca Manzoni, Sara Silva, and Leonardo Vanneschi.
   A quantitative study of learning and generalization in genetic programming.
   In Sara Silva, James A. Foster, Miguel Nicolau, Penousal Machado, and Mario Giacobini, editors, *Genetic Programming - 14th European Conference, EuroGP 2011*, volume 6621 of *Lecture Notes in Computer Science*, pages 25–36, Torino, Italy, April 2011. Springer.

10. Luca Manzoni.
    Some formal properties of asynchronous cellular automata.
    In Stefania Bandini, Sara Manzoni, Hiroshi Umeo, and Giuseppe Vizzari, editors, *Cellular Automata - 9th International Conference on Cellular Automata for Research and Industry, ACRI 2010*, volume 6350 of *Lecture Notes in Computer Science*, pages 419–428, Ascoli Piceno, Italy, September 2010. Springer.
    First International Workshop on Asynchronous Cellular Automata.

11. Luca Manzoni, Leonardo Vanneschi, and Giancarlo Mauri.
Definition of a crossover based distance for genetic algorithms.
In Martin Pelikan and Jürgen Branke, editors, *Genetic and Evolutionary Computation Conference, GECCO 2010*, pages 1473–1474, Portland, Oregon (USA), July 2010. ACM.

Some ideas and figures have appeared in the following preprints:

1. Mauro Castelli, Luca Manzoni, and Leonardo Vanneschi.
An Efficient Genetic Programming System with Geometric Semantic Operators and its Application to Human Oral Bioavailability Prediction.
arXiv:1208.243v1 [cs.NE].

2. Jérôme Chandesris, Alberto Dennunzio, Enrico Formenti, and Luca Manzoni.
Computational Aspects of Asynchronous CA.
arXiv:1105.0065v1 [cs.FL].

Other publications realized during the doctorate are the followings:

1. James McDermott, David R. White, Sean Luke, Luca Manzoni, Mauro Castelli, Leonardo Vanneschi, Wojciech Jaśkowski, Krzysztof Krawiec, Robin Harper, Kenneth De Jong, and Una-May O'Reilly.
Genetic programming needs better benchmarks.
In *Genetic and Evolutionary Computation Conference, GECCO 2012*, pages 791–798, Philadelphia, USA, July 2012. ACM.

2. Mauro Castelli, Luca Manzoni, and Leonardo Vanneschi.
A method to reuse old populations in genetic algorithms.
In Luis Antunes and Helena Sofia Pinto, editors, *XV Portuguese Conference on Artificial Intelligence EPIA 2011*, volume 7026 of *Lecture Notes in Computer Science*, pages 138–152, Lisbon, Portugal, October 2011. Springer.

3. Mauro Castelli, Luca Manzoni, and Leonardo Vanneschi.
Reinsertion of old genetic material: Second chance gp.
In *XV Portuguese Conference on Artificial Intelligence EPIA 2011*, Lisbon, Portugal, October 2011.

4. Stefania Bandini, Lorenza Manenti, Luca Manzoni, and Sara Manzoni.
Dealing with crowd crystals in mas-based crowd simulation: a proposal.
In Roberto Pirrone and Filippo Sorbello, editors, *International Conference on Advances in Artificial Intelligenece AI*IA 2011*, volume 6934 of *Lecture Notes in Computer Science*, pages 92–103, Palermo, Italy, September 2011. Springer.

5. Mauro Castelli, Luca Manzoni, and Leonardo Vanneschi.
The effect of selection from old populations in genetic algorithms.
In *Genetic and Evolutionary Computation Conference, GECCO 2011*, pages 161–162, Dublin, Ireland, July 2011. ACM.

6. Mauro Castelli, Luca Manzoni, and Leonardo Vanneschi.
Multi objective genetic programming for feature construction in classification problems.
In Carlos A. Coello Coello, editor, *Learning and Intelligent OptimizatioN, LION 5*, volume 6683 of *Lecture Notes in Computer Science*, pages 503–506, Rome, Italy, January 2011. Springer.

7. Lorenza Manenti, Luca Manzoni, and Sara Manzoni.
Towards an application of graph structure analysis to a mas-based model of proxemic distances in pedestrian systems.
In Andrea Omicini and Mirko Viroli, editors, *Workshop Nazionale degli Oggetti agli Agenti WOA 2010*, volume 621, Rimini, Italy, September 2010. CEUR Workshop Proceedings.

8. Mauro Castelli, Luca Manzoni, Sara Silva, and Leonardo Vanneschi.
A comparison of the generalization ability of different genetic programming frameworks.
In *IEEE Congress on Evolutionary Computation, CEC 2010*, pages 94–101, Barcelona, Spain, July 2010.

ACKNOWLEDGMENTS

There are many people that I want and need to thank for their support in the years of my doctorate.

I want to thank my advisor, Leonardo Vanneschi, whose support and guidance during my doctorate was essential, Alberto Dennunzio and Enrico Formenti, for helping me in the research on Cellular Automata. I also want to thank many people that I met in the Department of Computer Science at the Università degli Studi di Milano-Bicocca: Stefania Bandini, Daniela Besozzi, Gianpiero Cattaneo, Sara Manzoni, Giancarlo Mauri, Gabriella Pasi, Giuseppe Vizzari, and many others.

I want to thank all the other PhD students and postdocs that I have met during these years: Stefano Beretta, Mauro Castelli, Andrea Gorrini, Lorenza Manenti, Carlo Maj, Luca "Izzy" Panziera, Yuri Pirola, Antonio Enrico Porreca (aka "Trippo"), and the "new" PhD students Andrea Citrolo, Riccardo Colombo, and Marco Nobile.

I want to thank the people of that I have met during my visiting periods outside Italy: Sara Silva, Alberto Moraglio, Andrea Mambrini, Antonio Lima, Hiroshi Umeo and all the people from INESC-ID and ISEGI in Lisbon, from the Department of Computer Science of the University of Birmingham, and from the Faculty of Information and Communication Engineering in the Osaka Electro-Communication University.

I want also to thank all my family, my mother, my father, my sister Sara, my brother Andrea and also Anna, Hermes and Rina.

I apologize to all the people that are not in this list but that deserve to be thanked. I know that, despite all the good intentions, I have certainly forgot someone.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

Since its infancy, Computer Science has been inspired by natural processes. In fact, between the 1940s and the 1950s, two different approaches to the use of natual inspiration in computing were born.

The first approach is the use of nature-inspired *computational* models as models for natural phenomena. It is important to focus on the computational part of the model. A model that is computable allows the scientist to produce predictions of natural phenomena and the engineer to perform simulations *in silico*.

A second approach is the use of nature-inspired algorithms to solve computational problems in an effective way. Since many combinatorial problems are currently unsolvable by exact methods for instances of any significant size [159], many approximation and/or heuristic methods have been developed [205, 134]. One obvious source of inspiration for solving those problems was nature. In fact, some of these problems have a natural analogue whose solution is efficiently given by a natural phenomenon. Thus, an algorithm mimicking the phenomenon was considered a possible way to efficiently solve the same problem on a computing device [93].

The models and algorithms that has been developed following the two approaches had an history of cross-pollination and sharing of ideas. The same model used one time to predict the outcome of a physical experiment could have been used to solve a combinatorial problem (for example, the recently developed model of Reaction Systems [51, 52]). Nonetheless, a certain distinction of uses exists. For example, Cellular Automata are widely used as a model of physical phenomena (for example for modeling the flow of a liquid [29]) while Genetic Algorithms main use is function optimization [162].

In this thesis the main theme is the study of the dynamics of different bio-inspired models in order to allow the communication and sharing of ideas between different approaches. There are four main approaches studied:

- Genetic Algorithms (GA);
- Genetic Programming (GP);
- Reaction Systems (RS);

- Cellular Automata (CA).

While the first two share a similar heritage and inspiration, the Darwinian theory of evolution [42], and are mainly used in solving optimization problems, the other two were born and are studied as a way to model natural phenomena and do not share a lot. Reaction Systems are a newly developed model by Rozenberg and coworkers [51, 52], while Cellular Automata are one of the oldest models in Computer Science, that number among its creators some of the founders of the discipline, like Von Neumann [210, 149].

The study of the dynamics of these models requires different tools and abilities. Genetic Algorithms have sprout a plethora of different analysis methods involving different mathematical tools that ranges from the knowledge of Markov processes [175] to the modeling using a dynamical process in a continuous space [211]. Genetic Programming's dynamics is difficult to formally study in full generality and our knowledge needs to resort to experimental validation [199]. Due to their young age, Reaction Systems provides a playground for various techniques, since there is still no clear preferred method of research. As one of the oldest method, Cellular Automata have long been studied for their computational abilities [154], for their link with language theory [182], and as a discrete time dynamical system [120]. Thus, there is a large body of theory that can make used of results of ergodic theory, measure theory, and other areas of Mathematics.

Therefore, the study of these four systems seems to be inevitably separated, with little to no possibility to cross-pollination. However, a possible exchange of ideas and techniques could provide deeper insight into current results and the generation of new results with different techniques. In other areas of knowledge this sharing was successful (see, for example, the unexpected connection, called *Monstrous moonshine*, between two different mathematical object [37]). Thus, it could be wise to explore different areas to show if a beneficial exchange could be possible.

In the rest of the chapter we provide an overview of the main contribution of this thesis to the four different considered areas.

1.1 GENETIC ALGORITHMS

GA are a well-known optimization method in which the goal is to maximize (or, equivalently, minimize) a function, called fitness function, whose inputs are usually codified as fixed length binary strings. The algorithms maintains a multiset, called population, of those inputs, called individuals. At every iteration, called generation, four steps are performed. As a first step the fitness function is evaluated for every individual in the population. Then a selection process take place in which a new population of the same size of the original is generated by extracting (with reinsertion) individuals from the origi-

nal population with a probability that depend on their fitness value. Then the crossover takes place. This operation, that is the main topic of this part of the thesis, allows different individuals to exchange information (called genetic material). Finally, an operation, called mutation, that perform bit-flip operations on the individuals according to a given distribution, is performed. After a termination criterion is met - for example a sufficient number of generations has passed - the algorithm returns the best individual found (i.e., the one with the maximum fitness value).

In this part of the thesis the main object of study is the crossover operator, that provides an exchange of genetic material between individuals in the same population. The dynamics of crossover is certainly the most complex to study when compared to the ones generated by selection or mutation.

While, given a population and an individual is easy to understand if the individual can be generated by repeated applications of crossover between individual of the given population, it is less easy to understand how many of those applications are sufficient. The work presented provides an answer to this question. An important aspect of the algorithm that answers this question is that it can be computed efficiently, thus providing the possibility of an application of theory to practice.

The tools applied are not widely used when study GA. They are a weak version of topology - a Čech topology - and the modeling of crossover as a weak closure operator on this weak topological space. This is a different application of existing techniques that model the dynamics of GA and other evolutionary methods as a process inside a topological space [141, 139, 140], thus providing another example on how topological techniques can allow to better model and understand the dynamics of GA.

This part of the thesis is organized as follows: Chapter 2 provides a very brief overview of the approaches to the study of GA theory, focusing on approaches that are historically significant, of widespread use or related to the study performed. Chapter 3 provides an overview of the weak notions of topologies, from Fréchet to Čech topologies. Finally, Chapter 4 describes the proposed model of crossover.

1.2 GENETIC PROGRAMMING

GP is an evolution of GA in which the algorithms modifies, instead of fixed length binary strings, entire programs. Usually these programs are in a lisp-like language and codified as trees [162]. The added complexity of the individuals makes the analysis of the dynamics of GP more complex than the one of GA. Thus, the study of GP often needs an empirical part.

In this part of the thesis we studied the learning process of GP with the definition of measures able to quantify it. In particular, we studied what instances are difficult to learn for GP. Subsequently, we defined a new tunable benchmark, inspired by the famous NK-lanscapes benchmark for GA, with the intention to expose the generation of programs by GP.

A promising new GP technique, the Semantic GP, has recently be defined Moraglio and coworkers [142]. This new technique has nice theoretical properties and produces better solution than standard GP. However, in its original definition it was expensive in term of computational time and space. We defined a new implementation for Semantic GP that was proved to maintain all the qualities of Semantic GP but that makes it applicable in real scenario since its computational requirements have been reduced. In fact, out implementation of Semantic GP proved to be faster than standard GP by an order of magnitude.

This part of the thesis is structured as follows: Chapter 5 provides an introduction to GP and to the main issues that are studies in the remaining chapters. Chapter 6 introduces and studies the measures to quantifies the learning ability of GP. Chapter 7 introduces a new benchmark inspired by NK-landscapes for GA. Finally, Chapter 8 explains our fast Semantic GP implementation.

1.3 REACTION SYSTEMS

RS are a promising newly developed formal model inspired by chemical reactions. In this model the main concept is the one of reactions, that is, an object that can be activated by a set of chemicals, can be inhibited by another set of chemicals, and, when activated and not inhibited, generates new chemicals. By combining more reactions and a starting set of chemical it is possible to obtain dynamics that mimics the one obtained in nature. Furthermore, this system is particularly effective as a mean of representing functions.

Up until now, RS have been studied with different approaches. From a theoretical point of view, we studied their combinatorial properties, in particular focusing on the extremal combinatorics aspect. That is, we were searching properties that always hold for "large enough" RS. Since RS are a new formalism, the early success of this approach can help shaping a set of methods and tools to explore them.

From a practical point of view, the simplicity and easiness of expressing function with RS made them the ideal target for developing an evolutionary algorithm based on them. This algorithm - a variation of GP - that uses RS as a way of representing functions, has been called Evolutionary Reaction Systems (EvoRS). While missing years of study and fine-tuning, it proved to be on par or superior to cur-

rent machine learning techniques. Thus, we now have a rare case of a model that can be studied both as a model of chemical reactions and as a function optimizer, providing a link between two different approaches to bio-inspired computation.

This part of the thesis is organized as follows: Chapter 9 provides an introduction to RS. Chapter 10 studies their properties into an extremal combinatorics setting. Finally, Chapter 11 introduces EvoRS and provide both a comparison with current state-of-the-art techniques and a first study on the parameter tuning of the new algorithm.

1.4 CELLULAR AUTOMATA

CA are one of the oldest models in Computer Science. A CA is a n-dimensional lattice of identical automata that synchronously update their state according to their internal state and the state of their neighbours. The dynamics of CA has long been studied with the tools of discrete dynamical systems [120]. Some properties of interest are, for example, the sensitivity to initial conditions (the effect popularized as the *butterfly effect*), positive expansivity, topological transitivity and many others [120].

Recently, asynchronous models of CA has been introduced. These models can help in the modeling of real-life systems (see, for example, the system in defined in [6]). In these systems, synchronicity is not always present. Thus, the introduction of CA models with different levels of asynchronicity.

We introduced and studied a CA model with the least possible synchronicity, a fully asynchronous CA. In particular its dynamical properties and computational ability have been studied. Furthermore, we introduced a new axiomatic framework for the study of asynchronous CA. Almost all updating schemes that defines asynchronous CA models can be seen as different axioms to impose on a distribution of set of cells to be updated. By studying what are the results obtainable from different sets of axioms we can infer properties that are valid for entire classes of updating schemes. In particular we studied the properties of a particular set of axioms that imposes some "fairness conditions" on the distribution of updates. We called the model obtained m-ACA, for m-Asynchronous CA.

This part of the thesis is structured as follows: Chapter 12 introduces CA and the necessary notions for the remaining chapters. Successively, Chapter 13 defines fully-Asynchronous CA and studies its dynamical properties. Chapter 14 studies the computational properties of fully-Asynchronous CA. Finally, Chapter 15 introduces and studies m-Asynchronous CA.

Part I

A THEORETICAL MODEL OF GENETIC ALGORITHMS

2

INTRODUCTION TO THE THEORY OF GENETIC ALGORITHMS

In this part of the thesis we will briefly recall what are Genetic Algorithms (GA) and introduces the necessary notations. Successively, some of the most prominent approaches to the study of GA are shortly recalled (for a more complete introduction see the book by Rowe [165] or some survey articles [56, 153]).

2.1 INTRODUCTION TO GA

Let $\mathcal{X}$ be a finite set (the *solution space*) and $g : \mathcal{X} \to \mathbb{R}$ be a function (the *objective function*). The goal is to find $\operatorname{argmin}_{x \in \mathcal{X}}\{g(x)\}$, i.e., the *optimum* (or one of the *optima* if it is not unique). To perform this search using GA it is necessary to encode the elements of $\mathcal{X}$ as binary string of fixed length. That is, there exists a surjective map $c : \{0,1\}^n \to \mathcal{X}$. The search performed by GA is on the $\{0,1\}^n$ and not on $\mathcal{X}$.

Let P be a multiset of elements of $\{0,1\}^n$. A GA iteratively performs the following operations depicted in Fig. 1 (each iteration is called a generation):

1. **Fitness evaluation**. For every element $p \in P$ e compute its fitness, that is $g(c(p))$.
2. **Selection**. We perform $|P|$ extractions from P (with reinsertion) with a probability distribution that depends on the fitness values of the elements to obtain a new multiset P'.
3. **Crossover**. In this phase $|P|$ elements of $\{0,1\}^n$ are generated by pairs of elements of P' in the following way:
   a) Let $p_1, p_2 \in P'$ and let $k \in \{1, \dots, n-1\}$ be a randomly chosen integer.
   b) Let p' be defined as $\forall i \in \{1, \dots, k\}$, p' in position i is equal to p_1 in position i and $\forall j \in \{k+1, \dots, n\}$, p' in position j is equal to p_2 in position j. In this phase, depending on the definition, another element may be generated.

   In this way a new multiset P'' is obtained.

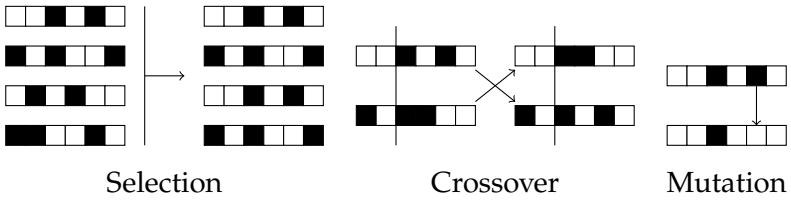


Figure 1: A graphical depiction of the Selection, Crossover and Mutation phases for GA.

4. **Mutation**. Each bit of each element of P'' is flipped according to a fixed (small) probability.

The algorithm stops when a termination criteria is met, for example when a certain number of iterations have been performed.

GA proved to be effective in many real world problems. However, to supplement the empirical results, a theoretical framework has been deemed necessary. While the similarity with the Darwinian theory of evolution can provide an idea on the behaviour of GA, this is not a formal proof that allow us to conclude that GA can always find one of the optima.

2.1.1 *A Brief Overview of GA Theory*

Here we provide an extremely brief overview of some approaches to the study of GA. We refer to the cited articles for a more complete and in-depth explanation.

2.1.1.1 *Schema Theory*

The first theory that was developed to understand the behaviour of GA was the *Schema Theory* [80]. While this theory, at least in his original form, was criticized, it still represent the first example of the necessity of a formal analysis of GA. We will recall the concept of schema and the schema theorem. We will then briefly explain why its applicability is limited and how schema theory has evolved.

A schema s is a subset of $\{0,1\}^n$ for which there exists a $A \subseteq \{1,\dots,n\}$ such that $\forall i \in A,\ \forall x,y \in \{0,1\}^n,\ x_i = y_i$. For example, the set $\{0000, 0001, 0100, 0101\}$ is a schema and it is usually denoted by $0*0* \in (\{0,1\} \cup \{*\})^n$, where this notation indicates that every individual whose first and third elements are 0 is part of the schema. Given $x \in \{0,1\}^n$ and a schema s, we say that s is represented in x iff $x \in s$. The order of a schema is the number of non-$*$ symbols in it and its order is the difference between the first and the last non-$*$ symbols in the schema. The fitness ratio of a schema s in a population P is the ratio between the average fitness of the individuals of $P \cup s$ and the average fitness of P.

The idea of the original schema theory was to study what schemata are represented more and more as the generations passes. The *schema theorem* states that given a standard GA with fitness-proportional selection, having mutation probability p_{mut} and crossover probability p_{cross}, a schema s of order $k(s)$, length $l(s)$ and fitness ratio $r(s,t)$ represented $N(s,t)$ times at generation t is such that:

$$E[N(s,t+1)] = \left(1 - p_{cross}\frac{l(s)}{n-1} - p_{mut}k(s)\right) r(s,t)N(s,t)$$

As it is possible to see, the schema theorem, in its original formulation, gives only an expectation on the number of individuals belonging to a certain schema after one generation. Thus, it cannot be used to give predictions at more than one step. The schema theorem was originally used to reach some conclusion that it cannot imply, at least in his original form. This use generated some criticism on the schema theory (see, for example, [67]). However, this theory survived and by the use of more refined techniques, it remained an important tool [161], not only of historical value.

2.1.1.2 *The Infinite Population Model*

One interesting approach was proposed by Vose [211], which studied GA as a discrete-time dynamical system over a continuous space. Consider a vector of positive reals of length 2^n such that all entries sum to 1, and call it a population vector. To every entry of the vector is associated an element of $\{0,1\}^n$ and the value of the entry represents the fraction of the individuals that are present in current population. While for finite populations this vector has only rational entries, when the population is assumed to be infinite, irrational entries are also possible. The study of GA with a population of infinite size simplifies the study of the dynamics since the system considered is a deterministic discrete-time dynamical system in a continuous space.

2.1.1.3 *GA as Markov Processes*

The modeling of GA by means of the Schema Theory, at least in its first form, had many shortcoming. An approach able to prove the convergence of GA was the modeling as Markov processes [174, 43, 151] (see [83] for an introduction to stochastic processes). In this case GA are modeled as a stochastic process. This process, being dependant only by the current population (at least for classical GA), can be entirely described by a transition matrix. By studying the properties of this matrix, it is possible to obtain properties of GA. For example, classical GA without elitism do not converge to a uniform population of optimal solutions while the use of elitism assures convergence.

2.1.1.4 *Runtime Analysis*

The aim of runtime analysis is to provide an expected upper bound on the runtime of many Evolutionary Algorithms. The goal is certainly more ambitious that providing convergence, and the first results were either about simple problems or simple algorithms. In the past years runtime analysis techniques become more and more sophisticated (for a survey see [153]).

Usually, runtime analysis studies $(\mu + \lambda) - EA$, that is, an evolutionary algorithm in which λ new individuals are created at every generation and the best μ between the offspring and the parents survive into the next generation (a variation is the $(\mu, \lambda) - EA$ in which the selection happens only between the offspring). The simplest algorithm is obtained when $\mu = \lambda = 1$. In this case the population has size 1, a new individual is created at every generation and substitutes the current one if it has a better fitness (the case of equal fitness generates two different variations of the algorithm with different properties).

One of the first results concerns $(1 + 1) - EA$ on the one-max problem with the classical GA mutation. In this case it was found that the expected time to reach the optimal solution is $O(n \log n)$, where n is the length of the individuals. Note that a $(1 + 1) - EA$ does not resemble commonly used EA. Thus, in the subsequent years, the effect of using a population was considered. And in some cases it was found that the optimization time decreases to polynomial from superpolynomial for a certain class of functions [98]. Current research on runtime analysis is focused both in providing results for more "realistic" EA and problems.

2.2 MODELING OF CROSSOVER

The number of contributions published so far aiming at modelling the GAs dynamics is so large that it is impossible to discuss all of them. The interested reader is referred, for instance, to the numerous papers of Vose and coworkers (for instance [211, 212, 173, 172, 22, 11, 10]).

The study of GAs crossover has been carried on in different ways so far. The traditional approach dates back to the early years of the field and it is based on the schema theory [93, 80]. Successively, more effective methods for investigating the dynamics induced by crossover have been defined by considering the transition matrix given by it and then studying the Markov process it induces (see for instance [211] and [165]).

A different approach is the one discussed in [185, 188, 213], where the topological space induced by crossover is modelled by structures such as hypergraphs and recombination spaces. Although related, here the perspective is different. In fact we aim at defining a distance between populations, which allows us to simplify the structures required for formalizing the model.

The work produced in the last few years by Moraglio and coworkers deserves a particular discussion, given that it is strongly related to the one reported here. In many of its references, among which for instance [141, 139, 140], Moraglio gives a geometrical interpretation of many kinds of crossover, including one-point crossover. This allows us to derive distance from operators in a conceptually simple way. One of the many contributions of the Moraglio's work stands, in our opinion, in the fact that it shades a light on the importance of studying topologies induced by genetic operators and defining operator-based distances to study evolutionary algorithms (EAs). The main difference in the work presented in this chapter is that we focus on the definition of a distance between populations. Also, the mathematical tools we use to model GAs (presented in Section 4.2) are different from, although related with, the ones used by Moraglio. At present, it is difficult to compare the effectiveness of our work to the Moraglio's one. However, we believe that an alternative approach to existing ones can be interesting for a large part of the EAs community, possibly opening discussions on pros and cons and/or stimulating researchers to investigate possible integrations.

3

PRETOPOLOGIES

3.1 INTRODUCTION

In this chapter the most important weak notions of topologies that have been described in the past are presented. Their importance is due to their use in field that are not necessarily linked with topology. In fact, these weak topologies were used to model aspects of some crossover operations for genetic algorithm [186] and in defining the approximation operators of rough set theory [30].

Let us recall that the standard approach to topology con be formalized by three different, but logically equivalent approaches which will be briefly discussed in this section.

3.1.1 *Topology by open or, equivalently, closed sets*

Let us start with the usual approach to topological spaces defined assigning to a set X, sometimes called the *universe* of points, a suitable collection of subsets as *opens* sets. Precisely, according to the standard Kelley textbook about topology [108] we have that:

Definition 3.1.1. A *topology of open sets* for X is a family $\mathcal{O}(X)$ of subsets of X, each member of which is called *open set*, satisfying the following conditions:

(O1) the empty set $\emptyset$ and the whole universe X belong to $\mathcal{O}(X)$;

(O2) the union of the members of any arbitrary family from $\mathcal{O}(X)$ belongs to $\mathcal{O}(X)$;

(O3) the intersection of the members of any finite family from $\mathcal{O}(X)$ belongs to $\mathcal{O}(X)$.

The pair $(X, \mathcal{O}(X))$ is a *topological spaces*. When no confusion seems possible we denote a topological space simply by X.

Let us recall that on the power set $\mathcal{P}(X)$ of the universe X, the *complementation map* $^c : \mathcal{P}(X) \mapsto \mathcal{P}(X)$ associating with any subset A its set theoretical complement $A^c := X \setminus A$ is an involutive $((A^c)^c = A)$ self–duality (i.e., anti-isomorphism with itself) [23, p. 3]. In particular

the involution condition implies that the complementation map is a bijection of $\mathcal{P}(X)$ onto itself, which is self-inverse, i.e., the inverse of $(\cdot)^c$ is just the same mapping: $[(\cdot)^c]^{-1} = (\cdot)^c$.

Let X be a topological space, then a subset of X is said to be *closed* iff its set theoretical complement is open. The collection of all closed set will be denoted by $\mathcal{C}(X)$ and so $C \in \mathcal{C}(X)$ iff $C^c \in \mathcal{O}(X)$. The following is a straightforward consequence of the de Morgan laws of the complementation map on $\mathcal{P}(X)$.

Proposition 3.1.1. *Let $\mathcal{O}(X)$ be a topology of open sets for X. Then the collection $\mathcal{C}(X)$ of corresponding closed sets satisfies the following conditions:*

(C1) the empty set $\emptyset$ and the whole universe X are closed, i.e., they belong to $\mathcal{C}(X)$;

(C2) the intersection of the members of any arbitrary family of closed sets is closed;

(C3) the union of the members of any finite family of closed sets is closed.

In the now outlined approach to topology open sets are taken as the primitive notion. An alternative, but logically equivalent, approach to topology consists in considering as primitive the notion of closed set by a family $\mathcal{C}(X)$ of subsets of X satisfying the condition (C1)–(C3) and then introducing as derived notion the one of open set as any subsets of X whose set theoretical complement is closed. The categories of topologies of open sets and of topologies of closed sets are isomorphic between them.

Proposition 3.1.2. *Let X be a given universe.*

1. *Let $\mathcal{O}(X)$ be a topology of open sets for X, then $\mathcal{O}(X)^{\blacktriangle} := \{C \in \mathcal{P}(X) : C^c \in \mathcal{O}(X)\}$ is a topology of closed sets for X.*
2. *Let $\mathcal{C}(X)$ be a topology of closed sets for X, then $\mathcal{C}(X)^{\blacktriangledown} := \{O \in \mathcal{P}(X) : O^c \in \mathcal{C}(X)\}$ is a topology of open sets for X.*
3. *Let $\mathcal{O}(X)$ be a topology of open sets for X, then $(\mathcal{O}(X)^{\blacktriangle})^{\blacktriangledown} = \mathcal{O}(X)$.*
4. *Let $\mathcal{C}(X)$ be a topology of closed sets for X, then $(\mathcal{C}(X)^{\blacktriangledown})^{\blacktriangle} = \mathcal{C}(X)$.*

3.1.2 *Topology by closure or, equivalently, interior operators*

The second approach to topology is the one introduced by Kuratowski, based on the primitive notion of closure operator on a *universe* of points X.

Definition 3.1.2. Let X be a set. A *Kuratowski closure* operator over X is a mapping $\overline{\cdot} : \mathcal{P}(X) \mapsto \mathcal{P}(X)$ with the following properties:

(KC1) $\overline{\emptyset} = \emptyset$.

(KC2) For all $A \in \mathcal{P}(X)$ $\quad A \subseteq \overline{A}$.

(KC3) For all $A, B \in \mathcal{P}(X)$ $\quad \overline{A} \cup \overline{B} = \overline{A \cup B}$.

(KC4) For all $A \in \mathcal{P}(X)$ $\quad \overline{\overline{A}} = \overline{A}$.

Remark. Since we will use in the sequel the weak notion of *Tarski closure* operator, we only mention here that this is a closure operator in which the additivity condition (KC3) is substituted by the weak condition of sub-additivity:

(TC3) For all $A, B \subseteq X$ $\quad \overline{A} \cup \overline{B} \subseteq \overline{A \cup B}$.

Since in general one has the condition $A \subseteq \overline{A}$ it is interesting to consider those subsets for which the equality holds.

Proposition 3.1.3. *Let X be a universe of points.*

1. *Let $\overline{\cdot} : \mathcal{P}(X) \mapsto \mathcal{P}(X)$ be a Kuratowski closure operator on the universe X. Then the collection $\mathcal{C}(X) := \{C \in \mathcal{P}(X) : C = \overline{C}\}$ is a topology of closed sets for X, i.e., the conditions (C1)–(C3) are satisfied.*

2. *On the other hand, let $\mathcal{C}(X)$ be a topology of closed sets for X. If for any subset A of X one defines the subset $\overline{A} := \bigcap\{C \in \mathcal{C}(X) : A \subseteq C\}$, then the mapping $A \in \mathcal{P}(X) \mapsto \overline{A} \in \mathcal{P}(X)$ is a Kuratowski closure operator, i.e., the conditions (KC1)–(KC4) are satisfied.*

3. *Starting from a Kuratowski closure operator, the just introduced topology of closed sets $\mathcal{C}(X)$ according to point a) is such that the associated Kuratowski closure operator according to point b) is the original one.*

The dual notion of Kuratowski closure is the notion of Kuratowski interior.

Proposition 3.1.4. *Let $(X, \overline{\cdot})$ be a universe equipped with a Kuratowski closure operator. Then the mapping $^{o} : \mathcal{P}(X) \mapsto \mathcal{P}(X)$ associating with any subset $A \in \mathcal{P}(X)$ the subset*

$$A^{o} := (\overline{A^{c}})^{c}$$

is a Kuratowski interior *operator in the sense that the following conditions are satisfied:*

(KI1) $X^{o} = X$.

(KI2) For all $A \in \mathcal{P}(X)$ $\quad A^{o} \subseteq A$.

(KI3) For all $A, B \in \mathcal{P}(X)$ $\quad (A \cap B)^{o} = A^{o} \cap B^{o}$.

(KI4) For all $A \in \mathcal{P}(X)$ $\quad (A^{o})^{o} = A^{o}$.

3.1.3 *Topology by neighbourhoods*

Another approach to define topology that is equivalent to the previous ones introduced is the use of neighbourhoods.

Definition 3.1.3. Let X be a set. It is a *neighbourhood space* iff for every $x \in X$ there exists a non-empty family $\mathcal{N}(x)$ of subsets of X that satisfies the following properties:

(P1) For all $N \in \mathcal{N}(x)$, $x \in N$.

(P2) For all $N_1, N_2 \subseteq X$, $N_1 \in \mathcal{N}(x)$ and $N_1 \subseteq N_2$ imply $N_2 \in \mathcal{N}(x)$.

(P3) For all $N_1, N_2 \in \mathcal{N}(x)$, $N_1 \cap N_2 \in \mathcal{N}(x)$

(P4) For all $N \in \mathcal{N}(x)$, there exists $M \in \mathcal{N}(x)$ with $M \subseteq N$ such that for all $y \in M$, $M \in \mathcal{N}(y)$.

A neighbourhood structure is related to the notion of topology by the following two definitions

Definition 3.1.4. Let X be a neighbourhood space. A set $Y \subseteq X$ is *open* iff for all $y \in Y$, $Y \in \mathcal{N}(y)$. That is, an open sets is a neighbourhoods of all its points.

Definition 3.1.5. Let X be a topological space. For all $x \in X$ and for all $Y \subseteq X$, Y is a neighbourhood of x iff there exists an open set Z such that $x \in Z \subseteq Y$.

Using the following two definitions it is possible to show that neighbourhoods are another way of define topologies.

3.2 FRÉCHET CLOSURE OPERATOR

A Fréchet closure operator on a universe X (see [74], and for a more complete treatment [168]) is the most general form of closure since it is based on only two very natural axioms expressing the intuitive, but minimal, requirements about closure: that the closure of the empty set must be the empty set itself and that the closure of a set contains the set.

Definition 3.2.1. Let X be a set. A *Fréchet closure* operator over X is a mapping $\overline{\cdot} : \mathcal{P}(X) \mapsto \mathcal{P}(X)$ with the following properties:

(Cl1) $\overline{\emptyset} = \emptyset$.

(Cl2) For all $A \subseteq X$, $A \subseteq \overline{A}$.

Example 3.2.1. Let $X = \{0, 1, 2\}$ and define $\overline{\cdot}$ on $\mathcal{P}(\{0, 1, 2\})$ as:

$$\begin{array}{lcllcl} \overline{\emptyset} & = & \emptyset & \overline{\{0,1,2\}} & = & \{0,1,2\} \\ \overline{\{0\}} & = & \{0,1,2\} & \overline{\{0,1\}} & = & \{0,1\} \\ \overline{\{1\}} & = & \{0,1,2\} & \overline{\{0,2\}} & = & \{0,2\} \\ \overline{\{2\}} & = & \{1,2\} & \overline{\{1,2\}} & = & \{0,1,2\} \end{array}$$

It is immediate that $\overline{\cdot}$ respects the two properties of Fréchet closure but neither idempotency nor subadditivity. In fact, $\overline{\{0\}} \cup \overline{\{1\}} = \{0,1,2\} \supset \{0,1\} = \overline{\{0,1\}}$ (i.e., no subadditivity) and $\overline{\{2\}} = \{1,2\} \neq \overline{\overline{\{2\}}} = \{0,1,2\}$ (i.e., no idempotency).

Definition 3.2.2. Let $(X, \overline{\cdot})$ be a universe equipped with a Fréchet closure operator. Then the mapping on the power set $^{o} : \mathcal{P}(X) \mapsto \mathcal{P}(X)$ associating with any subset A the subset $A^{o} := \left(\overline{A^{c}}\right)^{c}$, called the *interior* of A, defines a *Fréchet interior* operator which satisfies the following conditions:

(In1) $X^{o} = X$.

(In2) For all $A \subseteq X$, $A^{o} \subseteq A$.

By conditions (Il1) and (Cl1), for any subset A we obtain the following inclusion chain

$$A^{o} \subseteq A \subseteq \overline{A}$$

in such a way that A^{o} can be considered a *lower approximation* of A and $\overline{A}$ its *upper approximation*, producing in this way a *rough approximation* of the same subset defined by the pair of subsets: $r(A) := (A^{o}, \overline{A})$.

3.3 FRÉCHET (V)-SPACES

A slightly more structured closure operator is the one introduced by Fréchet in [74] and characterized by the additional property of subadditivity [177, 136].

Definition 3.3.1. Let X be a set. A *(V)-closure* operator over X is a mapping $\overline{\cdot} : \mathcal{P}(X) \mapsto \mathcal{P}(X)$ with the following properties:

(Cl1) $\overline{\emptyset} = \emptyset$.

(Cl2) For all $A \subseteq X \quad A \subseteq \overline{A}$.

(Cl3w) For all $A, B \subseteq X \quad \overline{A} \cup \overline{B} \subseteq \overline{A \cup B}$.

The following first result is very useful.

Proposition 3.3.1. *The condition (Cl3w) is equivalent to the isotonicity condition:*

$$A \subseteq B \quad \textit{implies} \quad \overline{A} \subseteq \overline{B} \tag{1}$$

Proof. Let the (CL3w) be true. Then from $A \subseteq B$, i.e., $B = A \cup B$, we get $\overline{B} = \overline{A \cup B} \supseteq \overline{A} \cup \overline{B} \supseteq \overline{A}$.

Vice versa, let the isotonicity condition be true. Since both the subsets $A, B \subseteq A \cup B$, then $\overline{A}, \overline{B} \subseteq \overline{A \cup B}$, and so $\overline{A} \cup \overline{B} \subseteq \overline{A \cup B}$. □

A (V)-space is the first kind of weak topology defined by a notion of neighborhood for each point of the universe.

Definition 3.3.2. Let X be a set. A *neighborhood structure* is a mapping $\mathcal{N} : X \mapsto \mathcal{P}(\mathcal{P}(X))$ assigning to any point x of X a family $\mathcal{N}(x) = \{V_\alpha(x)\} \subseteq \mathcal{P}(X)$ of subsets of X, called *neighborhoods* of x, satisfying the two conditions:

(N1) for all $x \in X$ the collection $\mathcal{N}(x)$ of subsets assigned to x must be non empty ($\mathcal{N}(x) \neq \emptyset$). In other words, every point has at least one neighborhood;

(N2) for any fixed point $x \in X$, each of its neighborhoods $V_\alpha \in \mathcal{N}(x)$ must contain x.

The set X equipped with the mapping $\mathcal{N}$ is said to be a *(V)-space* .

Now, it is possible to show the following:

Proposition 3.3.2. *Any (V)-space generates in a canonical way an operator of (V)–closure.*

1. First of all it is necessary to define the (V)–closure operator starting from the neighborhood structure of (V)-space.
   Let $' : \mathcal{P}(X) \mapsto \mathcal{P}(X)$ be the mapping defined for any subset A of X by the rule:

   $$A' = \{x \in X : \forall N \in \mathcal{N}(X)\,,\ (A \cap N) \setminus \{x\} \neq \emptyset\}$$

   A' is said to be the *derived* set of A and any of its points is called an *accumulation point* of A.
   Let $\mathrm{cl} : \mathcal{P}(X) \mapsto \mathcal{P}(X)$ be the mapping defined for any subset A of X by the rule:

   $$\mathrm{cl}(A) = A \cup A'$$

2. Now we need to prove that cl satisfies the properties of (V)-space closure.

   It is immediate that $\mathrm{cl}(\emptyset) = \emptyset$ and that $A \subseteq \mathrm{cl}(A)$ for all $A \subseteq X$. The only property that needs proving is sub-additivity.

$$\begin{aligned}
(A \cup B)' &= \{x \mid \forall N \in \mathcal{N}(x)\ \ (N \cap (A \cup B)) \setminus \{x\} \neq \emptyset\} \\
&= \{x \mid \forall N \in \mathcal{N}(x)\ \ (N \cap A) \setminus \{x\} \neq \emptyset \text{ or } (N \cap B) \setminus \{x\} \neq \emptyset\} \\
&\supseteq \{x \mid \forall N \in \mathcal{N}(x)\ \ (N \cap A) \setminus \{x\} \neq \emptyset\} \cup \\
&\quad \{x \mid \forall N \in \mathcal{N}(x)\ \ (N \cap B) \setminus \{x\} \neq \emptyset\} \\
&= A' \cup B'
\end{aligned}$$

   The result is then that $\mathrm{cl}(A) \cup \mathrm{cl}(B) = A \cup A' \cup B \cup B' = (A \cup B) \cup (A' \cup B') \subseteq (A \cup B) \cup (A \cup B)' = \mathrm{cl}(A \cup B)$ proving that the sub-additivity property holds.

Remark. The properties that the neighborhoods must all be non-empty (they must contain at least one subset of X) is necessary to obtain all

the properties of the closure. In fact, let $x \in X$ be a point such that $\mathcal{N}(x) = \emptyset$. Then $\mathrm{cl}(\emptyset) = \emptyset' = \{y : \forall N \in \mathcal{N}(y),\ \emptyset \cap N \neq \emptyset\}$ but since x has no neighborhood then $x \in \mathrm{cl}(\emptyset)$, contrary to the requirement i): $\mathrm{cl}(\emptyset) = \emptyset$.
This also means that every closure operator with less axioms than a (V)-space but still with $\overline{\emptyset} = \emptyset$ cannot also be expressed by a neighbourhood structure (since there is no weaker condition on the neighbourhood structure if non-emptiness is preserved). In particular, a Fréchet topology cannot be uniquely expressed with a neighbourhood structure.

Remark. Note that defining directly the closure as $\mathrm{cl}(A) = \{x \in X : \forall N \in \mathcal{N}(x),\ A \cap N \neq \emptyset\}$ does not work since under this definition it can be untrue that for all $N \in \mathcal{N}(x)$ the point x belongs to N, and in this case it is possible to have that $A \nsubseteq \mathrm{cl}(A)$ for some A.

Remark. Another interesting question is whether there is more than one neighborhood structure that generates the same (V)-closure operator.
The answer is positive. In fact, given a (V)-space it is easy to see that for every $x \in X$, when the collection $\mathcal{N}(x)$ of neighborhoods of x is considered then also $\mathcal{N}'(x) = \{N' : \exists N \in \mathcal{N}(x) \text{ s.t } N \subseteq N'\}$ is a collection of neighborhoods of x, both generating the same (V)-closure operator.

3.3.1 *The pre-topology induced from (V)-closure operators*

Also if the (V)-space closure operator is sufficiently weak, it is always possible to induce from it a structure of *pre-topological space* of closed elements, but without the further important condition of being closed with respect to the finite union, condition which characterizes the topology of closed sets.

Proposition 3.3.3. *Any (V)–closure operator determines the collection of "closed" elements defined as its "fixed" points:*

$$\mathcal{C}(X) := \left\{C \in \mathcal{P}(X) : \overline{C} = C\right\} \tag{2}$$

which satisfies the following properties:

(PC1) the empty set is closed: $\emptyset \in \mathcal{C}(X)$*;*

(PC2) the whole space is closed: $X \in \mathcal{C}(X)$*;*

(PC3) for any collection of closed elements $\{C_\alpha\} \subseteq \mathcal{C}(X)$ *their meet is closed too:* $\cap C_\alpha \in \mathcal{C}(X)$*. In other words,* $\mathcal{C}(X)$ *is a complete meet semi–lattice.*

Proof. (PC1) is nothing else than condition (Cl1) of definition 3.2.1. Moreover, from condition ii) of the same definition we have that in

particular $X \subseteq \overline{X}$, with $\overline{X}$ a subset of X and so necessarily $X = \overline{X}$, i.e., (PC2) holds.
Finally, for any arbitrary family of closed sets $\{C_\alpha)\} \subseteq \mathcal{C}(X$ we have that $\cap_\alpha C_\alpha \subseteq \{C_\alpha\}$, and from the isotonicity condition of proposition 3.3.1 it follows that $\overline{\cap_\alpha C_\alpha} \subseteq \{\overline{C_\alpha}\}$, and so $\overline{\cap_\alpha C_\alpha} \subseteq \cap_\alpha \overline{C_\alpha} = \cap_\alpha C_\alpha$, which making use of condition ii) of definition 3.3.1 leads to (PC3). □

Let us recall that, according to [138] (and see also [23, p. 111]), the only conditions (PC2) and (PC3) defines $\mathcal{C}(X)$ as a *Moore family*, satisfying the further (in general not required) condition of containing the least element o of the lattice (condition (PC1)). Or, following [36], the structure $\mathfrak{CS} := \langle X, \mathcal{C}(X)\rangle$ is a *closure system*.

Remark. Also in this case one can put the problem whether a family of closed sets is characterized by a unique (V)–closure operator.
This problem has a negative answer as the following Monteiro example from [137] shows.

Example 3.3.1. Let $X = \{0, 1, 2\}$ and let us consider the two (V)–closure operators.
Closure 1.

$$\begin{array}{rclrcl} \overline{\emptyset} & = & \emptyset & \overline{\{0,1,2\}} & = & \{0,1,2\} \\ \overline{\{0\}} & = & \{0,1\} & \overline{\{0,1\}} & = & \{0,1,2\} \\ \overline{\{1\}} & = & \{1\} & \overline{\{0,2\}} & = & \{0,1,2\} \\ \overline{\{2\}} & = & \{2\} & \overline{\{1,2\}} & = & \{0,1,2\} \end{array}$$

Closure 2.

$$\begin{array}{rclrcl} \overline{\emptyset} & = & \emptyset & \overline{\{0,1,2\}} & = & \{0,1,2\} \\ \overline{\{0\}} & = & \{0,2\} & \overline{\{0,1\}} & = & \{0,1,2\} \\ \overline{\{1\}} & = & \{1\} & \overline{\{0,2\}} & = & \{0,1,2\} \\ \overline{\{2\}} & = & \{2\} & \overline{\{1,2\}} & = & \{0,1,2\} \end{array}$$

These two (V)–closure operators are different since the closure of the singleton $\{0\}$ is $\{0, 1\}$ in the first case whereas it is $\{0, 2\}$ in the second one, but the two closure generate the same collection of closed sets.

Summarizing, we have the following implications:

$$\boxed{(V)-\text{space}} \Rightarrow \boxed{(V)-\text{closure operator}} \Rightarrow \boxed{\text{pre}-\text{Topology}}$$

but without any identification among the involved structures, in the sense that the symbol $\Rightarrow$ means a simple logical implication.

Let us remark that in the (V)–closure induced from (V)-spaces the only missed axiom with respect to the Tarski approach is idempotency, but this property can be recovered introducing two further closure operators on the basis of a (V)-closure satisfying the standard Tarski conditions.

3.3.2 *The Tarski closure operator induced from pre-topological spaces*

Let us investigate now as any pre-topological space induced in a canonical way a Tarski closure operator.

Proposition 3.3.4. *Let X be a universe.*
Then every pre-topology for X, i.e., every collection $\mathcal{C}(X)$ of subsets of X satisfying the conditions (PC1)–(PC3), determines a Tarski closure operator $^* : \mathcal{P}(X) \mapsto \mathcal{P}(X)$ defined for any subset A of X as follows:

$$cl_l(A) := \bigcap\{C \in \mathcal{C}(X) : A \subseteq C\}$$

Proof. The resulting operator cl_l is indeed a Tarski closure operator:

(TC1) $cl_l(\emptyset) = \emptyset$. It is immediate since $\emptyset = \overline{\emptyset}$ by condition (Cl1) defining a (V)-closure.

(TC2) $A \subseteq cl_l(A)$. It is immediate by the definition of cl_l.

(TC3) $cl_l\,(cl_l(A)) = cl_l(A)$. We have that for all $C \in \mathcal{C}$ such that $A \subseteq C$, $cl_l(A) \subseteq C$ by its definition. Then $cl_l\,(cl_l(A)) = cl_l(A)$.

(TC4) $cl_l(A) \cup cl_l(B) \subseteq cl_l(A \cup B)$. This property, when restated as $A \subseteq B \Rightarrow cl_l(A) \subseteq cl_l(B)$ is immediately true by the definition of cl_l.

The complete lattice abstract proof.

If $\mathcal{C}(\Sigma)$ is any family of elements from a complete lattice Σ, then for any element $a \in \Sigma$ let us consider the lattice meet $a^* := \wedge\{c \in \mathcal{C}(\Sigma) : a \leqslant c\}$, which exists owing to the completeness of the lattice Σ and the fact that the involved family is not empty since $a \leqslant 1$ with $1 \in \mathcal{C}(\Sigma)$ by (PC2).

(TC1) Trivially, $0^* = \wedge\{c \in \mathcal{C}(\Sigma) : 0 \leqslant c\}$ but from condition $0 \leqslant 0$ and the axiom (PC1), i.e., $0 \in \mathcal{C}(\Sigma)$, it follows that $0 \in \{c \in \mathcal{C}(\Sigma) : 0 \leqslant c\}$, which leads to the (TC1).

(TC2) Since a^* is the meet of the family $\mathcal{C}(a) := \{c \in \mathcal{C}(\Sigma) : a \leqslant c\}$ and in a complete lattice trivially a is the meet of the family $\Sigma(a) := \{b \in \Sigma : a \leqslant b\}$, from $\mathcal{C}(a) \subseteq \Sigma(a)$ it follows the (TC2).

(TC3) Applying the just proved (C2) to the element a^* we get $a^* \leqslant (a^*)^*$; moreover, from the particular case $a^* \leqslant a^*$ and the fact that owing to (PC3) $a^* \in \mathcal{C}(\Sigma)$, it follows that $(a^*)^* = \wedge\{d \in \mathcal{C}(\Sigma) : a^* \leqslant d\} \leqslant a^*$, i.e., the (TC3).

(TC4) Finally, if $a \leqslant b$ then $\{c \in \mathcal{C}(\Sigma) : b \leqslant c\} \subseteq \{d \in \mathcal{C}(\Sigma) : a \leqslant d\}$, which trivially leads to the monotonicity condition $a^* \leqslant b^*$ equivalent to (TC4). □

A closure operator, also if in the weakest form of the Fréchet closure considered in section 3.2 satisfies the minimal condition of increasing (the closure of a set always includes this latter). Also in the present Tarski l-case we have that for any possible subset A one has that

$A \subseteq cl_l(A)$. So it is interesting to single out the collection of all the possible subsets for which the equality holds:

$$\mathcal{C}_l(X) := \{H \in \mathcal{P}(X) : H = cl_l(H)\}$$

This is the collection of all Tarski l-closed subsets, and each elements of this set is called l-closed.

3.3.3 *The Tarski closure operator induced from a (V)-closure*

First of all, for a given subset A let us denote by $\overline{A}^n$, with $n \in \mathbb{N}$, the (V)-closure operator iterated n times on the set A:

$$\overline{A}^0 = A,\ \overline{A}^1 = \overline{A},\ \overline{A}^2 = \overline{\overline{A}}, \ldots, \overline{A}^{n+1} = \overline{(\overline{A}^n)}, \ldots$$

Let us note that the following ordering chain holds:

$$A \subseteq \overline{A}^1 \subseteq \overline{A}^2 \subseteq \ldots \subseteq \overline{A}^n \subseteq \overline{A}^{n+1} \subseteq \ldots \tag{3}$$

Moreover, from an iterated application of the isotonicity condition of the (V)-closure operators one obtains the following result:

$$A \subseteq B \quad \text{implies} \quad \overline{A}^n \subseteq \overline{B}^n \text{ for every } n \in \mathbb{N} \tag{4}$$

To continue it is necessary to recall the definition of *semi-continuity*.

Definition 3.3.3. Given two lattices P and Q a function $f : P \mapsto Q$ is *semi-continuous* iff for every subset U of P, if $\bigvee U$ exists then $f(\bigvee U) = \bigvee f(U)$.

From now on will be assumed the semi-continuity of the (V)-closure $\overline{\cdot}$ on every chain $A \subseteq \overline{A} \subseteq \overline{A}^2 \subseteq \ldots$. The case when the assumption is not fulfilled will be treated in Appendix 3.5.

We can define another closure operator cl_u as:

$$cl_u(A) = \bigcup_{n \in \mathbb{N}} \overline{A}^n \tag{5}$$

We are going to show that cl_u is a Tarski closure operator.

- $cl_u(\emptyset) = \emptyset$. Immediate since $\overline{\emptyset} = \emptyset$.
- $A \subseteq cl_u(A)$. It is immediate since $A \subseteq \overline{A} \subseteq cl_u(A)$.
- $cl_u(cl_u(A)) = cl_u(A)$. From the just proved property applied to the subset $cl_u(a)$ we get that $cl_u(A) \subseteq cl_u(cl_u(A))$. Now, applying the isotonicity of (4) to the inclusion $A \subseteq cl_u(A)$ one has that $\overline{A}^i \subseteq \overline{cl_u(A)}^i$ and so a fortiori $\bigcup_{i \in \mathbb{N}} \overline{A}^i \subseteq \bigcup_{i \in \mathbb{N}} \overline{cl_u(A)}^i$, i.e., $cl_u(A) \subseteq cl_u(cl_u(A))$.

To prove the vice versa it is only necessary to show that

$$\overline{\bigcup_{n\in\mathbb{N}} \overline{A}^n} \subseteq \bigcup_{n\in\mathbb{N}} \overline{A}^n$$

The inequality give us that $\mathrm{cl}(\mathrm{cl}(A)) \subseteq \mathrm{cl}(A)$. By semi-continuity we have that:

$$\overline{\bigcup_{n\in\mathbb{N}} \overline{A}^n} = \bigcup_{n\in\mathbb{N}} \overline{\overline{A}^n}$$

then it is immediate that:

$$\begin{aligned} \textstyle\bigcup_{n\in\mathbb{N}} \overline{\overline{A}^n} &= \textstyle\bigcup_{n\in\mathbb{N}} \overline{A}^{n+1} \\ &\subseteq A \cup \textstyle\bigcup_{n\in\mathbb{N}} \overline{\overline{A}^n} \\ &= \textstyle\bigcup_{n\in\mathbb{N}} \overline{A}^n \end{aligned}$$

This proves that $\overline{\mathrm{cl}(A)} \subseteq \mathrm{cl}(A)$ and, as a consequence, that $\mathrm{cl}(\mathrm{cl}(A)) \subseteq \mathrm{cl}(A)$.

Then for every A one has that $\mathrm{cl}(A) \supseteq \mathrm{cl}(\mathrm{cl}(A))$. Since $\mathrm{cl}(A) \subseteq \overline{\mathrm{cl}(A)} \subseteq \mathrm{cl}(\mathrm{cl}(A))$ we conclude that $\mathrm{cl}(A) = \mathrm{cl}(\mathrm{cl}(A))$.

$\mathrm{cl}_u(A) \cup \mathrm{cl}_u(B) \subseteq \mathrm{cl}_u(A \cup B)$. As shown in proposition 3.3.1, this property can be restated as the isotonicity (1): $A \subseteq B \Rightarrow \mathrm{cl}_u(A) \subseteq \mathrm{cl}_u(B)$. Since also the isotonicity expressed by (4) holds, $A \subseteq B$ implies $\overline{A}^i \subseteq \overline{B}^i$ for every $i \in \mathbb{N}$, as an immediate consequence the required property for cl_u holds.

It is interesting to note that if for any the proper subset A of X we have that $\overline{A} \supset A$, then cl_u gives the indiscrete topology.

Since the previous result holds only when the closure operator is semi-continuous it is interesting to find examples of closures that respect this property and closures that do not respect it.

Example 3.3.2. Consider the set $\mathbb{R}$ of real numbers where the closure of a set A is defined as the smallest interval $[a, b]$ such that $a, b \in \mathbb{N}$, $A \subseteq (a, b)$. It is easy to verify that this function is a (V)-closure. It is immediate that for every finite ascending chain $A \subseteq \overline{A} \subseteq \ldots$ the property of semi-continuity holds. Since the supremum of every chain is $\mathbb{R}$ when the starting set is non-empty, the property of semi-continuity holds for all ascending chain in the form $A \subseteq \overline{A} \ldots$. The Tarski closure that can be obtained with from this (V)-closure is the indiscrete topology.

Example 3.3.3. Consider the set $\mathbb{N}$ of natural numbers. Now, define the closure of a non-empty set A in the following way:

- $\overline{A} = A \cup \{z_{\text{even}}\}$ where z_{even} is the smallest even number not contained into A if such a number exists.

- $\overline{A} = A \cup \{z_{odd}\}$ where z_{odd} is the smallest odd number not contained into A if such a number exists and A contains all the even numbers.
- $\overline{A} = A$ otherwise.

It can be verified that $\overline{\cdot}$ is a (V)-closure. This closure does not respect the semi-continuity property. In fact, consider a set A that does not contain an infinity of even numbers and contains no odd numbers. Then $\bigcup_{n\in\mathbb{N}} \overline{A}^n$ is the set of all even numbers. It is immediate that that $\bigcup_{n\in\mathbb{N}} \overline{\overline{A}^n}$ is still the set of all even numbers but $\overline{\bigcup_{n\in\mathbb{N}} \overline{A}^n}$ is the set of all even numbers plus the number 1. This provides a counterexample for semi-continuity. In this case a Tarski closure cannot be recovered in the way previously illustrated.

There is an immediate relation between the three closure operators cl_u, cl_l and $\overline{\cdot}$: for every subset A of X one has that

$$A \subseteq cl_l(A) \subseteq \overline{A} \subseteq cl_u(A) \tag{6}$$

We can see this relation as a way to approximate a (V)-closure (which is not idempotent and so it is not Tarski) by using a pair of Tarski closures. Given a (V)-closure operator $\overline{\cdot}$, it is possible to give two Tarski closure pair (cl_l, cl_u) such that the above inequalities (6) holds; in particular cl_l is the *lower approximation closure* and cl_u the *upper approximation closure* of the original (V)-closure operator. Moreover, for all $i \in \mathbb{N}$ it is $\overline{A}^i \subseteq cl_u(A)$.

As usual for any Tarski closure operator one can single out the collection of all *closed sets*, as the fixed points with respect to the involved closure operator. In the context of the present closure operator cl_u this collection will be denoted by:

$$\mathcal{C}_u(X) := \{K \in \mathcal{P}(X) : cl_u(K) = K\}$$

Generally,

Proposition 3.3.5. *A subset A of X is a closed sets of cl_u iff $\overline{A} = A$. In other words, and recalling (2),*

$$\mathcal{C}(X) = \mathcal{C}_u(X) \subseteq \mathcal{C}_l(X)$$

Proof. For proving this statement, first of all let us consider the case when the first equality is not true. Then either $A \subset \overline{A}$ and $A = cl_u(A)$, which is impossible, or $A = \overline{A}$ and $A \subset cl_u(A)$ which is also impossible.

On the other hand, if A is u-closed, $A = cl_u(A)$, then by the above set theoretical inequalities (6) it is also $A = cl_l(A)$, i.e., $cl_u(A) \subseteq cl_l(X)$. □

3.4 ČECH TOPOLOGIES

A Čech topological space is a slightly more structured version of a (V)-space where the condition of subadditivity is substituted by a stronger one of equality[206, 186]. These spaces were called by Čech *closure spaces*.

Definition 3.4.1. Let X be a set. A *Čech closure* over X is a function $\overline{\cdot} : \mathcal{P}(X) \mapsto \mathcal{P}(X)$ with the following properties:

i. $\overline{\emptyset} = \emptyset$.

ii. For all $A \subseteq X \quad A \subseteq \overline{A}$.

iii. For all $A, B \subseteq X \quad \overline{A} \cup \overline{B} = \overline{A \cup B}$.

Associated to the Čech topology there is also its neighbourhood structure.

Definition 3.4.2. Consider a set X. Every Čech topology over X is defined by a neighbours structure $\mathcal{N}(x)$ that associates to every element $x \in X$ a non-empty set of subsets of X with the following properties:

P1. $x \in N$ for all $N \in \mathcal{N}(x)$.

P2. If $N, N' \in \mathcal{N}(x)$ then there is $N'' \in \mathcal{N}(x)$ such that $N'' \subseteq N \cap N'$.

P3. If $N \in \mathcal{N}(x)$ and $N \subseteq N'$ then $N' \in \mathcal{N}(x)$.

Recall that a base for a neighbouhood of x is a set $\mathcal{B}$ such that for all $V \in X$ such that $\exists B \in \mathcal{B}$ with $B \subseteq V$ then V is inside the neighbourhood of x. Recall that a subbase for the neighbourhood of x is a collection$\mathcal{S}$ of sets whose finite intersections form a base for the neighbourhood.

A structure $\mathcal{N}_B$ that satisfy only *P1* and *P2* is a base for the Čech topology and a structure $\mathcal{N}_S$ that satisfy only *P1* is a subbase. The neighbours structure is a stronger version of a filter. In fact, a set of non-empty sets that satisfy *P2* and *P3* is a filter.

The Čech closure operator can also be defined using the neighbourhood structure:

$$\overline{A} = \{x \in X \mid \forall N \in \mathcal{N}(x)\ N \cap A \neq \emptyset\}$$

It is immediate that the operator defined in this way is a Čech closure operator:

1. $\overline{\emptyset} = \{x \in X \mid \forall N \in \mathcal{N}(x)\ N \cap \emptyset \neq \emptyset\} = \emptyset$

2. Let $x \in A$. Then $x \in \overline{A}$ by property P1.

3. Let $A, B \subseteq X$, then:

$$\begin{aligned}\overline{A \cup B} &= \{x \in X \mid \forall N \in \mathcal{N}(x)\ N \cap (A \cup B) \neq \emptyset\} \\ &= \{x \in X \mid \forall N \in \mathcal{N}(x)\ (N \cap A) \cup (N \cap B) \neq \emptyset\} \\ &= \{x \in X \mid \forall N \in \mathcal{N}(x)\ N \cap A \neq \emptyset\} \cup \\ &\quad \{x \in X \mid \forall N \in \mathcal{N}(x)\ N \cap B \neq \emptyset\} \\ &= \overline{A} \cup \overline{B}\end{aligned}$$

In a way similar to the definition of the Čech closure operator, we can define a Čech interior operator:

$$(A)^\circ = \{x \in X \mid A \in \mathcal{N}(x)\}$$

3.4.1 *From Čech topologies to topologies*

Since from the Čech closure (resp. Čech interior) operator it is possible to obtain a closure (resp. interior) operator, every Čech closure (resp. Čech interior) operator induces a topology on X. Note that the induced closure operator does not always generate the starting Čech topology.

Let X be a space and $\overline{\cdot}$ a Čech closure over it. Then it is possible to obtain a topological closure operator cl by the following procedure. Let $\mathcal{C}$ be the set $\{C \subseteq X \mid \exists B \subseteq X\ \overline{B} = C\}$ (i.e., the set of pseudoclosed sets of X). Let $\mathcal{C}_{top} \subseteq \mathcal{C}$ be defined as $\{C \in \mathcal{C} \mid \overline{C} = C\}$. The closure operator cl is defined as:

$$cl(A) = \bigcap \{B \in \mathcal{C}_{top} \mid A \subseteq B\}$$

It is now necessary to check that cl respects all the properties of a topological closure operator:

- $cl(\emptyset) = \emptyset$. The empty set is in $\mathcal{C}$ and also in $\mathcal{C}_{top}$. This implies that $cl(\emptyset) = \emptyset$.
- $A \subseteq cl(A)$. It is immediate since all the sets in the intersection that defines cl are supersets of A.
- $cl(cl(A)) = cl(A)$. Since $cl(A)$ is the intersection of all the sets of $\mathcal{C}_{top}$ that are supersets of A, the same setsare also supersets of $cl(A)$. Since $A \subseteq cl(A)$, no other set can partecipate in the intersection. So, $cl(cl(A)) = cl(A)$.
- $cl(A) \cup cl(B) = cl(A \cup B)$. The implication for the $\subseteq$ part can be reformulated to $A \subseteq B \Rightarrow cl(A) \subseteq cl(B)$. This is immediate by the definition of cl. The implication in the $\supseteq$ part can be show in the following way. Let C, D be sets in $\mathcal{C}_{top}$. Then $\overline{C} \cup \overline{D} = \overline{C \cup D}$ and $\overline{\overline{C} \cup \overline{D}} = \overline{C} \cup \overline{D}$. Then $\overline{C \cup D} = \overline{C} \cup \overline{D} = \overline{\overline{C} \cup \overline{D}} = \overline{\overline{C} \cup \overline{D}} = \overline{C \cup D}$. So, if both C and D are members of $\mathcal{C}_{top}$ also

$C \cup D$ is member of $\mathcal{C}_{top}$. This means that $cl(A \cup B) = \bigcap\{C \in \mathcal{C}_{top} \mid A \cup B \subseteq C\} \subseteq \bigcap\{C \cup D \mid C, D \in \mathcal{C}_{top}\ A \subseteq C \text{ and } B \subseteq D\} \subseteq \bigcap\{cl(A) \cup D \mid D \in \mathcal{C}_{top}\ B \subseteq D\} = cl(A) \cup cl(B)$.

Since cl respects all the property of a topological closure, this means that it is possible to obtain a topological space starting from a Čech closure.

Example 3.4.1. Consider the Čech topology given by the following Čech closure operator:

$$\begin{array}{rclrcl} \overline{\emptyset} & = & \emptyset & \overline{\{0,1,2\}} & = & \{0,1,2\} \\ \overline{\{0\}} & = & \{0\} & \overline{\{0,1\}} & = & \{0,1,2\} \\ \overline{\{1\}} & = & \{1,2\} & \overline{\{0,2\}} & = & \{0,1,2\} \\ \overline{\{2\}} & = & \{1,2\} & \overline{\{1,2\}} & = & \{0,1,2\} \end{array}$$

Now consider the set $\mathcal{C}'$ of pseudoclosed sets where idempotency holds:

$$\mathcal{C}' = \{\emptyset, \{0\}, \{0,1,2\}\}$$

This set give a topology over $\{0,1,2\}$ but there is no way to recover the original Čech topology. In fact, the induced topology is also a Čech topology, meaning that there are at least two possible Čech topologies that induce this topology.

From the previous example it is immediate that there is a surjective mapping between Čech topologies and topologies but this mapping is not injective.

3.4.2 *Finite graph and Čech Topologies*

If X is finite there is an association between directed graphs and Čech topologies[186].

Every Čech topology over a finite set has an associated finite directed graph. Let $(X, \mathcal{N})$ be a Čech topological space. We can build the graph (X, E) where for every $x, y \in X$ the edge $(x, y) \in E$ if and only if $x \in \bigcap_{N \in \mathcal{N}(y)} N$, i.e., if x is in all the neighbourhoods of y.

Every finite directed graph has an associated Čech topology. Let (V, E) be a directed graph. For every vertex $v \in V$ let $out(v)$ be the set of outgoing edges of v. Define $\mathcal{N}(x) = \{A \subseteq V \mid out(x) \cup \{x\} \subseteq A\}$. The neighbourhood structure defined immediately respects P1, it respects P2 since $out(x) \cup \{x\}$ is a subset of every set in $\mathcal{N}(x)$ and also respects P3.

On finite graphs the notion of Čech closure and Čech interior can be easily defined. Given a graph $G = (V, E)$ and a set $A \subseteq V$, $\overline{A}$ is the set of all vertex $v \in V$ such that:

1. $v \in A$ *or*

2. There exists $v' \in A$ such that $(v, v') \in E$ (i.e., v is directly connected to a node in A).

On the other side the Čech interior of A (i.e., $(A)^\circ$) is defined as the set of all vertex $v \in A$ with the property that $\nexists v' \in V \setminus A$ such that $(v', v) \in E$ (i.e., all the vertex of A that does not have an ingoing edge from a vertex outside A). It is possible to show that this definition of Čech interior and Čech closure is the same as the neighbourhood based one.

Example 3.4.2. Consider the following Čech topology structure $\mathcal{N}_1$ over $X = \{0, 1, 2\}$:

$$\begin{aligned}
\mathcal{N}(0) &= \{\{0,1\},\{0,1,2\}\} \\
\mathcal{N}(1) &= \{\{1,2\},\{0,1,2\}\} \\
\mathcal{N}(2) &= \{\{2\},\{0,2\},\{1,2\},\{0,1,2\}\}
\end{aligned}$$

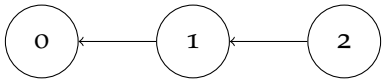


Figure 2: The graph induced by the Čech topology $\mathcal{N}_1$

In this structure consider the following Čech interiors:

$$\begin{aligned}
(\{0\})^\circ &= \emptyset & (\{0,1\})^\circ &= \{0\} \\
(\{1\})^\circ &= \emptyset & (\{0,2\})^\circ &= \{2\} \\
(\{2\})^\circ &= \{2\} & (\{1,2\})^\circ &= \{1,2\}
\end{aligned}$$

The fact that $\cdot^\circ$ is not idempotent is evident: $(\{0,1\})^\circ = \{0\}$ but $\left((\{0,1\})^\circ\right)^\circ = \emptyset$.

We can also consider the pseudoclosed sets:

$$\begin{aligned}
\overline{\{0\}} &= \{0\} & \overline{\{0,1\}} &= \{0,1\} \\
\overline{\{1\}} &= \{0,1\} & \overline{\{0,2\}} &= \{0,1,2\} \\
\overline{\{2\}} &= \{1,2\} & \overline{\{1,2\}} &= \{0,1,2\}
\end{aligned}$$

Also in this case it is evident that the Čech closure is not idempotent: $\overline{\{2\}} = \{1,2\}$ but $\overline{\overline{\{2\}}} = \{0,1,2\}$.

The closure operator induced by $\overline{\cdot}$ is the following:

$$\begin{aligned}
\mathrm{Cl}\{0\} &= \{0\} & \mathrm{Cl}\{0,1\} &= \{0,1\} \\
\mathrm{Cl}\{1\} &= \{0,1\} & \mathrm{Cl}\{0,2\} &= \{0,1,2\} \\
\mathrm{Cl}\{2\} &= \{0,1,2\} & \mathrm{Cl}\{1,2\} &= \{0,1,2\}
\end{aligned}$$

An other question is if $\overline{A \cap B}$ is $\overline{A} \cap \overline{B}$ for every set $A, B \subseteq X$. A short example show that it is not true.

Example 3.4.3. Consider the following Čech topology structure $\mathcal{N}_2$ over $X = \{0, 1, 2\}$:

$$\begin{aligned}\mathcal{N}(0) &= \{\{0\}, \{0,1\}, \{0,2\}, \{0,1,2\}\}\\ \mathcal{N}(1) &= \{\{0,1,2\}\}\\ \mathcal{N}(2) &= \{\{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}\}\end{aligned}$$

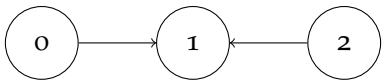


Figure 3: The graph induced by the Čech topology $\mathcal{N}_2$

Now consider $A = \{0\}$ and $B = \{2\}$. We immediately have that:

$$\begin{aligned}\overline{A} \cap \overline{B} &= \{0,1\} \cap \{1,2\} = \{1\}\\ \overline{A \cap B} &= \overline{\emptyset} = \emptyset\end{aligned}$$

In fact, we only have the condition that $\overline{A \cap B} \subseteq \overline{A} \cap \overline{B}$ from the fact that $C \subseteq D \implies \overline{C} \subseteq \overline{D}$

$$\begin{aligned}\overline{A \cap B} &\subseteq \overline{A}\\ \overline{A \cap B} &\subseteq \overline{B}\\ &\Downarrow\\ \overline{A \cap B} \cap \overline{B} &\subseteq \overline{A} \cap \overline{B}\\ \overline{A \cap B} &\subseteq \overline{A} \cap \overline{B}\end{aligned}$$

A similar condition holds for the Čech interior operator. Consider A and B. We are interested in checking if the property $(A)^\circ \cup (B)^\circ = (A \cup B)^\circ$ holds. Like in the previous case, the property is not true.

Example 3.4.4. Consider the neighbourhood structure $\mathcal{N}_3$ over $X = \{1, 2, 3\}$ defined as follows:

$$\begin{aligned}\mathcal{N}(0) &= \{\{0,1\}, \{0,1,2\}\}\\ \mathcal{N}(1) &= \{\{2\}, \{0,1\}, \{1,2\}, \{0,1,2\}\}\\ \mathcal{N}(2) &= \{\{1,2\}, \{0,1,2\}\}\end{aligned}$$

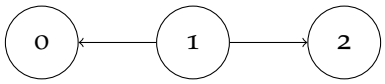


Figure 4: The graph induced by the Čech topology $\mathcal{N}_3$

Now consider $A = \{0\}$ and $B = \{1, 2\}$. We immediately have that:

$$\begin{aligned}(A)^\circ \cup (B)^\circ &= \emptyset \cup \{1,2\} = \{1,2\}\\ (A \cup B)^\circ &= (\{0,1,2\})^\circ = \{0,1,2\}\end{aligned}$$

In fact, the property that holds is that for all $A, B \subseteq X$, $(A)^\circ \cup (B)^\circ \subseteq (A \cup B)^\circ$. This follows directly from the fact that $A \subseteq B \Rightarrow (A)^\circ \subseteq (B)^\circ$:

$$\begin{array}{rcl} (A)^\circ & \subseteq & (A \cup B)^\circ \\ (B)^\circ & \subseteq & (A \cup B)^\circ \\ & \Downarrow & \\ (A)^\circ \cup (B)^\circ & \subseteq & (A \cup B)^\circ \cup (B)^\circ \\ (A)^\circ \cup (B)^\circ & \subseteq & (A \cup B)^\circ \end{array}$$

An other interesting property is the following: given two sets $A, B \subseteq X$, exists $C \subseteq X$ such that $\overline{C} = \overline{A} \cap \overline{B}$? The answer is, in the general case, negative, as it is possible to see in Example 3.4.2. The result of $\overline{\{1\}} \cap \overline{\{2\}}$ is $\{1\}$, which is not the Čech closure of any subset of $\{0, 1, 2\}$ since it is not the closure of any of its subsets.

A similar consideration is valid when considering the Čech interior operator and the following property: given two sets $A, B \subseteq X$, exists $C \subseteq X$ such that $(C)^\circ = (A)^\circ \cup (B)^\circ$? In the general case the answer is negative. This can be seen in Example 3.4.3. The result of $(\{0, 1\})^\circ \cup (\{1, 2\})^\circ$ is $\{0, 2\}$ which is not he Čech interior of any subset of $\{0, 1, 2\}$ since it is not the interior of itself or of any of it supersets.

3.4.3 *Čech topologies and convergence*

Let $(X, \mathcal{N})$ be a Čech topological space. Consider a filter F on X. The filter F converges to the point $x \in X$ if it contains the neighbourhood filter of x (i.e., $\mathcal{N}(x) \subseteq F$). The notation for this is $F \to x$.

The relation defined in this way denotes a *convergence space*. We recall that the relation $\to$ must satisfy the following properties:

1. *Centered*. The set of all the sets that contains x must converge to x:

$$\{A \mid x \in A\} \to x$$

2. *Isotone*. Let F, G be two filters with $F \subseteq G$. If $F \to x$ the $G \to x$.

3. *Directed*. Let F and G be two filters that converges to x. Then there exists $H \subseteq F \cap G$ such that $H \to x$. Note that from the scond property we also have that $F \cap G \to x$.

In fact in a Čech topological space with the given relation $\to$ the last properties holds in a stronger way. The relation is not only directed but also *infinitely directed* since the convergence to x is assured not only for finite but also for infinite intersection.

From the convergence structure we can recover the Čech topological space. A set N is a neighbourhood of x iff $N \in F$ for all $F \to x$. This means that

$$\mathcal{N}(x) = \bigcap \{F \mid F \to x\}$$

Note that every convergence space that is infinitely directed has a representation in terms of Čech topological spaces (the converse is also true).

The procedure to obtain the neighborhood of a point from a convergence space can be also carried out when the space is not infinitely directed. In this case the resulting Čech topology, when converted to a convergence space, has a weaker notion of convergence.

3.5 ITERATING THE CLOSURE OPERATOR A TRANSFINITE NUMBER OF TIMES

Definition 3.5.1. A set P with a linear order $<$ is *well-ordered* iff every non-empty subset of P has a least element.

The *well-ordering* principle states that every set can be well-ordered. This principle is unprovable in Zermelo-Fraenkel set theory (ZF) without the axiom of choice. In fact, the well-ordering principle is equivalent to the axiom of choice.

Definition 3.5.2. A set P is *transitive* iff $\forall x\ x \in P \Rightarrow x \subseteq P$. A set is an ordinal if it is transitive and well-ordered by $\in$.

The first ordinal is $0 = \emptyset$, the second ordinal is $1 = \{\emptyset\} = \{0\}$, the third ordinal is $2 = \{\emptyset, \{\emptyset\}\} = \{0, 1\}$. Over ordinals it is possible to define a successor operator: given an ordinal α, $\alpha + 1$ is $\alpha \cup \{\alpha\}$. An ordinal α is called a *successor ordinal* if $\alpha = \beta + 1$ for some ordinal β. An ordinal is called a *limit ordinal* if it is not 0 or the successor of some other ordinal. The ordinals obtainable starting from $\emptyset$ and iterating only the successor operator are called *finite ordinals*. Some properties of ordinal numbers are:

1. If α and β are ordinals, then either $\alpha \subseteq \beta$ or $\beta \subseteq \alpha$. In other words, two ordinals are always comparable.

2. Given a non-empty set of ordinals X, then $\cup X$ is an ordinal. The ordinal $\cup X$ is $\sup\{X\}$.

The first limit ordinal is denoted by ω and it is the set of all finite ordinals. It is possible to take is successor $\omega + 1$ and also repeat the limit taking process to obtain $\omega \cdot 2$, $\omega \cdot 3$, ω^2, etc.

As an example, consider the set $\mathbb{N}$ of natural numbers. Its *order type* with the usual ordering is ω. If we consider the same set but with a different well-ordering we can obtain different order types. For example with the ordering $<_1$, such that $1 <_1 2 <_1 3 \ldots <_1 0$ the order type obtained is $\omega + 1$. With an ordering where every even number is greater than every odd number the order type obtained is $\omega \cdot 2$. This examples show that different order types do not necessarily correspond to different cardinalities. The ordinals presented up to now are all countable. To obtain the first uncountable ordinal ω_1 we

consider the supremum of the set of countable ordinals. For every ordinal it is always possible to find a bigger ordinal by taking the successor or the supremum of all smaller ordinal. This means that **Ord**, the class of all ordinals, is not a set since otherwise it would be an ordinal and we could take its successor negating the fact the it contains all ordinals.

With the notion of ordinals it is possible to define the iteration of the closure operator for every $\lambda \in \mathbf{Ord}$. Given a set X equipped with a closure operator $\overline{\cdot}$ for all $A \subseteq X$, $\overline{A}^{\lambda}$ is defined as:

$$\overline{A}^{\lambda} = \begin{cases} A & \text{if } \lambda = 0 \\ \overline{\overline{A}^{\beta}} & \text{if } \exists \beta : \ \lambda = \beta + 1 \\ \bigcup_{\beta < \lambda} \overline{A}^{\beta} & \text{otherwise} \end{cases}$$

A first question that can arise is the real necessity of this definition. In fact, it is possible to produce some examples where the necessity of an uncountable number of iterations is needed. For example, consider a well ordering of the reals and the (V)-space closure operator given by $\overline{A} = \min\{\mathbb{R} \setminus A\}$ for all proper subsets of $\mathbb{R}$. The closure operator defined in this way is idempotent only for the empty set and for $\mathbb{R}$. Consider a countable set $A \subset \mathbb{R}$. Now consider a countable union $B = A \cup \overline{A} \cup \overline{A}^2 \cup \ldots$. Every set in the union is countable since the closure operator adds only one element at time. Since B is the countable union of countable sets it is itself countable. This means that $B \neq \mathbb{R}$ and it is not a fixed point for the iteration of the (V)-space closure. The previous example shows that to reach a fixed point it may be necessary to iterate more than a countable number of times.

Recall that for any cardinal κ, κ^+ denotes the next cardinal, also in the usual construction every cardinal κ is also an ordinal and, in particular, it is the first ordinal to have cardinality κ.

Proposition 3.5.1. *Let X be a set, with $|X| = \kappa$. Let $\overline{\cdot} : \mathcal{P}(X) \mapsto \mathcal{P}(X)$ be a function such that $A \subseteq \overline{A}$ for all $A \subseteq X$. Then*

$$cl(A) = \bigcup_{\lambda < \kappa^+} \overline{A}^{\lambda}$$

is such that that $cl(cl(A)) = cl(A)$.

Proof. Suppose otherwise. Then there exists a set A such that $\mathrm{cl}(A) \subset \mathrm{cl}(\mathrm{cl}(A))$. This means that it is possible to build the following family of sets:

$$A_0 = \overline{A} \setminus A, \ A_1 = \overline{A}^2 \setminus \overline{A}, \ldots, A_{\omega} = \overline{A}^{\omega+1} \setminus \overline{A}^{\omega}, \ldots$$

and, generally, $A_{\lambda} = \overline{A}^{\lambda+1} \setminus \overline{A}^{\lambda}$. Everyone of these sets is non-empty and disjoint from all the others. Consider two distinct sets A_{β} and A_{λ}. Withous loss of generality suppose $\beta < \lambda$. Then $A_{\lambda} = \overline{A}^{\lambda+1} \setminus \overline{A}^{\lambda} \subseteq$

$\overline{A}^{\lambda+1} \setminus \overline{A}^{\beta+1} \subseteq \overline{A}^{\lambda+1} \setminus \left(\overline{A}^{\beta+1} \setminus \overline{A}^{\beta}\right) = \overline{A}^{\lambda+1} \setminus A_\beta$. This means that for all $\beta, \lambda \in \mathbf{Ord}$ with $\beta \neq \lambda$ the set A_λ is disjoint from A_β.

Let B be a set defined as $\bigcup_{\lambda<\kappa^+} A_\lambda$. Since all the sets in the union are disjoint and non-empty:

$$|B| = \left| \bigcup_{\lambda<\kappa^+} A_\lambda \right| = \sum_{\lambda<\kappa^+} |A_\lambda| = \kappa^+ \sup_{\lambda<\kappa^+} \{|A_\lambda|\} \geqslant \kappa^+$$

where the first two equalities are by definition and the last equality can be found in **Lemma 5.8** of [99]. Also, since every element of the union is a subset of X, $B \subseteq X$ and $|B| \leqslant \kappa$. This means that the assumption that there exists a set such that $\mathrm{cl}(A) \subset \mathrm{cl}(\mathrm{cl}(A))$ is wrong. Then $\mathrm{cl}(A) \supseteq \mathrm{cl}(\mathrm{cl}(A))$. Since $\mathrm{cl}(A) \subseteq \overline{\mathrm{cl}(A)} \subseteq \mathrm{cl}(\mathrm{cl}(A))$ then $\mathrm{cl}(A) = \mathrm{cl}(\mathrm{cl}(A))$. □

Remark. The previous proposition assumes the axiom of choice since the existance of a unique successor cardinal and the inequality used in the proof require it.

The passage from a (V)-closure to a Tarski closure has been done using semi-continuity by defining the Tarski closure as:

$$\mathrm{cl}(A) = \bigcup_{n\in\mathbb{N}} \overline{A}^n$$

When the hypothesis of semi-continuity is dropped a Tarski closure can still be recovered in a similar way but the definition must allow iteration after ω. A working definition could be:

$$\mathrm{cl}(A) = \bigcup_{\lambda\in\mathbf{Ord}} \overline{A}^\lambda$$

The proof that cl is a Tarsky closure follow a pattern similar to the one used when semi-continuity was assumed. When using this definition from every (V)-space closure it is always possible to obtain a Tarski closure.

4

A THEORETICAL MODEL FOR ONE-POINT CROSSOVER

4.1 INTRODUCTION

Defining a distance based on the neighborhood structure induced by the genetic operators is a basic step for analysing various dynamics of the search process of Genetic Algorithms (GAs) [93, 80]. For instance, it is useful if we want to monitor population diversity (see for instance [80, 85, 26, 192, 59]) or if we want to calculate well-known indicators of problem hardness such as fitness distance correlation (see among others [102, 193]). Operator-based distance can make calculating distance and analysing the search process more accurate [102, 199]. Defining a distance, or a function to measure similarity, that is, in some sense "bound" to (or "consistent" with) the genetic operators informally means that if two objects (individuals or populations) are *close* to each other, or similar, one can be transformed into the other with a few applications of the operator(s). This has been recently formalized in [128] and we rely on that definition here.

We want to define a distance that is bound to standard one-point GAs crossover [93, 80]. While defining a mutation-based distance can be an easy task for GAs (for instance, Hamming distance is naturally bound to one-point mutation [102]), defining a crossover-based distance can be an issue, mainly because the neighborhood induced by crossover strongly depends on the population where individuals evolve (this difficulty has already been recognized in many references, for instance [102, 193]). Nevertheless, defining a crossover-based measure would be extremely important, given that crossover is often the main operator used by GAs to carry out search.

Here, we solve the problem of the influence of populations on neighborhoods by focusing on the definition of a crossover-based distance between populations. Once that function is defined, we show how it can be used to define a family of distances between individuals.

The definition of a crossover based distance that respects the dynamics of crossover could be used to obtain fitness-distance correlation (fdc) values that are more representative of the real problem difficulty. In fact, fdc value strongly depends on the distance chosen.

Hence, a distance that better models the GA dynamics could allow an assessment of the problem difficulty of GA that could be more reliable. Another use of a crossover based distance is the study of the mean distance between a population and all the other possible population. A population with a low mean distance can be a better initial population for GA because it can allow a better exploration of the search space. Furthermore, the study of a topology over populations can lead to a better understanding of the possible dynamics of GA.

The distance between populations that we introduce is (consistently with the definition given in [128]) strictly related to the minimum number of steps required to transform a population P_1 into another population P_2 by iteratively applying one-point crossover to randomly chosen individuals in P_1. It is important to note that here the proposed model diverges from the classical GA definition. In fact, we do not track the presence of multiple instances of the same individual (since it does not provide new genetic material) and we are not restricting the population size. However, the results obtained give a lower bound on the minimum number of steps required by a traditional GA.

Contrary to what one may imagine, we also show that calculating this distance can be computationally cheap and we present an algorithm that performs this calculation in polynomial time with respect to the population size and the number of genes composing the individuals.

This chapter is organized as follows. In Section 2.2 we revise previous and related work. In Section 4.2, some basic mathematical notions that we use to model GAs are recalled. In Section 4.3, the model used for computing the proposed distance is studied and some of its general properties are discussed. In Section 4.4, an alternative and more concise way of representing populations is introduced and an efficient algorithm to compute the proposed distance using this representation is given. Finally, Section 4.5 discusses and concludes the chapter.

4.2 BASIC NOTIONS

In this section some basic notions and some notations that are necessary for the continuation of this chapter are introduced.

We denote by $[i, j]$ with $i, j \in \mathbb{N}$ the set $\{i, i+1, \ldots, j-1, j\} \subseteq \mathbb{N}$. We denote by SC_n the set $\{[i, j] \mid 1 \leqslant i \leqslant j \leqslant n\}$ for a fixed $n \in \mathbb{N}$.

A finite alphabet will be denoted by Σ. The set of all the strings of a given length composed of symbols from Σ is denoted by Σ^n. A element $x \in \Sigma^n$ is denoted by $x_1, \ldots, x_n$. The notation $x_{[i,j]}$ is a shortcut for $x_i, x_{i+1}, \ldots, x_{j-1}, x_j$.

Recall that a *lattice* $\mathcal{L}$ is a non-empty set L endorsed with a partial ordering $<_L$ such that for any two elements $a, b \in L$ the *join* $a \vee b$ (i.e.,

the least upper bound of a and b) and the *meet* $a \wedge b$ (i.e., the greatest lower bound of a and b) operators are uniquely defined in$L$ [23].

A lattice is *bounded* if $\bigvee L$ (i.e., a maximal element for L) and $\bigwedge L$ (i.e., a minimal element for L) exist. A lattice is *complete* if for every subset S of L then both $\bigvee S$ and $\bigwedge S$ exist. Note that every finite lattice is complete.

Given a lattice $\mathcal{L} = (L, <_L)$ a subset O of L is a *lower set* if for all $x \in L$ such that there exists $y \in O$ with $x <_L y$ we have that $x \in O$. The set of all lower sets of a lattice $\mathcal{L}$ is denoted by $\mathcal{O}(\mathcal{L})$ and it is by itself a lattice with respect to set inclusion. See [23] for a reference on lattices.

4.3 CROSSOVER DISTANCE DEFINITION

4.3.1 *Crossover relations*

In this section we introduce the simplified model for GA with one-point crossover used to define the proposed distance. In this model, populations can be any subset of the set of strings of length n over an alphabet Σ. Hence, we are not considering fixed-size populations and we are not considering presence of multiple copies of the same individual in the population.

Definition 4.3.1. A one-point crossover relation R_I is a binary relation over $\Sigma^n \times \Sigma^n$ such that for all $x, y, x', y' \in \Sigma^n$:

$$\begin{aligned}(x,y)R_I(x',y') \quad \Leftrightarrow \quad & \exists k \in [0,n] \text{ s.t. } x' = x_{[1,k]}y_{[k+1,n]} \\ & \text{and } y' = y_{[1,k]}x_{[k+1,n]}\end{aligned}$$

In other words, two pairs of elements are in one-point crossover relation when the second pair can be obtained by one-point crossover from the first pair. The relation R_I is reflexive, symmetric but not transitive. It is immediate that its transitive closure is an equivalence relation that partitions $\Sigma^n \times \Sigma^n$ into $\left(\binom{|\Sigma|}{2} + |\Sigma|\right)^n$ equivalence classes (i.e., for every position it is possible to choose a pair of symbols that are not necessarily distinct).

Definition 4.3.2. A one-point crossover relation R_P over $\mathbf{P} = \mathcal{P}(\Sigma^n)$ is a relation such that for all $P_1, P_2 \in \mathbf{P}$:

$$\begin{aligned}P_1 R_P P_2 \quad \Leftrightarrow \quad & \forall x' \in P_2 \; \exists y' \in \Sigma^n \; \exists x, y \in P_1 \\ & \text{s.t. } (x,y)R_I(x',y')\end{aligned}$$

In other words, two subsets of Σ^n are in relation if and only if every element of the second subset can be obtained by crossover from elements of the first subset. Notice that there are some assumptions in this model of crossover. First of all not all elements of the first population need to contribute to obtain the second population. Furthermore,

only one of the offspring need to be inserted into the resulting population. Finally, the current model does not take into account the size of the population that can be obtained (e.g., $\Sigma^n R_P \emptyset$).

Example 4.3.1. Consider the following two populations:

$$\begin{aligned} P_1 &= \{0110, 0101, 0011\} \\ P_2 &= \{0111, 0001, 0000\} \end{aligned}$$

We have that $P_2 R_P P_1$ since all of the element of P_1 can be obtained with one application of one-point crossover to pair of elements of P_2. In fact, 0110 can be obtained from 0111 and 0000, 0101 can be obtained from 0111 and 0001, while 0011 can be obtained from 0000 and 0111. It is not true that $P_1 R_P P_2$ since 0000 cannot be obtained from one application of one-point crossover from any pair of elements of P_1.

The relation R_P is reflexive. The Example 4.3.1 shows that R_P is not a symmetric relation. Neither the relation is transitive. Note that, by definition, if $P_1 R_P P_2$ then for all $P_2' \subseteq P_2$ and for all $P_1' \supseteq P_1$ it is true that $P_1' R_P P_2'$.

The main idea that will be carried on is to define a Čech closure $\overline{\cdot}$ such that for any population P we have that $\overline{\{P\}}^i$ is the set of populations that can be obtained after i generations using only crossover as genetic operator. In this way it is possible to define a closure $[\![\cdot]\!]$ such that for any two populations P_1 and P_2:

- P_2 can be obtained using only crossover from P_1 iff $P_2 \in [\![\{P_1\}]\!]$.
- The minimal $k \in \mathbb{N}$ such that $P_2 \in \bigcup_{i=0}^{k} \overline{\{P_1\}}^i$ is the minimum number of generations needed to obtain P_2 from P_1.

Such a closure can be defined in the following way:

Definition 4.3.3. The *crossover closure* is a function $\overline{\cdot} : \mathcal{P}(\mathbf{P}) \mapsto \mathcal{P}(\mathbf{P})$ defined, for every $A \subseteq \mathbf{P}$, as :

1. When $A = \emptyset$, $\overline{A} = \emptyset$.
2. When $A = \{P\}$, $\overline{\{P\}} = \{P' \in \mathbf{P} \mid P R_P P'\}$.
3. Otherwise, $\overline{\{P_1, P_2, \ldots, P_k\}} = \bigcup_{i=1}^{k} \overline{\{P_i\}}$.

The first property we need to prove is that $\overline{\cdot}$ is a Čech closure. In this way we will be able to use the properties of Čech closures when needed.

Proposition 4.3.1. *The crossover closure is a Čech closure.*

Proof. The closure of $\emptyset$ is $\emptyset$ by definition. Since the relation R_P is reflexive, for all $A \subseteq \mathbf{P}$ it is immediate that $A \subseteq \overline{A}$. The property of additivity holds by the definition of crossover closure. □

Furthermore, for all $P_1, P_2 \in \mathbf{P}$, $P_2 \in \overline{\{P_1\}}^k$ iff there exist

$$P_1 = Q_0, Q_1, \ldots, Q_{k-1}, Q_k = P_2 \in \mathbf{P}$$

such that $Q_i R_P Q_{i+1}$ for all $i \in [0, k-1]$. In other words, we are requiring that one application the closure $\overline{\cdot}$ effectively represents one generation.

Proposition 4.3.2. *For all* $P_1, P_2 \in \mathbf{P}$ *and for all* $k \in \mathbb{N}$, $P_2 \in \overline{\{P_1\}}^k$ *(*Property 1*) iff there exists* $Q_0, \ldots, Q_k \in \mathbf{P}$ *with* $Q_0 = P_1$, $Q_k = P_2$ *and such that* $Q_i R_P Q_{i+1}$ *for* $i \in [0, k-1]$ *(*Property 2*).*

Proof. Suppose that *Property 1* holds. We prove by induction on k that *Property 2* follows. When $k = 0$ it is immediate that for any $P_1, P_2 \in \mathbf{P}$ $P_2 \in \{P_1\}$ iff $P_1 = P_2$ which immediately gives $P_1 R_P P_1$ by reflexivity of R_P.

Suppose that *Property 1* implies *Property 2* up to k. We prove that the implication is also true for $k+1$. Take any $P_1, P_2 \in \mathbf{P}$ such that $P_2 \in \overline{\{P_1\}}^{k+1}$. There exists a population $P' \in \overline{\{P_1\}}^k$ such that $P' R_P P_2$ by definition of $\overline{\cdot}$. By induction hypothesis there exists $P_1 = Q_0, Q_1, \ldots, Q_k = P'$ such that $Q_i R_P Q_{i+1}$ for all $i \in [0, k-1]$. The sequence $P_1 = Q_0, \ldots, Q_k = P', Q_{k+1} = P_2$ is such that $Q_i R_P Q_{i+1}$ for all $i \in [0, k]$ proving that *Property 2* holds for $k+1$.

Now assume *Property 2*. We prove that it implies *Property 1*. When $k = 0$ the statement is vacuously true.

Suppose that *Property 2* implies *Property 1* up to k. We prove that the implication also holds for $k+1$. Take any $P_1, P_2 \in \mathbf{P}$ such that there exists $P_1 = Q_0, Q_1, \ldots, Q_k, Q_{k+1} = P_2$ such that *Property 2* holds. By induction hypothesis $Q_k \in \overline{\{P_1\}}^k$. By definition of $\overline{\cdot}$ we have that $P_2 \in \overline{\{P_1\}}^{k+1}$. Thus, *Property 1* holds for $k+1$. □

Proposition 4.3.2 shows that iterating the closure of a set of populations k times is equivalent to collecting all the populations reachable from the considered set in at most k generations using one-point crossover.

4.3.2 *The Structure of the Closure*

We study the structure of $\overline{\{P\}}$ for all $P \in \mathbf{P}$. In many cases a chain of sets of populations $\{P\} \subseteq \overline{\{P\}} \subseteq \overline{\{P\}}^2 \subseteq \ldots$ could be substituted by a chain of populations $P \subseteq P' \subseteq P'' \subseteq \ldots$ that is more easily tractable.

Definition 4.3.4. Let $\mu_P : \mathcal{P}(\mathbf{P}) \mapsto \mathbf{P}$ be defined as:

$$\mu_P(\{P_1, \ldots, P_k\}) = \bigcup_{i=1}^{k} P_i$$

Proposition 4.3.3. *For all* $P \in \mathbf{P}$, $\overline{\{P\}}$ *is the lattice* $\left(\mathcal{P}\left(\mu_P\left(\overline{\{P\}}\right)\right), \subseteq\right)$.

Proof. It is immediate that for all $P' \in \overline{\{P\}} \implies P' \subseteq \mu_P\left(\overline{\{P\}}\right)$. We only need to prove the inverse implication. Note that $\{x\} \in \overline{\{P\}} \Leftrightarrow \{x\} \subseteq \mu_P\left(\overline{\{P\}}\right)$ and that $\emptyset \in \overline{\{P\}}$. Then we only need to prove that $\overline{\{P\}}$ is closed under finite union. Let $P_1, P_2 \in \overline{\{P\}}$ then for all $x' \in P_1 \cup P_2$ there exists $y' \in \Sigma^n$ and $x, y \in P$ such that $(x, y)R_I(x', y')$ since either $x' \in P_1$ or $x' \in P_2$. This means that $P_1 \cup P_2 \in \overline{\{P\}}$.

Since $\overline{\{P\}}$ is the power set of $\mu_P\left(\overline{\{P\}}\right)$ it is a lattice with respect to the set inclusion ordering with union and intersection as join and meet operations. □

Note that $\overline{\{P\}}$ has $\mu_P\left(\overline{\{P\}}\right)$ as maximal element and $\emptyset$ as minimal element.

Proposition 4.3.4. *For all* $P \in \mathbf{P}$ *and for all* $i \in \mathbb{N}$, $\overline{\{P\}}^i$ *is a lattice.*

Proof. $\{P\}$ is a lattice. $\overline{\{P\}}$ is a lattice by Proposition 4.3.3. Suppose that $\overline{\{P\}}^i$ is a lattice. We prove that $\overline{\{P\}}^{i+1}$ is also a lattice. Due to the additivity property of Čech closures we have that $A \subseteq B \implies \overline{A} \subseteq \overline{B}$. Also, since for all $P' \in \overline{\{P\}}^i$, $P' \subseteq \mu_P\left(\overline{\{P\}}^i\right)$ we have that

$$\overline{\{P\}}^{i+1} = \bigcup_{P' \in \overline{\{P\}}^i} \overline{\{P'\}} \subseteq \overline{\mu_P\left(\overline{\{P\}}^i\right)}$$

The other direction of the inclusion is immediate because $\mu_P\left(\overline{\{P\}}^i\right) \in \overline{\{P\}}^i$. Since $\mu_P\left(\overline{\{P\}}^i\right) \in \mathbf{P}$ we have that the closure of $\mu_P\left(\overline{\{P\}}^i\right)$ is a lattice. □

As a direct corollary given by the proof of the previous proposition, we have that:

Corollary. *For all* $P \in \mathbf{P}$ *and for all* $i \in \mathbb{N}$ *with* $i > 0$, $\overline{\{P\}}^i$ *is the lattice of* $\mathcal{P}\left(\mu_P\left(\overline{\{P\}}^i\right)\right)$ *ordered by set inclusion.*

To a sequence $\{P\} \subseteq \overline{\{P\}} \subseteq \overline{\{P\}}^2 \subseteq \ldots$ we can associate the sequence $\mu_P(\{P\}) \subseteq \mu_P\left(\overline{\{P\}}\right) \subseteq \mu_P\left(\overline{\{P\}}^2\right) \subseteq \ldots$ that is also equivalent to the sequence $S_0(P) \subseteq S_1(P) \subseteq S_2(P) \subseteq \ldots$ where $S_0(P) = P$ and $S_i(P) = \mu_P\left(\overline{\{S_{i-1}\}}\right)$ when $i > 0$. This means that, in order to know if a population P' exists inside $\overline{\{P\}}^i$, all we have to do is to determine if P' is a subset of $S_i(P)$. Also, to know if there exists a population P' inside $\overline{\{P\}}^i$ such that a certain element x is in P', all we have to do is seeking if $x \in S_i$. Now, we study the properties that hold in both representations.

Definition 4.3.5. Let $P \in \mathbf{P}$. Then for all $i \in \mathbb{N}$ the set $S_i(P) \in \mathbf{P}$ is defined as:

$$S_i(P) = \begin{cases} P & \text{if } i = 0 \\ \mu_P\left(\overline{\{S_{i-1}(P)\}}\right) & \text{otherwise} \end{cases}$$

The function $next : \mathbf{P} \mapsto \mathbf{P}$ is defined as $next(P) = S_1(P)$.

The first property that can be reported from a representation to the other is the presence of a population in a closure. In this case the sentence $P_2 \in \overline{\{P_1\}}^i$ simply becomes $P_2 \subseteq S_i(P_1)$. First of all we need the following proposition linking $S_i(P)$ with the Čech closure.

Proposition 4.3.5. *For all* $P \in \mathbf{P}$ *and for all* $i \in \mathbb{N}$ *with* $i > 1$, $\overline{\{P\}}^i = \overline{\{S_{i-1}(P)\}}$.

Proof. By induction on i, consider $i = 1$, then $\overline{\{P\}} = \overline{\{S_0(P)\}}$ by definition of $S_0(P)$. Consider the equivalence true for i, we are going to prove it for $i+1$. $\overline{\{P\}}^{i+1} = \overline{\overline{\{P\}}^i} = \overline{\overline{\{S_{i-1}(P)\}}}$. By the proof of Proposition 4.3.4 we have that $\overline{\overline{\{S_{i-1}(P)\}}} = \overline{\mu_P\left(\overline{\{S_{i-1}(P)\}}\right)}$ that, by definition of $S_i(P)$, is equal to $\overline{S_i(P)}$. □

The desired corollary is then the following one.

Corollary. *For all* $P_1, P_2 \in \mathbf{P}$ *and for all* $i \in \mathbb{N}$ *with* $i > 0$, $P_2 \in \overline{\{P_1\}}^i$ *iff* $P_2 \subseteq S_i(P_1)$.

Proof. $P_2 \in \overline{\{P_1\}}^i$ is equivalent to $P_2 \in \overline{\{S_{i-1}(P_1)\}}$. Since for $i > 0$, $\overline{\{P\}}^i$ contains all the subsets of $\mu_P\left(\overline{\{P\}}^i\right)$, $P_2 \in \overline{\{S_{i-1}(P_1)\}}$ is equivalent to $P_2 \subseteq \mu_P\left(\overline{\{S_{i-1}(P_1)\}}\right) = S_i(P_1)$. □

Proposition 4.3.6. *Let* $P_1, P_2 \in \mathbf{P}$ *such that there exists* $i \in \mathbb{N}$ *with* $P_2 \subseteq S_i(P_1)$. *Then the following holds:*

$$\min\{i \in \mathbb{N} \mid P_2 \subseteq S_i(P_1)\} = \max\{\min\{i \in \mathbb{N} \mid \{x\} \subseteq S_i(P_1)\} \mid x \in P_2\}$$

Proof. Let the two sides of the equation be called ℓ_1 and ℓ_2 respectively and suppose ℓ_1 and ℓ_2 are different from 0 (the proposition is immediately proved otherwise). Suppose $\ell_1 < \ell_2$. This is impossible since when $P_2 \subseteq S_{\ell_1}(P_1)$ we have also that $\{x\} \subseteq S_{\ell_1}(P_1)$ for all $x \in P_2$. Hence, $\ell_1 \geqslant \ell_2$. Suppose $\ell_1 > \ell_2$. This is also impossible since we have that $\{x\} \subseteq S_{\ell_2}(P_1)$ for all $x \in P_2$. By Corollary 4.3.2 this means that $\{x\} \in \overline{\{P_1\}}^{\ell_2}$ for all $x \in P_2$. By the lattice structure of the closure of $\overline{\{P_1\}}^{\ell_2}$ we also have that $\bigcup_{x \in P_2}\{x\} = P_2 \in \overline{\{P_1\}}^{\ell_2}$. This means that $P_2 \in S_{\ell_2}(P_1)$. Hence $\ell_1 = \ell_2$. □

Now we prove that computing these minimal values on the sets $\overline{\{P\}}^i$ is similar to computing them on the sets $S_i(P)$, hence we can use the latter sets to obtain information on the former ones.

Proposition 4.3.7. *For all* $P_1, P_2 \in \mathbf{P}$ *such that there exists* $i \in \mathbb{N}$ *with* $P_2 \in \overline{\{P_1\}}^i$ *the following holds:*

$$\min\{i \in \mathbb{N} \mid P_2 \in \overline{\{P_1\}}^i\} = \begin{cases} 1 & \text{if } P_2 \subset P_1 \\ 0 & \text{if } P_1 = P_2 \\ \min\{i \in \mathbb{N} \mid P_2 \subseteq S_i(P_1)\} & \text{otherwise} \end{cases}$$

Proof. In the first case we have that $P_2 \subset P_1$ implies that the value of i must be at least 1. It is 1 because of the fact that the closure $\overline{\{P_1\}}$ is a lattice of subsets that contains P_1 and, consequently, P_2. The second case is immediate since $P_1 \in \{P_1\}$. The third case is provided by Corollary 4.3.2 if we assume that the minimal i is not 0. Since $P_2 \subseteq S_0(P_1)$ is equivalent to $P_2 \subseteq P_1$ we have that this conditions are covered by the other two cases. □

4.3.3 *Distance Definition*

First of all, we define a quasi-metric that will be successively used to define a metric over $\mathbf{P}$.

Recall that a quasi-metric is a function d such that:

1. For all x, y, $d(x, y) \geqslant 0$ and $d(x, y) = 0 \Leftrightarrow x = y$.
2. For all x, y, z, $d(x, y) \leqslant d(x, z) + d(y, z)$.

Note that iterating the closure operator $\overline{\cdot}$, we always reach a fixed point after a finite number of steps (i.e., there exists $k \in \mathbb{N}$ such that $\overline{\cdot}^k$ and $\overline{\cdot}^{k+1}$ are the same function), since the Čech closure in monotone and the domain $\mathbf{P}$ is finite. In fact, the fixed point is necessarily reached after at most $|\mathbf{P}|$ iterations.

Let k^* be the integer $\min\{k \in \mathbb{N} \mid \forall U \subseteq \mathbf{P} \ : \ \overline{U}^k = \overline{U}^{k+1}\}$ (i.e., the first $k \in \mathbb{N}$ that allows any possible iteration of $\overline{\cdot}$ to reach a fixed point). Note that for any two populations P_1, P_2 if $P_2 \notin \overline{\{P_1\}}^{k^*}$ then P_2 is not reachable by crossover from P_1.

Definition 4.3.6. Let $f_P : \mathbf{P} \times \mathbf{P} \mapsto \mathbb{R}_+$ be defined as:

$$f_P(P_1, P_2) = \begin{cases} \min\{k \in \mathbb{N} \mid P_2 \in \overline{\{P_1\}}^k\} & \text{if } P_2 \in \overline{\{P_1\}}^{k^*} \\ k^* + 1 & \text{otherwise} \end{cases}$$

Proposition 4.3.8. *The function* f_P *is a quasi-metric.*

Proof. It is immediate that f_P is always not negative. Also, $f_P(P_1, P_2) = 0$ iff $P_2 \in \{P_1\}$ (i.e., iff $P_1 = P_2$).

For the sake of argument, suppose that the triangle inequality does not hold. Then there exist $P_1, P_2, P' \in \mathbf{P}$ such that $f_P(P_1, P') + f_P(P', P_2) < f_P(P_1, P_2)$. Without loss of generality suppose all considered values of f_P less than $k^* + 1$. By Proposition 4.3.2 there exist $P_1 = Q_0, \ldots, Q_{f_P(P_1,P')} = P'$ and $P' = S_0, \ldots, S_{f_P(P',P_2)} = P_2$ such that $Q_i R_P Q_{i+1}$ for all $i \in [0, f_P(P_1, P') - 1]$ and $S_i R_P S_{i+1}$ for all $i \in [0, f_P(P', P_2) - 1]$. It is possible to concatenate the two sequences to obtain $P_1 = Q_0, \ldots, Q_{f_P(P_1,P')} = P' = S_0, \ldots, S_{f_P(P',P_2)}$. Also by Proposition 4.3.2 we have that $P_2 \in \overline{\{P_1\}}^k$ with $k = f_P(P_1, P') + f_P(P', P_2)$. By the definition of f_P we have that $f_P(P_1, P_2) \leqslant f_P(P_1, P') + f_P(P', P_2)$, contradicting the initial assumption of triangle inequality to be false. □

From a quasi-metric it is immediate to define a metric by summing to f_P itself with swapped arguments in order to obtain symmetry.

Definition 4.3.7. Let $d_P : \mathbf{P} \times \mathbf{P} \mapsto \mathbb{R}_+$ be defined as:

$$d_P(P_1, P_2) = \frac{1}{2}\left(f_P(P_1, P_2) + f_P(P_2, P_1)\right)$$

Note that for every population it is possible to define a distance between individuals.

Definition 4.3.8. Let $P \in \mathbf{P}$. Then the function $d_I^P : \Sigma^n \times \Sigma^n \mapsto \mathbb{R}_+$ is defined as:

$$d_I^P(x, y) = d_P\left((P \setminus \{x\}) \cup \{y\}, (P \setminus \{y\}) \cup \{x\}\right)$$

Proposition 4.3.9. *For all $P \in \mathbf{P}$, the function d_I^P is a distance.*

Proof. Both symmetry and the triangle inequality are inherited from the fact that d_P is a distance. The only property that need proving is that for all $x, y \in \Sigma^n$, $x = y \Leftrightarrow d_I^P(x, y) = 0$. This is immediate since $(P \setminus \{x\}) \cup \{y\}$ can be equal to $(P \setminus \{y\}) \cup \{x\}$ only when $x = y$. □

Note that all the steps from Definition 4.3.3 are not dependent on the explicit definition of the crossover relation. In fact, all the definitions and propositions remain valid also for any other relation. Their extension to other kinds of crossover is then immediate.

Also, note that it is possible to use Corollary 4.3.7 and Proposition 4.3.6 to decompose the computation of the distance between populations into a series of computations of $\min\{i \in \mathbb{N} \mid \{x\} \in S_i(P)\}$ for some individual x and population P. Therefore an efficient method to carry on this computation translates immediatly in an efficient way of computing the proposed distance.

4.4 A CONCISE MODEL FOR POPULATIONS

In this section we define a succinct representation for populations.

Definition 4.4.1. We define as *crossover granules* $(SC_n, \subseteq)$ the set $SC_n = \{[i,j] \mid 1 \leqslant i \leqslant j \leqslant n\}$ ordered by set inclusion.

Proposition 4.4.1. *$(SC_n, \subseteq)$ is a lattice.*

Proof. Let $[i,j], [h,k] \in SC_n$. We show that both $[i,j] \wedge [h,k]$ and $[i,j] \vee [h,k]$ exist and are $[\max\{i,h\}, \min\{j,k\}]$ (that will be denoted by $[M_1, m_1]$) and $[\min\{i,h\}, \max\{j,k\}]$ (that will be denoted by $[m_2, M_2]$) respectively.

It is immediate that $[M_1, m_1] \subseteq [i,j]$ and $[M_1, m_1] \subseteq [h,k]$. It is necessary to prove that every other $[\ell_1, \ell_2] \in SC_n$ such that $[\ell_1, \ell_2] \subseteq [i,j]$ and $[\ell_1, \ell_2] \subseteq [h,k]$ is also such that $[\ell_1, \ell_2] \subseteq [M_1, m_1]$. Suppose $[M_1, m_1] \neq \emptyset$ otherwise the property is vacuously true. For the sake of argument suppose that there exists $a \in [\ell_1, \ell_2]$ such that $a \notin [M_1, m_1]$. This means that either $a < M_1$ or $a > m_1$. Without loss of generality suppose we are in the first case. Then either $a < i$ or $a < h$. This means that $a \notin [i,j]$ or $a \notin [h,k]$ in contradiction with one of the hypotheses. Thus, $[i,j] \wedge [h,k] = [M_1, m_1]$.

It is also immediate that $[m_2, M_2] \supseteq [i,j]$ and $[m_2, M_2] \supseteq [h,k]$. It is necessary to prove that every other $[\ell_1, \ell_2] \in SC_n$ such that $[\ell_1, \ell_2] \supseteq [i,j]$ and $[\ell_1, \ell_2] \supseteq [h,k]$ is also such that $[\ell_1, \ell_2] \supseteq [m_2, M_2]$. For the sake of argument suppose that there exists $a \in [m_2, M_2]$ such that $a \notin [\ell_1, \ell_2]$. By definition $m_2 \leqslant a \leqslant M_2$. Since $a \notin [\ell_1, \ell_2]$ then either $\ell_1 > a \geqslant m_1$ or $\ell_2 < a \leqslant M_2$. Without loss of generality suppose that we are in the first case. Then $m_2 \notin [\ell_1, \ell_2]$ but $m_2 = i$ or $m_2 = h$. This means that $[\ell_1, \ell_2] \not\supseteq [i,j]$ or $[\ell_1, \ell_2] \not\supseteq [h,k]$, negating one of the hypotheses. Thus $[i,j] \vee [h,k] = [m_2, M_2]$.

Since both the meet and the join exist for every and are unique for every pair of elements, $(SC_n, \subseteq)$ is a lattice. □

Definition 4.4.2. Let $x \in \Sigma^n$, $[i,j] \in SC_n$ and $P \in \mathbf{P}$. We say that $x(i,j)$ is represented in P iff there exists $y \in P$ such that $y_{[i,j]} = x_{[i,j]}$.

Note that if $x(i,j)$ is represented in P it is also true that for all $h \geqslant i$ and for all $k \leqslant j$, $x(h,k)$ is represented in P. Also note that if $x(1,n)$ is represented in P then $x \in P$.

It is now possible to define the concept of representation for a population.

Definition 4.4.3. Fix $x \in \Sigma^n$. We define $r_x : \mathbf{P} \mapsto \mathcal{P}(SC_n)$ as $r_x(P) = \{[i,j] \in SC_n \mid x(i,j) \text{ is represented in } P\}$.

We now prove that $r_x(P)$ is always a lower set of SC_n.

Proposition 4.4.2. *For all $P \in \mathbf{P}$ and for all $x \in \Sigma^n$, $r_x(P) \in \mathscr{O}(SC_n)$.*

Proof. Consider $[i,j] \in r_x(P)$. By definition $x(i,j)$ is represented in P. This means that also all $x(h,k)$ with $h \geqslant i$ and $k \leqslant j$ are represented in P. In other words, $[h,k] \in r_x(P)$. Recall that the elements of SC_n in the form $[h,k]$ with $h \geqslant i$ and $k \leqslant j$ are all the elements of SC_n with $[h,k] \subseteq [i,j]$. Since they are in $r_x(P)$ we have that it is a lower set. □

Now we define a function on $\mathscr{O}(SC_n)$ that can be used to "mimic" the Čech closure defined over **P**.

Definition 4.4.4. Let $U \in \mathscr{O}(SC_n)$. We define $\mu_{SC} : \mathscr{O}(SC_n) \mapsto \mathscr{O}(SC_n)$ as

$$\begin{aligned}\mu_{SC}(U) \quad = \quad & \{[i,j] \in SC_n \mid \exists [h_1,k_1],[h_2,k_2] \in U \text{ s.t.} \\ & [i,j] = [h_1,k_1] \vee [h_2,k_2] = [h_1,k_1] \cup [h_2,k_2]\}\end{aligned}$$

Note that $\mu_{SC}(U)$ can also be formulated as:

$$\bigcup_{\substack{[h_1,k_1],[h_2,k_2] \in U \\ k_1 \geqslant h_2 - 1 \text{ or} \\ k_2 \geqslant h_1 - 1}} [h_1,k_1] \vee [h_2,k_2]$$

We are now going to state and prove the main result that allow us to work on the lower set of the lattice SC_n instead of **P**.

Theorem 4.4.3. *For all $x \in \Sigma^n$, the following diagram commutes:*

$$\begin{array}{ccc} \mathbf{P} & \xrightarrow{next} & \mathbf{P} \\ \downarrow r_x & & \downarrow r_x \\ \mathscr{O}(SC_n) & \xrightarrow{\mu_{SC}} & \mathscr{O}(SC_n) \end{array}$$

In other words, $r_x \circ next = \mu_{SC} \circ r_x$.

Proof. Fix $P \in \mathbf{P}$ and $x \in \Sigma^n$. Consider $z, v \in P$. Also, recall that $r_x(P)$ can be seen as $\bigcup_{y \in P} r_x(\{y\})$. Firstly, we are going to prove that $r_x(next(P)) \subseteq \mu_{SC}(r_x(P))$. Let $y \in \Sigma^n$ be such that there exists $w \in \Sigma^n$ such that $(z,v)R_I(y,w)$. Then $y \in next(P)$. Consider $r_x(\{y\})$. Let $[i,j] \in r_x(\{y\})$. This means that $y_{[i,j]} = x_{[i,j]}$ there can be two cases:

1. Either $[i,j] \in r_x(\{z\})$ or $[i,j] \in r_x(\{v\})$. In this case $[i,j] \in \mu_{SC}(r_x(\{z\}))$ or $[i,j] \in \mu_{SC}(r_x(\{v\}))$ (since μ_{SC} is monotone).

2. Neither $[i,j] \in r_x(\{z\})$ nor $[i,j] \in r_x(\{v\})$. In this case, by definition of crossover there we must have k such that $i \leqslant k < j$, $z_{[i,k]} = x_{[i,k]}$ and $v_{[k+1,j]} = x_{[k-1,j]}$ (or the same with z and v swapped). Hence $[i,k] \in r_x(\{z\})$ and $[k+1,j] \in r_x(\{v\})$. We have that $[i,k] \vee [k+1,j] = [i,k] \cup [k+1,j] = [i,j]$. Therefore $[i,j] \in \mu_{SC}(r_x(\{z\}) \cup r_x(\{v\}))$.

Combining both cases for all $[i,j] \in r_x(\mathit{next}(P))$ we have that $[i,j] \in \mu_{SC}(r_x(P))$.

We are now going to prove that $r_x(\mathit{next}(P)) \supseteq \mu_{SC}(r_x(P))$ and, combining with the previous result, that $r_x(\mathit{next}(P)) = \mu_{SC}(r_x(P))$. Consider $[i,j] \in \mu_{SC}(r_x(P))$. This means that there exists $[i,k]$ and $[h,j]$ in $r_x(P)$ such that $[i,k] \vee [h,j] = [i,j]$. Note that this means that $i \leqslant h$, $k \leqslant j$ and $h \leqslant k+1$. Since $[i,k]$ and $[h,j]$ are in $r_x(P)$ there exists two individuals $z, v \in P$ such that $z_{[i,k]} = x_{[i,k]}$ and $v_{[h,k]} = x_{[h,k]}$. Now consider the individual $y = z_{[i,k]}v_{[k+1,j]}$. Since it is obtained by the crossover z and v it is inside $\mathit{next}(P)$. But $r_x(\{y\})$ contains $[i,j]$ since $y_{[i,j]} = z_{[i,k]}v_{[k+1,j]} = x_{[i,k]}x_{[k+1,j]} = x_{[i,j]}$. Hence the $r_x(\mathit{next}(P)) \supseteq \mu_{SC}(r_x(P))$. □

As a corollary we immediately have that:

Corollary. *For all* $P \in \mathbf{P}$ *and for all* $x \in \Sigma^n$, $\{x\} \in S_1(P)$ *iff* $[1,n] \in \mu_{SC}(r_x(P))$.

Consequently, we have another way of computing a property of the Čech closure defined over populations by simply using the function μ_{SC} defined on SC_n:

$$\min\{i \in \mathbb{N} \mid \{x\} \subseteq S_i(P)\} = \min\{i \in \mathbb{N} \mid [1,n] \in \mu_{SC}^i(r_x(P))\}$$

Also, note that since the function μ_{SC} maps lower sets to lower sets, finding the first $i \in \mathbb{N}$ such that iterating μ_{SC} for i times gives a set containing $[1,n]$ is equivalent to calculating the number of iterations necessary to obtain SC_n as a result. Furthermore, it is immediate that μ_{SC} is monotone, hence we can always iterate μ_{SC} until a fixed point is reached (the monotonicity of μ_{SC} and the finiteness of SC_n assures us the this is always the case). If the fixed point reached is SC_n, then the number of steps used is the minimum number i such that $\{x\} \subseteq S_i(P)$, otherwise no such i exists.

4.4.1 *An Analysis of Computational Complexity*

The algorithm to compute the distance between two populations P_1 and P_2 is composed of two parts that must be carried on for all $x \in P_1$ (symmetrically, also for all $x \in P_2$):

1. Computing $r_x(P_2)$.
2. Finding the fixed point of μ_{SC}.

First of all, it is necessary to note that SC_n has size $O(n^2)$. Hence, the first step can be carried on in time $O(|P_2|n^3)$ steps (for any element we assume that we are comparing two strings of length n). In the second step we have that computing the fixed point of μ_{SC} can be carried on in time $O(n^7)$ steps. This time complexity is obtained

in the following way: since μ_{SC} is monotone we can have at most $|SC_n|$ steps before reaching a fixed point. Every set U we are managing is of size at most $|SC_n|$ and the computation of μ_{SC} considers a comparison (in time $O(n)$) of all the pairs of U. This means that we are doing at most $O(n^2)$ iterations, where every iteration consists of $O(n^4)$ operations that can be performed in time $O(n)$. Since these operations must be carried on for all elements of P_1, assuming all populations of size bounded by a certain constant m we have that the total computational time is $O\left(m^2n^3 + mn^7\right)$.

The time complexity bounds can be made more strict by considering two facts. The first one is that every lower set is defined by its set of maximal elements (that forms an antichain, i.e., a set such that every pair of elements is not comparable). The size of the maximal antichain of SC_n is n (its elements are $[1,1],[2,2],\ldots,[n,n]$). Therefore, we can consider only sets of size $O(n)$ instead of size $O(n^2)$ and reduce the number of comparisons to $O(n^2)$ instead of $O(n^4)$. Furthermore, we may notice that if the previously cited maximal antichain is not inside $r_x(P)$, we cannot obtain $[1,n]$ into the fixed point, hence we can consider only computations when the representation of a population contains the maximal antichain. Note that when we have a set with the maximal antichain, the first iteration will contain all the elements in the form $[i,i+1]$, the second iteration all the sets in the form $[i,i+2]$ and, more generally, the j^{th} iteration will contains all the sets in the form $[i,i+2^{j-1}]$. Since μ_{SC} applied to a lower set gives a lower set, we have that a fixed point is reached in $O(\log(n))$ iterations (instead of our previous bound of $O(n^2)$). Thus, the total running time can then be bounded by $O\left(m^2n^3 + mn^3\log(n)\right)$.

4.5 FINAL REMARKS

A crossover-based distance for genetic algorithms (GAs) has been defined. Furthermore, an algorithm of polynomial complexity in the population size and individuals length to compute this distance has been introduced. The novelty of the proposed approach consists in the following points:

- the defined distance is between populations (from which it is straightforward to obtain a family of distances between individuals), which makes modelling crossover easier;
- the representation of the GA dynamics by means of iteration of a Čech closure;
- the mathematical tools used for representing populations in our model (lower sets of a lattice).

The proposed distance could be applied to many different scenarios in GA. For example it can be of help in determining problem

difficulty when used for computing the fitness distance correlation. Also, it can improve the performances of GA when used as the distance for fitness sharing. There are also other applications specific to distances between populations. For example we can try to quantify the "quality" of the genetic material of a population by computing its distance to a set of other populations. A low average distance means that the genetic material in the population is "good" (i.e., it is easy to generate new individuals).

Future work is focused on the extension of this distance to other kinds of crossover, also with the long term goal of extending it to a wider range of evolutionary algorithms (EAs). In particular, a general (representation-independent) way of extending and computing this distance should be devised, in order to provide a coherent framework for the analysis of the EAs dynamics.

Part II

MEASURING AND INCREASING SOLUTION QUALITY IN GENETIC PROGRAMMING

5

INTRODUCTION TO GENETIC PROGRAMMING

In this part of the thesis we focus on introducing methods to assess and increase the performances of Genetic Programming (GP). We must firstly introduce Genetic Programming in Section 5.1, and then we will focus on a particular variation called Semantic GP (Section 5.2). We will then focus on the importance of generalization in GP (Section 5.3), that is essential to allow any machine learning algorithm to be used beyond the data provided as a training set. The remaining part of the chapter provides an overview on the benchmarks in GP (Section 5.4) and introduces a famous benchmark for Genetic Algorithms, the NK landscapes (Section 5.5), and the important concept of fitness landscape (Section 5.6) that is necessary to understand the significance of the NK landscapes benchmark for Genetic Algorithms. Finally, previous GP benchmarks are discussed in Section 5.7.

In the next chapters we will first introduce a way to measure the generalization ability of GP (Chapter 6) and then a benchmark for GP based on NK landscapes for GA (Chapter 7). Finally, we introduce a method that makes Semantic GP viable for real world problems and we test its generalization ability (Chapter 8).

5.1 WHAT IS GENETIC PROGRAMMING

Genetic Programming [162, 114] is the youngest paradigm inside the computational intelligence research area called Evolutionary Computation (EC) and consists in the automated learning of computer programs by means of a process mimicking Darwinian evolution. A GP algorithm works by maintaining and evolving a set (often called *population*) of so called *individuals*, each of which representing a program that is a potential solution to a problem. A *fitness* function (sometimes also called cost or quality function), defined over the space of all individuals, quantifies the ability of each one of them in solving the problem. After a (typically) random initialisation of the population, the evolutionary process, aimed at progressively improving the fitness of the individuals in the population, takes place by iterating two phases: selection (where the most promising solutions are probabilistically chosen for mating), and the application of the genetic operators (used to explore the space of solutions), which typically consist

in crossover (that exchanges parts of two parent solutions in order to generate new offspring) and mutation (that randomly modifies parts of some solutions). In GP individuals are traditionally represented as LISP-like trees. The standard convention for tree execution is that it proceeds by repeatedly evaluating the nodes in postfix order. For ex-

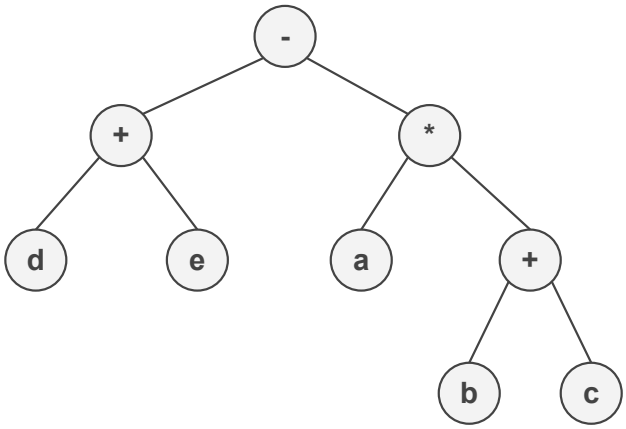


Figure 5: Example of GP individual.

ample, the tree in Fig. 5 represents the function $(d + e) - (a(b + c))$, where a, b, c, d, e are variables.

In the last few years, GP has been extensively used both in Industry and Academia and it has produced a wide set of results that have been defined *human-competitive* [115]. These results cover a wide variety of applicative domains, including quantum computing circuits, analog electrical circuits, design of antennas, mechanical systems, photonic systems, optical lens systems and sorting networks.

While these results have demonstrated the suitability of GP in tackling real-life problems, research has recently focused on developing new variants of GP in order to further improve its performances. In particular, efforts have been dedicated to an aspect that was only marginally considered up to some years ago: the definition of methods based on the semantics of the solutions [195, 131, 25, 19, 197, 117, 19, 20, 95, 96]. Though there is no universally accepted definition of semantics in GP, this term often refers to the behavior of a program, once it is executed on a set of data. For this reason, in many references, the term semantics is intended as the set of input-output pairs on the training data, and this is the terminology that we will adopt from now on.

In this research track, very recently, new genetic operators, called geometric semantic genetic operators, have been proposed in [142]. These operators have the interesting property of inducing a unimodal fitness landscape on any problem consisting in finding the match between a set of input data and a set of known output ones. Classification and regression (which are typical applications of Evolutionary Computing in general [190, 4] and GP in particular [39, 87]) are ex-

amples of this kind of problem. According to the theory of fitness landscapes [122], this should allow GP to easily solve all these problem.

Nevertheless, as stated in [142], these operators also have a serious limitation: they construct offspring that are bigger than their parents, and this makes the size of the individuals in the population grow exponentially with generations. In this way, after few generations, the population is composed by individuals that are so big that their fitness evaluation is unmanageable. This limitation makes these operators impossible to use in practice.

The solution suggested in [142] as a future work is to integrate in the GP algorithm a "simplification" phase, aimed at transforming each individual in the population into an equivalent (i.e. with the same semantics), but smaller program. However, also this solution has some problems: according to the language used to code individuals, simplification can be very difficult, and it is often a very time consuming task.

5.2 STATE OF THE ART ON THE USE OF SEMANTICS IN GP

The genetic operators in GP systems are usually designed with the only constraint of ensuring syntactic closure, i.e. producing syntactically valid offspring from any syntactically valid parent(s). As stated in [195], using such purely syntactical genetic operators, GP evolutionary search is conducted on the syntactical space of programs, with the only semantic guidance offered by the fitness function employed by selection.

As stated before, GP has been used to successfully solve real life problems; however the usage of purely syntactical genetic operators it is not able to describe the entire dynamic of the evolutionary process. Thus incorporating semantic awareness in the GP process could improve performance, extending its applicability to problems that are difficult with purely syntactic approaches. Under this perspective, several recent works have proposed the definition of semantic based methods.

These methods appeared in combination with crossover. McPhee et al. [131] used truth tables to analyze behavioral changes in crossover for boolean problems. They considered the semantics of two components in each tree: semantics of subtrees and semantics of context (the remainder of an individual after removing a subtree). They experimentally measured the variation of these semantic components throughout the GP evolutionary process. They payed special attention to fixed-semantic subtrees: subtrees such that the semantics of the entire tree does not change when they are replaced by another subtree. They showed that there may be many such fixed semantic subtrees when the tree size increases during evolution; thus it be-

comes very difficult to change the semantics of trees with crossover and mutation once the trees have become large.

While it is possible to represent behavior using truth tables, a more efficient technique is that of using reduced ordered binary decision diagrams (ROBDDs) [25] to create reduced canonical representations to measure behavioral difference.

In [19] semantic is used to define an algorithm called Semantically Driven Crossover (SDC). The SDC algorithm has been developed based on analysis of the behavioral changes caused by crossover. The key feature of this method is the use of a canonical representation of members of the population (reduced ordered binary decision diagrams-ROBDDs) to check for semantic equivalence without having to access the fitness function. Two trees are semantically equivalent if and only if they reduce to the same ROBDD. This is used to determine which participating individuals are copied into the next generation. If the offspring are semantically equivalent to their parents, the children are discarded and the crossover is repeated. This process is iterated until semantically different children are found. The authors argue that this results in increased semantic diversity in the evolving population, and a consequent improvement in the GP performance.

A new mechanism for studying the impact of subtree crossover in terms of semantic building blocks was proposed [130] in 2007. This approach allows to completely and compactly describe the semantic action of crossover, and provides insight into what does (or does not) make crossover effective. Results make it clear that a very high proportion of crossover events (typically over 75% in the presented experiments) are guaranteed to perform no immediately useful search in the semantic space.

In [196] the authors investigate the role of syntactic locality and semantic locality of crossover in GP. The results show that improving syntactic locality reduces code growth, and that leads to a slight improvement of the ability to generalize. By comparison, improving semantic locality significantly enhances GP performance, reduces code growth and substantially improves the ability of GP to generalize.

In [150] the authors proposed Semantics Aware Crossover (SAC), a crossover operator promoting semantic diversity, based on checking semantic equivalence of subtrees. It showed limited improvement on some test problems; it was subsequently extended to Semantic Similarity based Crossover (SSC) [197], which turned out to perform better than both standard crossover and SAC [197]. In particular, authors aim to incorporate semantics into the design of new crossover operators, so as to maintain greater semantic diversity and provide higher locality than standard crossover. The idea of SSC was then extended to mutation leading to a counterpart semantic mutation: Semantic Similarity based Mutation (SSM) [164]. The experimental

results in [164] confirm the superior performance of SSM compared to standard mutation.

In [19] semantics is used to test the effects of behavioral control at the point of the mutation operator. Using semantic analysis, authors present a technique known as semantically driven mutation (SDM), which can explicitly detect and apply behavioural changes caused by the syntactic modifications in programs caused by mutation. The SDM algorithm does not allow mutated programs to be produced when they are behaviorally equivalent to the original program. The aim of this is to avoid getting stuck in areas of the search space that have already been investigated. As in [19], the key feature of the semantically driven operator is the ability to canonically represent programs in such a way that it is possible to compare them, looking for equivalent behaviors.

In [116] the authors proposed a class of crossover operators for genetic programming aimed at making offspring programs semantically intermediate (medial) with respect to parent programs by modifying short fragments of code (subprograms). The approach is applicable to problems that define fitness as a distance between program output and a desired target. Based on that metric, the authors defined two measures of semantic "mediality", which they employed to design two crossover operators: one aimed at making the semantic of offspring geometric with respect to the semantic of parents, and the other aimed at making them equidistant to parents' semantics. When compared experimentally with four other crossover operators, both operators lead to success ratio at least as good as for the non-semantic crossovers, and the operator based on equidistance outperformed all others on some test cases.

Krawiec and coworkers in [117] have used a notion of semantic distance to propose a crossover operator for GP that is approximately a geometric crossover [141] in the semantic space. In the class of problems considered in [117], the fitness function is usually defined as a metric that measures the divergence between target and output values. As reported in [117], metric-based fitness functions are unimodal by definition because such fitness is a distance in the semantic space. Any linear combination of a pair of semantics is guaranteed to be not worse than the worse of them. Authors pointed out that there is no obvious way of exploiting this property due to the complexity of the genotype-phenotype mapping in GP. Thus, the prospects of designing a crossover operator that works in the genotype space and behaves geometrically in the corresponding semantic space are even more gloomy. Hence, rather than guaranteeing the geometric behavior, their operator tries to approximate it by analysing the offspring after it has been bred. This limit is overcome by the geometric semantic operators proposed in [142] and described in section 8.1. The work in [142] introduces a general method to derive exact semantic

geometric crossovers and mutations for different problem domains that search directly the semantic space. However, these operators by construction produce offspring that have approximately the double of the size of their parents (expressed as the total number of tree nodes). As a consequence, the size of the individuals in the population grows exponentially (as proven in [142]) and this makes these operators unusable in practice.

5.3 GENERALIZATION IN GP

5.3.1 *The Importance of Generalization*

The issue of generalization in Genetic Programming (GP) [162] has received a growing attention in the last few years (see [121] for a survey and [156] and references therein for a more recent discussion). A common agreement of many researchers is the so called minimum description length principle (see for instance [170]), which states that the best model is the one that minimizes the amount of information needed to encode it. Simpler solutions are thought to be more robust and generalize better [171], while complex solutions are more likely to incorporate specific information from the training set, thus overfitting it. A superficial interpretation of the minimum description length may be that bloat (i.e. an excess of code growth without a corresponding improvement in fitness [122, 178]) and overfitting should be two related phenomena: bloated programs could in fact use too much information, thus overfitting training data. Nevertheless, the observations in [179, 200] seem to contradict this idea. In [203] Vanneschi and coworkers defined measures to quantify bloat, overfitting and functional complexity of GP solutions, and they showed that functional complexity, rather than bloat, seems to be related to generalization. But the experimental results reported in [203] show that the proposed complexity measure has a positive correlation with overfitting only for a subset of the studied test problems, while for other problems no clear relationship appeared between this measure and the ability of GP to find general solutions.

5.3.2 *Studies on Generalization*

The contributions on generalization in GP are so numerous that it is impossible to analyze all of them. Thus, we focus on those contributions that we consider more similar and related to the present work, referring the interested reader to [121] for a survey on the issue. In 1995 Zhang and Mühlenbein investigated the relationship between generalization and parsimony and proposed an adaptive learning method that automatically balances these two factors. One year later, in [72], a new GP system called Compiling GP System was intro-

duced and in [16], Banzhaf and coworkers showed the positive effect of an extensive use of the mutation operator on generalization in GP using sparse data sets. More recently, in [77], Gagné and coworkers have investigated two methods to improve generalization in GP: the selection of individuals using a three data sets methodology, and the application of parsimony pressure to reduce the size of the solutions. In the last few years the idea of quantitatively studying the relationship between generalization and solution complexity was tackled in several contributions. For instance, in [7] the authors propose a theoretical analysis of GP from the perspective of statistical learning theory and prove the advantage of a parsimonious fitness using Vapnik-Chervonenkis theory. Another important contribution, even though not explicitly focused on GP, but on evolutionary algorithms in general, is represented by [81], where the authors measure behavioral complexity explicitly using compression, and use it as a separate objective to be optimized. In [209], the authors define two measures of complexity for the GP individuals: a genotypic measure and a phenotypic one. While the genotypic measure is related to counting the number of nodes of a tree and its subtrees, the phenotypic one, called *order of nonlinearity*, is related to functional complexity and consists in calculating the degree of the Chebyshev polynomial approximation of the function. The authors then use these two measures as criteria in a multi-objective system and they show that this system is able to counteract both bloat and overfitting. As already pointed out in the previous section, in [203] indicators of bloat, overfitting and complexity have been introduced and their mutual relationships have been investigated. The overfitting measure quantified the relationship between training and test fitness, normalizing it with the training and test fitness of the point with the best test fitness found by GP so far (the reader is referred to page 878 of [203] for the pseudo-code used to calculate this measure). The complexity measure was used to approximate the "degree of curvature" (or "ruggedness") of the function expressed by GP solutions and it was basically a weighted sum of the slopes of the segments joining the various training points in the single dimensions (the formal definition of this complexity measure can be found at page 880 in [203]).

5.4 BENCHMARKS IN GP AND THE NK LANDSCAPES

As already pointed out by Altenberg in 1997 [5], in the first part of the twentieth-century Wright [220] discovered an interesting feature of evolutionary dynamics: when the effect on fitness from altering the state of one gene depends on the state of other genes, the population can often evolve into multiple basins of attraction. This kind of interaction between genes is called *epistasis*. In other words, the presence of epistasis makes it possible for a population to converge towards dif-

ferent genotypic configurations, depending on its initial state. Thus, it is possible to indentify two different levels of abstraction, that are different to, but dependent from, each other: the microscopic level (the level of the single individuals, in which fitness depends on the reciprocal interactions between genes) and the macroscopic one (the level of populations, where it is possible to identify multiple different attractors). To clarify these ideas, Wright brought up the analogy with a landscape with multiple peaks, in which the evolution of a population can be seen as the movement up hill of its individuals, until they reach a (local or global) fitness peak. The term "adaptive landscape" or "fitness landscape" is nowadays frequently used to describe the presence of multiple basins of attraction in the space of genotypes for evolutionary dynamics. Under this perspective, it becomes interesting to investigate how a population can escape from a local fitness peak. Wright proposed the idea of stochastic fluctuations, thus introducing an embryonic version of a stochastic process for the optimization of multimodal functions. More recently, based on the concepts introduced by Wright, Kauffman proposed the NK landscapes set of functions, to investigate the way epistasis controls the number of local peaks of a fitness landscape [106, 107]. In these functions, the ruggedness of the fitness landscape can be controlled by modifying a single parameter that influences the epistatic level of the genome. Thus, the NK landscapes is a stochastic set of tunably difficult optimization problems. Described in Section 5.5, the NK landscapes have often been used as a benchmark for theoretical studies of Genetic Algorithms (GAs) [93, 80].

NK landscapes are based on the idea that potential solutions are represented as fixed length strings of symbols, and thus they are particularly suitable for GAs. To the best of our knowledge, no extension to this problem has been proposed so far for Genetic Programming (GP) [114, 162], where individuals are characterized by a dynamic size representation. Nevertheless, as clearly stated by the panel members and attendees of the EuroGP 2008 debate on Grand Challenges of GP (which took place on 27 March 2008 at the Evo* event in Naples, and whose main ideas have recently been developed and published in [156]), and as reasserted during the subsequent EuroGP 2009 and EuroGP 2010 debates, the GP community has the pressing necessity of defining new and reliable benchmarks. Those benchmarks should possibly be of tunable difficulty, thus allowing practitioners and theoreticians to study the dynamics of GP for difficult, as well as easier problems. These benchmarks should, above all, be used to study and discover new properties of the GP method itself, as it has been the case for NK landscapes in GAs.

Indeed, some efforts have already been done in the direction of defining useful benchmarks for GP (they are reviewed in Section 5.7), but still the goal of having many rigorous test problems is far from be-

ing achieved, probably due to the larger complexity of GP compared to GAs or other methods for parameters optimization.

5.5 THE NK LANDSCAPES FOR GAS

The NK family of landscapes, introduced by Kauffman between the end of the eighties and the first part of the nineties [106, 107], is a problem-independent model for constructing multimodal landscapes for GAs that can gradually be tuned from smooth to rugged. In the model, N refers to the number of (binary) genes in the genotype (i.e. the string length) and K to the number of genes that influence a particular gene (the epistatic interactions). By increasing the value of K from 0 to $N-1$, NK landscapes can be tuned from smooth to rugged. The fitness of a NK landscape is a function $f_{NK} : \{0,1\}^N \to [0,1)$, defined on binary strings with N bits. An "atom" with fixed epistasis level is represented by a fitness component $f_i : \{0,1\}^{K+1} \to [0,1)$ associated to each bit i. Its value depends on the allele at bit i and also on the alleles at K other epistatic positions. (K must fall between 0 and $N-1$). The fitness $f_{NK}(s)$ of $s \in \{0,1\}^N$ is the average of the values of the N fitness components f_i: $f_{NK}(s) = \frac{1}{N}\sum_{i=1}^{N} f_i(s_i, s_{i_1}, \ldots, s_{i_K})$, where $\{i_1, \ldots, i_K\} \subset \{1, \ldots, i-1, i+1, \ldots, N\}$. Many ways have been proposed to choose the K other bits from the N bits forming the bit string. Two possibilities are mainly used: adjacent and random neighborhoods. With an adjacent neighborhood, the K nearest bits to the bit i are chosen. With a random neighborhood, the K bits are chosen randomly on the bit string. Each fitness component f_i is specified by extension, *i.e.* a number $y^i_{s_i, s_{i_1}, \ldots, s_{i_K}}$ from $[0,1)$ is associated with each element $(s_i, s_{i_1}, \ldots, s_{i_K})$ from $\{0,1\}^{K+1}$. Those numbers are usually random numbers, uniformly distributed in the range $[0,1)$.

As reported by Kauffman, epistasis have a repercussion on fitness that is similar to a *house of cards* [110]: if a bit is modified in a given position, all the fitness components that interact with it are changed, without any correlation with their previous values. Thus, the modification of one single bit, for instance by the application of a bit-flip mutation operator, has on fitness the same disruptive effect as removing a card from a house of cards: to find the same fitness value again (i.e. to build the same house of cards once again) one has to restart from scratch and it is not possible to use any of the previously processed information for doing it. Thus, any algorithm that works by trying to optimize all genes at the same time clearly encounters serious problems in this kind of benchmark. The interested reader is referred to [5] for a survey on NK landscapes, where some properties of this benchmark, like the computational complexity, as well as some implementation details and alternative variants are discussed. Examples of the numerous contributions in which the NK landscapes

have been used to investigate the GA dynamics are [107, 214, 71, 102, 101, 3, 15, 79, 180, 207, 202, 152, 208].

5.6 FITNESS LANDSCAPES

The concept of fitness landscape was first proposed in [219] to study the evolutionary process in Biology. The notion of a fitness landscape underlying the dynamics of evolutionary adaptation optimization has proved to be one of the most powerful concepts in evolutionary theory, and it has been widely used to model the problem difficulty in evolutionary algorithms (EAs) [165, 199]. As reported in [187], implicit in the idea of fitness landscape is a collection of genotypes arranged in an abstract metric space, with each genotype next to those other genotypes which can be reached by a single application of a given genetic operator, as well as the fitness value.

As described, for instance, in [198], a fitness landscape can be seen as a three-dimensional map, which may contain peaks and valleys and the problem solver as a short-sighted explorer searching for the highest peak (for maximization problems). They can be formally modeled and are helpful to understand the ability of a searcher like GP to solve a problem. For example, a smooth and regular landscape with a single hill top (i.e. unimodal) is typical of an easy problem, while the opposite is true for a very rugged (i.e. multimodal) landscape, with many hills which are not as high as the best one. In the latter case, it is more difficult to find solutions (the highest peaks), since the algorithms can be trapped in any local peak.

Even though not without faults, the general knowledge associated to fitness landscapes is the more rugged landscape, the more difficult the problem. It has been known since Eigen's work [57] that the dynamics of optimization on a landscape depends crucially on detailed structure of the landscape itself. Extensive computer simulations [69, 70], have made it very clear that a complete understanding of the dynamics is impossible without a thorough investigation of the underlying landscape [58].

In practice, however, the visualization of the whole search space of a problem is difficult, if not even impossible given its generally huge size and the multi-dimensionality of the neighborhoods imposed by canonic genetic operators. Therefore, a number of methods that attempt to describe the relevant features of fitness landscapes by means of numeric indicators have been proposed [215, 204, 100]. For a complete review on fitness landscapes in EC (and GP in particular) the reader is referred to [199]. The most used indicator that relates problem difficulty with the underlying fitness landscape is fitness distance correlation (FDC), studied for GP in [201]. *FDC* quantifies the difficulty of a problem by expressing the correlation between the fitness

of a sample of individuals and their distance to one globally optimal solution.

5.7 PREVIOUS GP BENCHMARKS

In Koza's 1992 book [114], several problems that can be solved with GP are defined: k-even parity, h-multiplexer, various forms of symbolic regression, the artificial ant on the Santa Fe trail, the intertwined spirals problem, etc. For many years, and with few exceptions, those problems have mainly represented the only benchmarks that have been used in GP experimental studies. Only recently, the GP community has begun to use a larger set of test functions. It is the case, for instance, of the UCI repository datasets suite [73]. Among them, one can identify problems of different complexities, from trivial ones, like the IRIS dataset, to more difficult ones, like the thyroid cancer datasets and many others. Also, contributions have appeared using as test problems applications like classification of network intrusion (see for instance [183, 157]). As stated in [156], we could broadly classify currently used test function in GP into three categories: regression, classification and design. Typical used regression functions are sinus, polynomials, mexican hat, Mackey-Glass time series, etc. For classification, one may quote the UCI examples, the intertwined spirals, the parity problems, the multiplexer, Protein Localization, etc. The class of test problems classified as design contain applications like adder, multiplier, several other circuits, trusses, tables and other structures.

All the above quoted test problems, from the early ones introduced by Koza in 1992 to the more recent ones, are definitely interesting, because they cover a set of different possible applications. Nevertheless, it is also true that compared to the set of typical test problems used for other optimization methods, like for instance GAs or particle swarm optimization, this set of benchmarks is still a restricted one and, even more importantly, lacks a rigorous evaluation. In particular, the majority of these problems have their own, so to say, "fixed" complexity: it is not possible to define several instances of them, of different difficulties (from very easy problems to very hard ones, and including many intermediary cases) by changing some parameters (this is partially false, for instance, for the even parity and multiplexer problems, as already pointed out in [199], but it is surely true for all the problems that consist in mining a given dataset like, for instance, the UCI ones). Furthermore, as remarked in [156], the above quoted set of test problems is composed by functions of similar nature, and thus they are too similar to each other. For instance, these functions would require too a restricted set of operators and terminals to build individuals, compared to the huge variety of possibilities offered by GP.

Some attempts of defining new and tunably difficult test problems for GP have been made so far. One of the earliest ones probably consists in the introduction of the Royal Trees by Punch and coworkers [163]. The goal of that contribution was to devise a problem for GP that could be similar to, and share interesting properties with, the the Royal Road problem for GAs [133]. In particular, the Royal Trees are an example of "constructional problems", in the same vein as the Royal Roads, a concept that had already been introduced in GP by Tackett [189]. More or less in the same vein, contributions [199, 193] contain the extension of tunably difficult problems typical of GAs, like various forms of trap functions [45], to GP. In [86], Gustafson and coworkers introduced the Tree-String problem. The goal of this problem is to derive specific structure and content elements simultaneously. Instances are defined using a target solution consisting of a tree shape and content. Candidate solutions are then measured for their similarity to the target solution with respect to both tree shape and content objectives. Separating these two concepts, the Tree-String problem makes thus possible to shade a light on the complex dynamics created by the interdependencies of solution structures and contents. Furthermore, the authors show how the difficulty of the Tree-String problem can be tuned by simply modifying the number of used nodes and the size of the alphabet employed to code contents. Another interesting contribution, that is related to, but slightly different from, the previously quoted ones, is [41], where Daida and coworkers introduce the Lid problem. The Lid problem is tunably difficult, but, contrarily to Royal Trees, trap functions, and others, it has this property because tuning is accomplished through changes in structure. So, the difficulty of the problem, and the possibility to tune it, depend on structural mechanisms and not on the fitness landscape.

6

LEARNING ABILITY

6.1 INTRODUCTION

This chapter has three different, but related, goals:

1. Defining a new measure of functional complexity, that is rotationally invariant and that overcomes the limitations of the measure proposed in [203];
2. Defining a new measure to quantify the ability of GP to learn "difficult" points (our intuition of what a "difficult" point is will be explained in Section 6.2 where the measure will be defined) and studying its correlation with generalization;
3. defining a new fitness function (inspired by the two previously defined measures) to improve GP generalization ability in those cases where standard GP (i.e. GP that calculates fitness using the root mean squared error between outputs and targets) has poor generalization ability.

The chapter is structured as follows: In Section 6.2 we present the proposed measures. Section 6.3 introduces the real-life problems used as test cases. In Section 15.1.5 we present our experimental settings and we discuss the obtained results. The measures of overfitting and complexity proposed in [203] will be experimentally compared with the measures introduced in this work. Finally, Section 7.4 provides a summery and future directions of research.

6.2 THE PROPOSED MEASURES

The complexity measure proposed in this work, as the one introduced in [203], is inspired by the idea that complex functions should have a larger "degree of curvature" (or "ruggedness") than simple ones. But, contrarily to [203], in the present work we quantify the idea of "degree of curvature" using the following intuition: let g be a GP individual; the "degree of curvature" of g can be expressed by counting the number of pairs of "*close*" training points $\mathbf{a}$ and $\mathbf{b}$ (where both $\mathbf{a}$ and $\mathbf{b}$ are points that belong to the domain of g) for which the corresponding values $g(\mathbf{a})$ and $g(\mathbf{b})$ are "*far*".

In more formal terms, given a GP individual g, let S be the set $\{(\mathbf{x_1}, g(\mathbf{x_1})), (\mathbf{x_2}, g(\mathbf{x_2})), \ldots, (\mathbf{x_m}, g(\mathbf{x_m}))\}$ that is, the set that contains training points $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_m}$ associated with the corresponding values assumed by g on them.

Let $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_m} \in \mathbf{X}$ and $g(\mathbf{x_1}), g(\mathbf{x_2}), \ldots, g(\mathbf{x_m}) \in \mathbf{Y}$, with both $\mathbf{X}$ and $\mathbf{Y}$ being metric spaces equipped with metrics $d_{\mathbf{X}}$ and $d_{\mathbf{Y}}$ respectively. For any $i = 1, 2, ..., m$, and for any prefixed constant value δ, let $B_\delta(\mathbf{x_i})$ be the open ball of radius δ centered on $\mathbf{x_i}$ in metric space $\mathbf{X}$, i.e. $B_\delta(\mathbf{x_i}) = \{\mathbf{x_j} \mid d_{\mathbf{X}}(\mathbf{x_i}, \mathbf{x_j}) < \delta\}$. Analogously, for any $i = 1, 2, ..., m$, and for any prefixed constant value ε, let $B_\varepsilon(g(\mathbf{x_i}))$ be the open ball of radius ε centered on $g(\mathbf{x_i})$ in metric space $\mathbf{Y}$.

For every training point $\mathbf{x_i}$, we define the set:

$$V(\mathbf{x_i}) = \{\mathbf{x_j} \in B_\delta(\mathbf{x_i}) \mid g(\mathbf{x_j}) \notin B_\varepsilon(g(\mathbf{x_i})) \text{ and } \mathbf{x_j} \neq \mathbf{x_i}\}$$

This set contains all the points of the sample set that are close (i.e. nearer than a given δ) to $\mathbf{x_i}$ in the $\mathbf{X}$ metric space, but whose values under g are not close (i.e. farther than a given ε) to the value of $g(\mathbf{x_i})$ in the $\mathbf{Y}$ metric space.

We now consider the set $V = \bigcup_{i=1}^{m} V(\mathbf{x_i})$. V can be addressed as the set of points in $\mathbf{X}$ in which the function represented by the GP individual g is *rugged*, thus the fraction of training points that belong to set V can be used as a measure to quantify our intuition of "degree of curvature" (or "ruggedness"): $\frac{|V|}{m}$. Clearly, values near 1 denote a very rugged function, while values near 0 indicate flat (or straight) and thus less complex functions. It is interesting to consider not only the set $V(\mathbf{x_i})$ containing the *points* of ruggedness since, in the union $\bigcup_{i=1}^{m} V(\mathbf{x_i})$ one misses the information about which *pairs* of points that are close to each other have corresponding function values which are far apart.

So, we introduce the set $E = \bigcup_{i=1}^{m} (\{\mathbf{x_i}\} \times V(\mathbf{x_i}))$, which is a relation that associates each $\mathbf{x_i}$ with all the corresponding points in $V(\mathbf{x_i})$. Now, if we define the set $E_{tot} = \{(\mathbf{x_i}, \mathbf{x_j}) \mid \mathbf{x_j} \in B_\delta(\mathbf{x_i}) \setminus \{\mathbf{x_i}\}\}$, we can define our complexity measure as:

$$GBC = \frac{|E|}{|E_{tot}|}$$

We remark that, in case of symbolic regression problems, we usually have $\mathbf{X} \subseteq \mathbb{R}^n$ and $\mathbf{Y} \subseteq \mathbb{R}$. Thus, it is possible to calculate GBC, for instance, using the Euclidean distance as the $d_{\mathbf{X}}$ and $d_{\mathbf{Y}}$ metrics. Thus GBC is rotationally invariant, contrarily to what happens for the complexity measure defined in [203].

The acronym we have chosen as the name of this measure (GBC, which stands for Graph Based Complexity) depends on the fact that it is possible to represent it in terms of counting operations on a graph. Let $G = (\{\mathbf{x_1}, \ldots, \mathbf{x_m}\}, E_{tot})$ be a graph defined on the training points, where two vertices are connected if their distance on the metric space

$\mathbf{X}$ is less than δ. Now consider the subgraph $G_\varepsilon = (\{\mathbf{x_1}, \ldots, \mathbf{x_m}\}, E)$ which only contains the edges $(\mathbf{x_i}, \mathbf{x_j})$ of G such that the distance between $g(\mathbf{x_i})$ and $g(\mathbf{x_j})$ in the $\mathbf{Y}$ metric space is greater or equal than ε. The GBC measure is clearly equal to the ratio between the number of connections in G_ε and the number of connections in G.

The GBC function can be used to quantify the complexity of GP individuals. However, it is also clear that the same calculation can be performed using the known target values (instead of using the values assumed by the learned function) on the different training points. In particular, if we indicate by $f(\mathbf{x_i})$ the target value on a training point $\mathbf{x_i}$, it is possible to define a set V' (analogous to the set V previously defined) as follows: $V'(\mathbf{x_i}) = \{\mathbf{x_j} \in B_\delta(\mathbf{x_i}) \mid f(\mathbf{x_j}) \notin B_\varepsilon(f(\mathbf{x_i})) \text{ and } \mathbf{x_j} \neq \mathbf{x_i}\}$. And thus, it is also possible to define a set E' as follows: $E' = \cup_{i=1}^{m} (\{\mathbf{x_i}\} \times V'(\mathbf{x_i}))$. Following the same idea that we have used to define the GBC measure, we can state that $GBC_{target} = |E'|/|E_{tot}|$ is a measure that can be used to quantify the ruggedness of the target function.

Furthermore, we can also use *both* information coming from sets E and E' to quantify the ability of a GP individual to learn "difficult" points. For this aim we define the following measure, that we call GBLA (Graph Based Learning Ability):

$$GBLA = \frac{|E \triangle E'|}{|E_{tot}|}$$

where $\triangle$ represents the operator of symmetrical difference between sets. GBLA quantifies the number of training points where the target function is rugged and the learned function is flat, plus the number of training points where the learned function is rugged and the target function is flat. For simplicity, we call these points "difficult" points.

Both the definitions of GBC and GBLA are based on the definition of the $V(\mathbf{x_i})$ set. The elements of $V(\mathbf{x_i})$ depend on the choice of the two parameters δ and ε. Consequently, also the values of GBC and GBLA depend on these two parameters. Nevertheless, a set of preliminary experiments have indicated an interesting fact concerning these two parameters: if we consider many series composed by the values of the GBC of the best individual in the population for each iteration for several pairs of values of δ and ε, all these series have a positive value of their mutual cross correlation coefficient, with a magnitude of this coefficient approximately equal to 1. The same fact has also been observed for GBLA. Given that in the experimental study presented in Section 15.1.5 we are mainly interested in understanding the *correlation* of GBC and GBLA with other quantities during the GP runs (rather than the particular values of GBC and GBLA), we can assert that parameters δ and ε qualitatively affect neither the results of Section 15.1.5, nor the conclusions that we are able to draw from them. We have also repeated all the experiments reported in Section 15.1.5 for several other pairs of values of δ and ε and all the experimental

results (not shown here for lack of space) have confirmed that the values of δ and ε do not affect the qualitative interpretation of the results.

6.3 TEST PROBLEMS

The real-life applications considered in this chapter to test the proposed measures are three multidimensional regression problems, whose goal is to predict the value of as many pharmacokinetic parameters of a set of candidate drug compounds on the basis of their molecular structure. The first pharmacokinetic parameter we consider is human oral bioavailability (indicated with %F from now on), the second one is median lethal dose (indicated with LD50 from now on), also informally called toxicity, and the third one is called plasma protein binding levels (indicated with %PPB from now on). %F measures the percentage of the initial orally submitted drug dose that effectively reaches the systemic blood circulation after the passage from the liver. LD50 refers to the amount of compound required to kill 50% of the considered test organisms (cavies). %PPB quantifies the percentage of the drug initial dose that reaches blood circulation and binds the proteins of plasma. For a detailed discussion of these three pharmacokinetic parameters the reader is referred to [8]. The datasets we have used are the same as in [8]: the %F dataset consists in a matrix composed by 260 rows (instances) and 242 columns (features). Each row is a vector of molecular descriptor values identifying a drug; each column represents a molecular descriptor, except the last one, that contains the known target values of %F. Both the LD50 and the %PPB datasets consist in a matrix composed by 234 rows (instances) and 627 columns (features). Also in this case, each row is a vector of molecular descriptors identifying a drug and each column represents a molecular descriptor except the last one, that contains the known values of the target. For all these datasets training and test sets have been obtained by random splitting: at each different GP run, 70% of the molecules have been randomly selected with uniform probability and inserted into the training set, while the remaining 30% formed the test set.

6.4 EXPERIMENTAL STUDY

Experimental setting. A total of 120 runs were performed to obtain the results reported in this section. All the runs used a population of 200 individuals. The number of generations that we have performed was equal to 100 for the LD50 and %F datasets and to 500 for %PPB (the reason why we have executed a larger number of generations for %PPB will be clear on page 67 in the current section). Tree initialization was performed with the Ramped Half-and-Half method [114]

with a maximum initial depth of 6. The function set contained the four binary operators $+$, $-$, $\times$, and $/$, protected as in [114]. The terminal set contained 241 floating point variables for the %F dataset and 626 floating point variables for the LD50 and %PPB datasets. No random constants were added to the terminal set. Because the cardinalities of the function and terminal sets were so different, we have imposed a balanced choice between functions and terminals when selecting a random node.

Unless where explicitly pointed out, fitness was calculated as the root mean squared error (RMSE) between outputs and targets. Tournament selection was used with size 10. The reproduction (replication) rate was 0.1, meaning that each selected parent has a 10% chance of being copied to the next generation instead of being engaged in breeding. Standard tree mutation and standard crossover (with uniform selection of crossover and mutation points) were used with probabilities of 0.1 and 0.9, respectively. Selection for survival used elitism (i.e. unchanged copy of the best individual in the next population). A fixed maximum depth equal to 17 was used for the trees in the population.

We remark that these parameters are absolutely identical to the ones used in [203]. Given that, as pointed out in the previous section, the δ and ε parameters do not affect the qualitative interpretation of the results contained in this section, we report the results obtained for two arbitrary values, i.e.: $\delta = 0.06$ and $\varepsilon = 0.05$. All distance values have been normalized into the range [0,1] before comparing them with the values of δ and ε.

Experimental results: GBC and GBLA. In Figure 6 we report the median over 120 independent runs of the RMSE on the training set, the RMSE on the test set, the value of GBC, the value of 1-GBLA and finally the complexity and overfitting measures introduced in [203] for all the performed generations (first column: LD50; second column: %F; third column: %PPB). From now on we use the terms "complexity" and "overfitting" to indicate the complexity and overfitting measures introduced in [203] and with the terminology "RMSE on the test set" we indicate the RMSE on the test set of the individual with the best RMSE on the training set.

Let us focus first on the relationship between RMSE on the training and test set for the studied problems. For LD50 the RMSE on the training set steadily decreases during the whole evolution, while the RMSE on the test set keeps increasing after generation 30, also showing an irregular and oscillating behaviour. For %F both the RMSE on training and test set are steadily decreasing during the studied 100 generations. For %PPB the RMSE on the training set steadily decreases during the whole evolution, while the RMSE on the test set is decreasing until generation 50, and then increasing until generation 500, also showing some oscillations. We conclude that GP has a

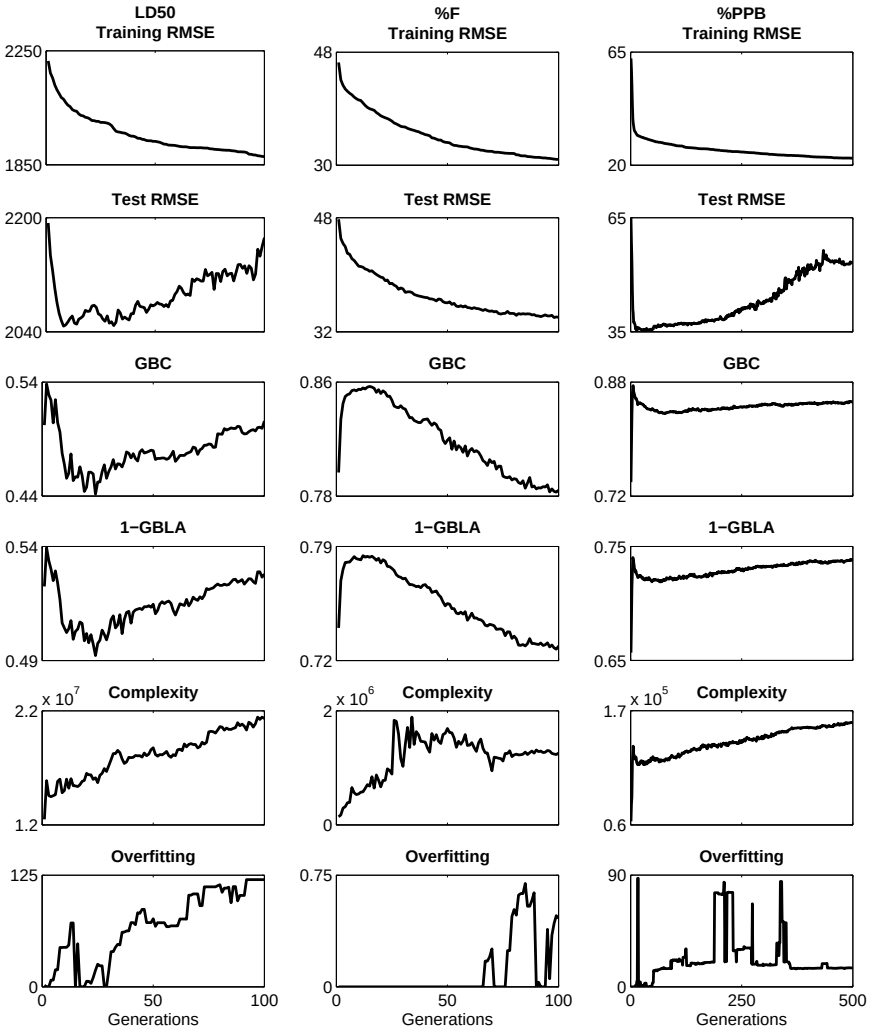


Figure 6: The first (respectively second and third) column reports the results for LD50 (respectively %F and %PPB). For each column, from top to bottom, we report the RMSE on the training set, the RMSE on the test set, the GBC, 1-GBLA and the values of the complexity and overfitting measures introduced in [203]. All these results are reported against generations and they are medians of the value assumed by the best individual (i.e. the one with the best RMSE on the training set) over 120 independent runs.

worse generalization ability for LD50 and %PPB than for %F. Let us now focus on the curves representing GBC and 1-GBLA. Both GBC and 1-GBLA are increasing after generation 30 for LD50, steadily decreasing (except for the initial part of the run) for %F and increasing (except for the first 50 generations) for %PPB. These results hint a relationship between the trend of the RMSE on the test set and the GBC and GBLA measures for all the studied problems. In particular, GBC seems to have a positive correlation with the RMSE on the test set and GBLA seems to have a negative correlation with the RMSE on the test set. Before studying in details these correlations, let us

first look at the trend of the complexity and overfitting measures. The complexity measure seems to have a less clear relationship with the RMSE on the test than GBC and GBLA for LD50 and %F. On the other hand, for %PPB the complexity measure seems to be growing with a higher speed than GBC and 1-GBLA, and thus it seems to have a stronger correlation with the RMSE on the test set. Finally, we point out that the overfitting measure has a more oscillating and less regular behavior than the other measures. Nevertheless, its general trend seems related to GBC and to the complexity measure for LD50 and to GBC, GBLA and complexity for %PPB. On the other hand, no clear relationship appears between the overfitting measure and any of the other measures for %F.

In order to better understand the mutual relationships between the quantities plotted in Figure 6, in Figure 7 we report the median values of the cross correlation at delay zero between them. Cross correlation is a standard method of estimating the degree to which two series $S_1 = \{x_1, x_2, ..., x_n\}$ and $S_2 = \{y_1, y_2, ..., y_n\}$ are correlated. It is defined by $r = (\sum_{i=d+1}^{n} |(x_i - \overline{S_1})(y_{i-d} - \overline{S_2})|)/(\sigma_{S_1}\sigma_{S_2})$, where $\overline{S_1}$ and $\overline{S_2}$ are the averages of the elements in the series S_1 and S_2 respectively, σ_{S_1} and σ_{S_2} are their respective standard deviations, and d is the delay (in this work, we have used $d = 0$). An introduction to cross correlation can be found, for instance, in [50]. Assume we want to calculate the correlation between GBC and RMSE on the test set, as it is the case, for instance, in the left-top plot of Figure 7. This can be done by considering as S_1 the series composed by the GBC values of the best individuals in the population at each generation and as S_2 the series of the respective RMSE values on the test set of the same individuals. Applying the same method, it is possible to calculate the cross correlation of any pair of measures reported in Figure 6. In Figure 7, we also draw two horizontal lines at the values -0.15 and 0.15, often empirically identified as thresholds between the presence of a correlation (either positive or negative) and the absence of it.

For LD50: during the whole run, GBC has a positive cross correlation with the RMSE on the test set and overfitting and a negative cross correlation with the RMSE on the training set after generation 40. GBLA has a negative cross correlation with the RMSE on the test set after generation 50, a positive cross correlation with the RMSE on the training set and cross correlation approximately equal to zero with overfitting. Complexity has a positive cross correlation with overfitting and a negative cross correlation with the RMSE on the training set. The cross correlation between complexity and RMSE on the test set is approximately equal to zero, except for some small oscillations at the end of the run.

For %F: GBC has a positive cross correlation with RMSE both on the training and test set (in particular after generation 50 and this value is steadily increasing during the run) and a cross correlation

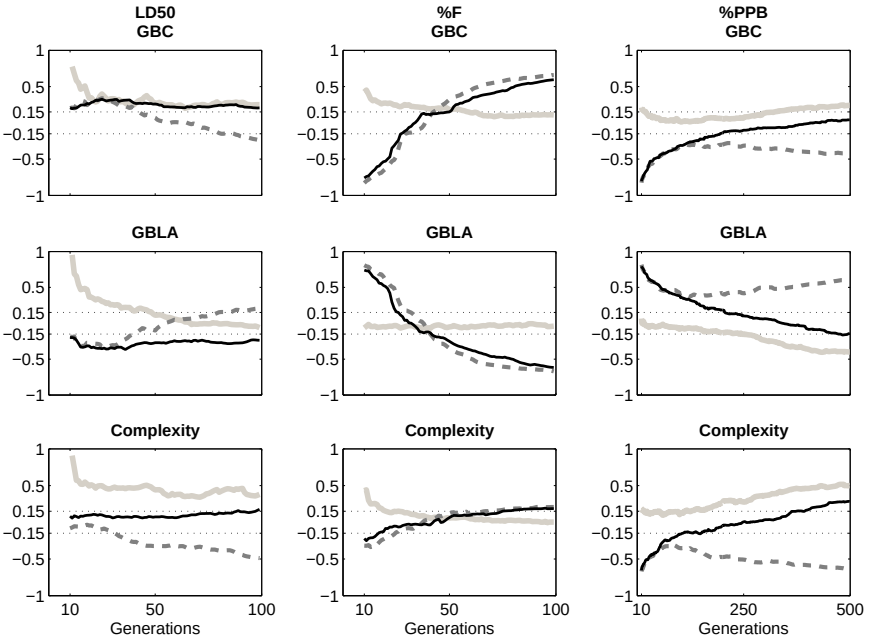


Figure 7: The first (respectively second and third) column reports the results for LD50 (respectively %F and %PPB). For each column, from top to bottom, we report the cross correlation of GBC, GBLA and complexity with the RMSE on the training set (dashed dark grey line), the RMSE on the test set (black line) and the overfitting measure (solid light grey line). All these results are reported against generations and they are medians over 120 independent runs.

approximately equal to zero with overfitting. GBLA has a negative (and steadily decreasing) cross correlation both with the RMSE on the training and on the test set and a cross correlation approximately equal to zero with overfitting. Finally, complexity has a cross correlation approximately equal to zero with overfitting and a positive cross corrlation with both RMSE on the training and test set (even though in both cases the cross correlation becomes larger than 0.15 only in the final part of the run). One general interesting thing to be remarked is also that in some cases the the cross correlations seem, so to say, to "lag" the raw data; e.g., in Figure 6, for %F, GBC appears to be strongly correlated with test RMSE after generation 10-20, but in Figure 7 the cross correlation is not positive until generation 50. For this reason we are planning to investigate other measures instead of cross correlation in the future.

For %PPB: the value of the cross correlation between GBC and the RMSE on the training and test set is negative, but steadily increasing, in the first 100 generations. For this reason, we have executed the simulations until generation 500, to see if some of these correlations became positive later in the evolution. We can see that the correlation between GBC and the RMSE on the test set becomes positive more or less at generation 350, and it keeps on growing, although

without becoming larger than 0.15. Because of the steadily growing trend of the curve of the cross correlation between GBC and RMSE on the test set, we hypothesize that this cross correlation would become positive later in the run. On the other hand, the cross correlation between GBC and RMSE on the training set is clearly negative during the whole run. Finally, the cross correlation between GBC and overfitting is positive (it becomes larger than 0.15 around generation 280, and it remains larger than 0.15 until the end of the run). GBLA has a negative cross correlation with both overfitting and RMSE on the test set and a positive cross correlation with RMSE on the training set. Finally, complexity has a positive cross correlation with both overfitting and RMSE on the test set and a negative cross correlation with RMSE on the training set.

Summarizing: GBC is positively correlated with the RMSE on the test set and GBLA is negatively correlated with the RMSE on the test set. These facts seem independent on the generalization ability of GP (i.e. they hold for all the studied problems). Furthermore, the magnitude of the correlation with the RMSE on the test set is larger for GBC and GBLA than for the complexity measure for all studied problems except %PPB. The negative values of the correlation between GBLA and RMSE on the test set can be interpreted as follows: GBLA quantifies the ability of GP to learn the "difficult" training points. It is intuitive that a good learning of those points leads GP to a poor generalization ability, because the solutions are too specialized on training data and thus overfit them. This can be caused, for instance, by the fact that those "difficult" points correspond to "noise", or even errors in the training data, or they are generally not useful to reliably reconstruct the target function. The existing relationship between GBC and GBLA with the RMSE on the test set seems to hint that the ideas used to define GBC and GBLA could be useful to build a new fitness function able to reduce the error on the test set. This is the goal of the next paragraph.

Experimental results: New Fitness Function. Using either GBC or GBLA as new fitness functions does not allow us to obtain interesting results. Consider, for instance, the case of GBLA: each function able to learn some particular points (the ones that are not considered as difficult) would have a good fitness, and thus it would receive a high probability of surviving and mating in the GP population, independently of the distance of that function from the target one. Besides our intuition, also a set of preliminary experimental results confirm that using either GBC or GBLA as fitness functions does not allow us to obtain better results than standard GP (i.e. GP that uses the RMSE as fitness) on the test set. Nevertheless, the ideas used to define GBC and GBLA can be used to define a new fitness function, assuming to integrate them with the error between learned values and target ones. A possibility could be to use them together with RMSE in a

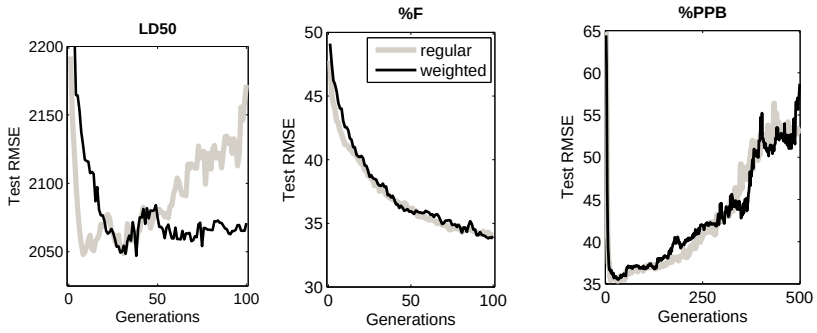


Figure 8: RMSE on the test set of standard GP (i.e. GP that uses the RMSE as fitness, indicated by "regular" in figure) and of GP that uses the new fitness function (indicated by "weighted" in the figure). Left: LD50. Middle: %F. Right: %PPB.

multi-objective method. Even though the idea is interesting, we want to define *one* new fitness function able to incorporate both the information derived from the RMSE and from the new measures.

The idea is to give a *weight* to the error in each training point. For this reason, we call the new fitness function "weighted_fitness". The weight should depend on how rugged the learned function is in that point, reducing the weight of the rugged points. The new fitness measure is:

$$\text{weighted\_fitness}(g) = \sum_{i=0}^{m} \frac{(f(\mathbf{x_i}) - g(\mathbf{x_i}))^2}{1 + |V(\mathbf{x_i})|}$$

where $\mathbf{x_1}$, $\mathbf{x_2}$, ..., $\mathbf{x_m}$ are the training points, g is a GP individual, the values $f(\mathbf{x_1})$, $f(\mathbf{x_2})$, ..., $f(\mathbf{x_m})$ represent the targets on those points. The fact that the denominator in the equation of the new fitness is $1 + |V(\mathbf{x_i})|$ instead of $|V(\mathbf{x_i})|$ is due to the fact that the value of $|V(\mathbf{x_i})|$ could be equal to zero. In Figure 8 we report an experimental comparison between standard GP and GP that uses the new fitness function The same experimental settings and number of runs used for the initial analysis are used also here. We report the median of the RMSE on test data for each performed generation for both these models. For LD50, GP that uses the new fitness function is able to obtain better results. For both %F and %PPB, GP using the new fitness function seems to return very similar results than standard GP. We conclude that GP using the proposed fitness function is able to better generalize (compared to standard GP) for some problems where standard GP has a poor generalization ability (as it is the case of the LD50), while it behaves comparably to standard GP when standard GP itself has a good generalization ability (like for %F). Nevertheless, problems where standard GP has a poor generalization ability and the new fitness function is not able to improve it exist (it is the case of %PPB). But at least, we have shown that in this last case, the new fitness function does not worsen the results. These results suggest that the

proposed fitness function could be a suitable one, given that in some cases it gives an advantage when standard GP has poor generalization, and when it doesn't, at least, it does not give any disadvantage. Furthermore, the new fitness function is simple to implement and computationally reasonable (we do not report statistics about the execution time for lack of space).

6.5 FURTHER REMARKS

A study of Genetic Programming (GP) learning ability has been presented, offering the three following contributions: first, we have defined a new measure (GBC) to quantify the functional complexity of GP individuals. Compared to another complexity measure defined in [203], GBC is rotationally invariant and it has a higher correlation with the quality of GP solutions on out-of-sample data. Secondly, we have presented a new measure (GBLA) aimed at quantifying the ability of GP to learn "difficult" points and we have shown that this measure is negatively correlated with the quality of GP solutions on out-of-sample data. Based on these ideas, the third contribution consisted in defining a new fitness function for GP. Experimental results have shown that this new fitness function allows GP to better generalize for some problems where standard GP has a poor generalization, without worsening the results in all other cases. This seems to indicate the suitability of this fitness function in any possible case. However, the new fitness function has to be further studied in the future.

7

BENCHMARKING: THE K-LANDSCAPES

7.1 INTRODUCTION

The work presented in this chapter extends the NK landscapes problem to GP. The new GP benchmark that we introduce is called K landscapes[1]. Its definition, together with a first experimental study aimed at showing the GP behavior on this benchmark, is contained in the continuation of the chapter.

The chapter is chapter as follows: in Section 7.2 we describe the K landscapes benchmark for GP. In Section 7.3, we present our experiments, including both a description of the used experimental settings and a discussion of the obtained results. Finally, Section 7.4 provides some further remarks and directions of research.

7.2 THE K LANDSCAPES FOR GP

We denote a GP individual by T, the root of T by $R(T)$, the depth of T by $D(T)$ and the set of children of a node N by $S(N)$. The set of all nodes of T is denoted by $N(T)$. The set of all possible subtrees of T is denoted by $\Psi(T)$. The set of all functional symbols used to code individuals is denoted by $\mathcal{F}$ and the set of terminal symbols by $\mathcal{T}$.

Given two numbers $a, b \in \mathbb{R}$, with $a < b$, let $\nu : \mathcal{F} \cup \mathcal{T} \mapsto [a, b]$ be a function chosen randomly between all (representable) functions from $\mathcal{F} \cup \mathcal{T}$ to $[a, b]$. The map ν returns a number in $[a, b]$ for each possible tree node, i.e. for each possible element of the set $\mathcal{F} \cup \mathcal{T}$. Also, given two numbers $c, d \in \mathbb{R}$ such that $c < d$, let $w : \mathcal{F} \times (\mathcal{F} \cup \mathcal{T}) \mapsto [c, d]$ be a randomly chosen function that returns a number in $[c, d]$ for each possible connection between two nodes in a tree. Here we choose $a = -1$, $b = 1$, $c = 0$ and $d = 1$ and from now on all the considerations will be done assuming these values for a, b, c and d. Choosing a negative value for a, we give to the ν function the possibility of returning negative values. For a given $K \in \mathbb{N}$, with K smaller or equal to the maximum admissible depth for the trees in

1 The "N" in the NK landscapes model represents the length of the genome codifying GAs individuals. This does not make sense in GP, where individuals have typically a variable sized representation. For this reason, we just use the term "K" landscapes to indicate the proposed benchmark.

the population (if any maximum depth is imposed) and a tree T, we define:

$$f_K(T) = \begin{cases} v(R(T)) + \sum_{C \in S(R(T))} (1 + w(R(T), C)) f_{K-1}(C) & \\ & \text{if } K > 0 \\ v(R(T)) & \text{otherwise} \end{cases}$$

When $K = 0$, $f_K(T)$ simply returns the value of the v function on the root of T. In all the other cases, $f_K(T)$ returns a weighted sum of the values of v applied to the nodes of T for the first K levels, where the weights are determined by the values of w.

The fitness function that we propose for the GP K landscapes is defined as the maximum value of f_K calculated over all the nodes of a GP tree, with a penalty given by a function of the difference between the depth of the tree and K:

$$F_K(T) = \frac{1}{1 + |K - D(T)|} \max_{T' \in \Psi(T)} \{f_K(T')\}$$

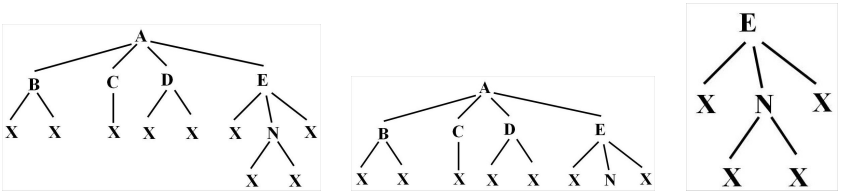
The problem that we define introducing this fitness function is a maximization one (i.e. larger values of the fitness are better).

Now we want to enunciate and prove some properties of the K landscapes problem defined by such a fitness function. But before doing it, we need to introduce the following concept:

Definition 7.2.1. Given a tree T and a $K \in \mathbb{N}$, we indicate with the term *summit* of T of level K a structure T′ that respects the following properties:

- if $K = 0$, T′ is only composed by the node $R(T)$.
- otherwise, $R(T') = R(T)$ and if the subtrees of $R(T)$ in T are $S_1, S_2, ..., S_h$, then the subtrees of $R(T')$ in T′ are $U_1, U_2, ..., U_h$, where for each $i = 1, 2, ..., h$ U_i is a summit of level $K - 1$ of S_i if S_i is different from the empty tree, and the empty tree otherwise.

The concept of summit of a tree should be clarified by Figure 9: the structure in Figure 9(b) is a summit of the tree in Figure 9(a), while the one in Figure 9(c), even though it is a subtree of the tree in Figure 9(a), is not a summit of the tree in Figure 9(a). In very informal terms, we could say that the summit of a tree T is a "top part" of T: for each path from the root to a leaf in T, a summit of T contains a subset of this path, from the root to a given node up to a given prefixed level K, or the whole path if the length of the path is $\leqslant K$. It is also worth pointing out that a summit of level K of a tree T does not necessarily have a depth equal to K, but can also have a smaller depth than K if it is the whole tree T. Furthermore, it is immediate to



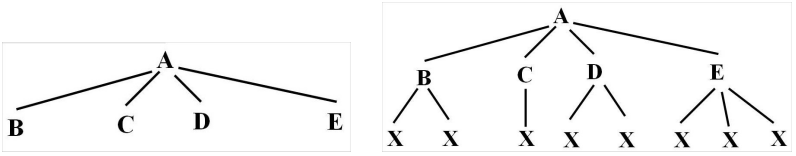
(a) (b) (c)

Figure 9: The structure in figure (b) is a summit of the tree in figure (a), while the one in figure (c) is not a summit of the tree in figure (a).

remark that, if T is the representation of a GP individual, a summit of T may not represent any GP individual, given that it may lack some terminals (indeed, it can represent an individual only if it is equal to T itself). For instance, if we indicate with X a general terminal symbol, the summit in Figure 9(b) contains one branch that is not complete (it is not ended by a terminal symbol), and thus it cannot represent a GP individual. It is possible to "transform" a summit of a tree into a GP individual by adding terminal symbols at the last level, where needed. We call this process *embedding*[2]:

Definition 7.2.2. Given a summit T′ of a tree T, an *embedding* of T′ is a tree that is identical to T′ except for the fact that terminal symbols are added in the branches that are not ended by terminal symbols in T′.

Figure 10 should clarify the concept of embedding: the tree in Figure 10(b) is an embedding of the summit in Figure 10(a), given that a terminal symbol is added in all the branches that are not ended by a terminal symbol.



(a) (b)

Figure 10: The tree in figure (b) is an embedding of the summit in figure (a).

We are now ready to enunciate and prove some properties of the proposed K landscapes.

2 An alternative, and maybe more suitable, term may be "extension", since with this process a tree is "extended" rather than "embedded" in another structure. Nevertheless, given the generality of the term "extension", we prefer to use the term "embedding".

Proposition 7.2.1. *For each admissible value of* K, *if there exists a tree* T *such that* $f_K(T) > 0$ *then all the optima in the search space must necessarily have a depth of at most* $K + 1$.

Proof. Let us consider the definition of $f_K(T)$. It is composed by a sum of terms. To calculate each one of these terms, the nodes of T have to be considered from the root up to the level K, or from the root to the leaf for the paths where the leaf is at depth smaller than K. Let us consider the structure composed by all the nodes that have to be analyzed in order to calculate $f_K(T)$. It is clearly a summit of level K of T. This summit can be transformed into a tree T': if the summit is equal to T, then $T' = T$. Otherwise T' is an embedding of T. In the first case T' has a depth of at most K, in the second case T' has a depth equal to $K + 1$.

It is interesting to remark that, in the first case, if the depth of T' is smaller than K, a tree T'' of depth K can be considered that has T' as one of its subtrees. By the definition of F_K, this new tree will have a better fitness value than T'. In fact, given that T' is a subtree of T'', we have that $\max_{S_1 \in \Psi(T'')}\{f_K(S_1)\} \geqslant \max_{S_2 \in \Psi(T')}\{f_K(S_2)\}$ and the term $\frac{1}{1+|K-D(T'')|}$ is equal to 1, while $\frac{1}{1+|K-D(T')|}$ is smaller than 1. Thus, we can say that for each tree T it is possible to generate a tree U that either has a depth equal to K or to $K + 1$: depending on the depth of the summit of T of level K, U can be equal to T itself, an embedding of its summit or a tree (like T'') that has its summit as a subtree.

Now we want to show that each tree that has a depth larger than $K + 1$ cannot be an optimum. Let T be a tree with a larger depth than $K + 1$. Let us consider the tree U obtained from T by applying the process described so far. Given that $f_K(T)$ is calculated using only the nodes that belong to the summit of level K of the subtrees of T, and given that the subtrees of T and the subtrees of U that maximize f_K have the same summit of level K, we have that $\max_{S_1 \in \Psi(T)}\{f_K(S_1)\} = \max_{S_2 \in \Psi(U)}\{f_K(S_2)\}$. But given that T has a larger depth than U and the depth of U is either K or $K + 1$, $\frac{1}{1+|K-D(T)|} < \frac{1}{1+|K-D(U)|}$. Thus the fitness of T must be smaller than the one of U. We conclude that T cannot be an optimum. □

Proposition 7.2.2. *For each* $K > 0$, *we have that* $f_K(T) > 0$ *for some tree* T *if and only if there exists* $N \in \mathcal{F} \cup \mathcal{T}$ *such that* $\nu(N) > 0$.

Proof. To prove this property, we prove separately both implications. First, let us prove that if $f_K(T) > 0$ for some tree T then $N \in \mathcal{F} \cup \mathcal{T}$ such that $\nu(N) > 0$ exists. The case $K = 0$ is obvious: if $f_K(T) > 0$ then ν applied to the root of T must return a positive value. In the other cases, $f_K(T) > 0$ implies that at least one of the terms of the sum that defines f_K must be positive. f_K is defined as a sum of terms, where each term is composed by an application of function ν to a node of T multiplied by an application of function w to a connection in T. Given

that w can only return positive values, there must exist at least one $N \in N(T) \subseteq \mathcal{F} \cup \mathcal{T}$ such that $\nu(N) > 0$.

Let us now prove that if an $N \in \mathcal{F} \cup \mathcal{T}$ such that $\nu(N) > 0$ exists, then $f_K(T) > 0$ for some tree T. Let us assume first that there exists $N \in \mathcal{T}$ such that $\nu(N) > 0$. Then the tree having N as the only node has a value of f_K equal to $\nu(N) > 0$. Now let us assume that $N \in \mathcal{F}$ such that $\nu(N) > 0$ exists and let us consider a tree T of depth $K+1$ with all the leaves at level $K+1$ and all the internal nodes equal to N. The summit T' of level K rooted at $R(T)$ is composed only of nodes with weight $\nu(N)$. This means that $f_K(T) = f_K(T') \geqslant \nu(R(T)) > 0$. □

We also remark that, when $f_K(T) \leqslant 0$ for all trees, and the tree depth is unbounded, for every tree it is always possible to find a tree with an higher fitness, even if this fitness is always negative. This is due to the penalty given by the term $\frac{1}{1+|K-D(T)|}$ that appears in the definition of F_K: as $D(T)$ increases, a negative value of the fitness increases too.

In our definition of the K landscapes, we want all the global optima to have a depth not larger than $K+1$, because, as it will be clear shortly, this allows us to define an algorithm to compute the globally optimal fitness. Thus, for obtaining this goal, from now on we impose that the ν function must assume a positive value for at least one element of its domain.

We are now ready to enunciate and prove the following property:

Proposition 7.2.3. *At least one of the optima of the* K *landscapes is a tree* T *that has all the nodes at the same level identical to each other.*

The tree represented in Figure 11 is an example of a tree that respects the property of Proposition 7.2.3. In fact, level 1 is composed by only B symbols, level 2 is composed by only C symbols and level 3 is composed by only X symbols; thus only symbols that are identical to each other appear in the same level.

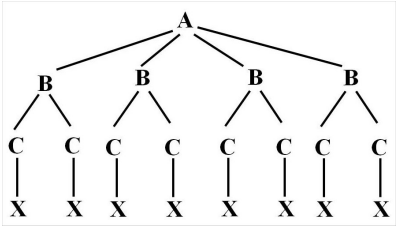


Figure 11: An example of a tree that respects the property of Proposition 7.2.3.

Now we prove Proposition 7.2.3:

Proof. Let us consider an optimum T for the K landscapes. Let $\Psi(T) = \{U_1, U_2, ..., U_h\}$ be the set of all the possible subtrees of T. For each

$i = 1, 2, ..., h$, let U'_i be the tree of depth at most $K + 1$ obtained from U_i by considering its summit of level K and eventually embedding it (the process that allows us to generate U'_i from U_i is explained at the beginning of the proof of Proposition 7.2.1). Now, among the trees $U'_1, U'_2, ..., U'_h$, let us consider the one that maximizes f_K and let us call it T' (actually the nodes that compose the summit of level K of T' are the ones that are used to calculate the fitness of T). We call T' an f_K-optimum.

It is easy to convince oneself that Proposition 7.2.3 holds by looking at Figure 12: let us assume that the tree in Figure 12(a) is f_K-optimum. Let T_1 be the left subtree of the tree in Figure 12(a) and T_2 be its right subtree (T_1 and T_2 are the subtrees respectively surrounded by a rectangle and an oval in Figure 12(a)). First of all, we remark that the property $f_K(T_1) = f_K(T_2)$ must hold. In fact, (without loss of generality) let $f_K(T_1)$ be larger than $f_K(T_2)$. If this is true, it is possible to build a new tree that is identical to the one in Figure 12(a), but with the only difference that the occurrence of T_2 is replaced by another occurrence of T_1; this new tree, represented in Figure 12(b), would have a larger value of f_K than the tree in Figure 12(a). But this would contradict the hypothesis that the tree in Figure 12(a) is f_K-optimum. We conclude that $f_K(T_1) = f_K(T_2)$ and also that the trees represented in Figures 12(b) and (c) have the same value of f_K as the one in Figure 12(a), and thus are also f_K-optima.

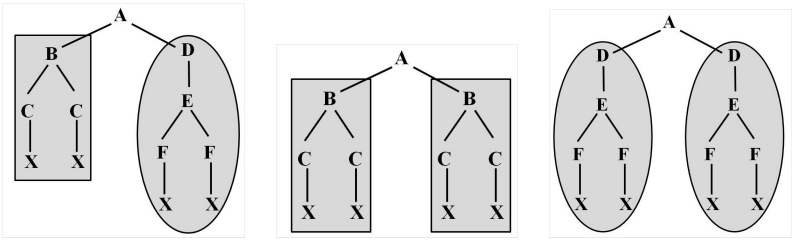


Figure 12: Suppose that (a) is a f_K-optimum. Then both (b) and (c) must also be f_K-optima (see the proof of Proposition 7.2.3).

This argument can be generalized, abstracting from the example of the trees represented in Figure 12: given an f_K-optimum, it is always possible to build another f_K-optimum that has all the subtrees of its root identical to each other (simply by "replicating" one of them, as it has been done for transforming the tree in Figure 12(a) into the tree in Figure 12(b)). In case these subtrees do not respect the property of Proposition 7.2.3, it is possible to iterate the process, transforming them into trees that have all the subtrees of the root identical to each other and that have the same value of f_K. Iterating this process until the last level of the tree, it is possible to build an f_K-optimum that respects the property of Proposition 7.2.3.

So far we have shown that, for every K, there exists an f_K-optimum that respects the property of Proposition 7.2.3. Now, we have two possibilities to define a candidate optimum for the K landscapes problem: (1) consider an f_K-optimum chosen between the trees of depth at most K. If it has depth K, then it is the candidate optimum. Otherwise, if its depth is smaller than K, we candidate a tree of depth K that respects the property of Proposition 7.2.3 and contains it as a subtree; (2) consider an f_K-optimum chosen between the trees of depth $K+1$ and that respects the property of Proposition 7.2.3. One optimum T of the K landscapes problem can be obtained either by possibility (1) or by possibility (2), because in both cases T contains an f_K-optimum and the value of the term $\frac{1}{1+|K-D(T)|}$ is either 1 or $\frac{1}{2}$. In both cases, it respects the property of Proposition 7.2.3 and this allows us to conclude. □

So far, we have proven some properties that at least one optimum of the proposed K landscapes must have. Using these properties, we are now able to calculate the optimal fitness of this problem. The argument that we use is similar to the one used in the proof of Proposition 7.2.3. Let $T_{\leqslant K}$ be an f_K-optimum of depth at most K. Let T_{K+1} be an f_K-optimum of depth $K+1$. To find an optimum for F_K we can suppose that both $T_{\leqslant K}$ and T_{K+1} respect the property of Proposition 7.2.3. Also, it is possible to consider $T_{\leqslant K}$ of depth K since, if its depth is less than K, it is possible to define a deeper tree that has $T_{\leqslant K}$ as a subtree. Thus, by the definition of F_K, it is possible to express the K landscapes optimal fitness as follows:

Proposition 7.2.4. *The optimal fitness of the* K *landscapes is:*

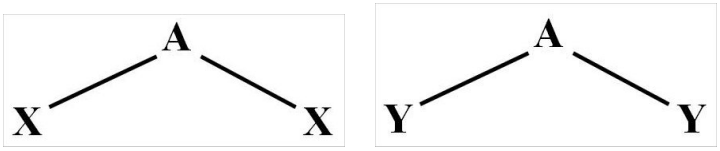
$$\begin{aligned} F_{opt} &= \max\{\tfrac{1}{1+|K-K|} f_K\left(T_{\leqslant K}\right), \tfrac{1}{1+|K-(K+1)|} f_k\left(T_{K+1}\right)\} \\ &= \max\{f_K\left(T_{\leqslant K}\right), \tfrac{1}{2} f_K\left(T_{K+1}\right)\} \end{aligned}$$

Furthermore, we are also able to define an algorithm to compute the optimal fitness and the structure of at least one of the optima: for all the tree depths form 0 to $K+1$, we simply exhaustively consider all the trees that respect the property of Proposition 7.2.3 (the number of these trees is much smaller than the number of all the trees of depth at most $K+1$), finding the one with the maximum fitness. Because of the properties that we have proven so far, we are sure that it will be one of the global optima. Even if the algorithm is exponential in K, it is usable in practice because K cannot be larger than the maximum depth allowed for the trees in the population. Thus, only "limited" values of K are used in practice.

We finally point out an interesting characteristic of the K landscapes problem introduced here: given a f_K-optimum of depth d, its subtrees of depth $d-1$ are *not necessarily* f_{K-1}-optima. This fact implies that it is difficult for GP to build an optimum starting from smaller sub-optima, as it is the case of the NK landscapes for GAs. The following

example shows a case where the subtrees of depth $d-1$ of a f_K-optimum of depth d are not f_{K-1}-optima.

Example. Let us consider the following set of functional and terminal symbols used to build GP individuals: $\mathcal{F} = \{A\}$, $\mathcal{T} = \{X, Y\}$. Let us consider the following values of the v and w functions: $v(A) = 1$, $v(X) = 0.75$, $v(Y) = 1$; $w(A, X) = 1$, $w(A, Y) = 0$. It is clear that it is possible to build only two trees of depth 1 that respect the property of Proposition 7.2.3. They are shown in Figure 13. Let T_1 be the tree



(a) (b)

Figure 13: The only two trees of depth 1 that respect the property of Proposition 7.2.3 and that can be built using the sets of symbols $\mathcal{F} = \{A\}$, $\mathcal{T} = \{X, Y\}$.

represented in Figure 13(a) and T_2 the one represented in Figure 13(b). The value of f_1 for these two trees is: $f_1(T_1) = 1 + 2 * (1 + 1) * 0.75 = 1 + 3 = 4$ and $f_1(T_2) = 1 + 2 * 1 * 1 = 3$. Thus, T_1 is an f_1-optimum. But, given that $v(Y) > v(X)$ the f_0-optimum is not the tree composed by the only X symbol (subtrees of depth 0 of T_1), but the one composed by the only Y symbol.

7.3 EXPERIMENTAL STUDY

Experimental Settings. All the results presented in this section have been obtained performing 100 runs of GP with a population size of 100 individuals running for 100 generations. The initialization method was ramped half-and-half with an initial maximum tree depth of 2. The selection method used was tournament selection with a tournament size of 10. The crossover method used was the standard subtree swapping crossover [114] with a crossover probability of 0.9. The mutation method used was point mutation [114] with a probability of 0.1. During the experiments elitism (i.e. unchanged copy of the best individual in the next population at each generation) was used and the maximum allowed tree depth was 17. The experiments were performed using values for the K parameter ranging from 2 to 13. Individuals have been built using 2 functional symbols, both with arity 2, and 4 terminal symbols. Since the fitness is computed considering only the values of the v and w functions, it was unnecessary to define a semantic for the functional and terminal symbols used. At each generation the fitness, size and depth of the best individual were recorded. Also the average fitness, depth and size of the population

were recorded. The median of these values over the 100 runs is reported in this section. A t-test has been performed over the fitness values obtained in the last generation for every value of K with the equality of the means considered as the null hypothesis.

Experimental Results. In all the figures from 15 to 19, different curves represent different values of K. For the same value of K we have used, in all these figures, the same line. For this reason, we report the legend of this figures only once, in Figure 14.

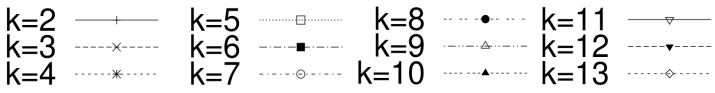


Figure 14: The legend for the plots reported in figures from 15 to 19.

In Figure 15 the median of the average depth of the GP trees in the population for the different values of K is presented. We can see that, for every considered K, the average depth after generation 20 is slightly above K. This result shows that the largest part of the search process is concentrated on trees of a depth near K. This means that the region of the search space that contains the individuals with the best fitness increases exponentially with K. This can be considered as a first hint of the fact that the problems get more difficult as K increases. It is also interesting to remark that the average size rises quickly from the initial value. This means that the penalization of individuals with depth different from K is effective in forcing the search process to sample trees of depth close to K.

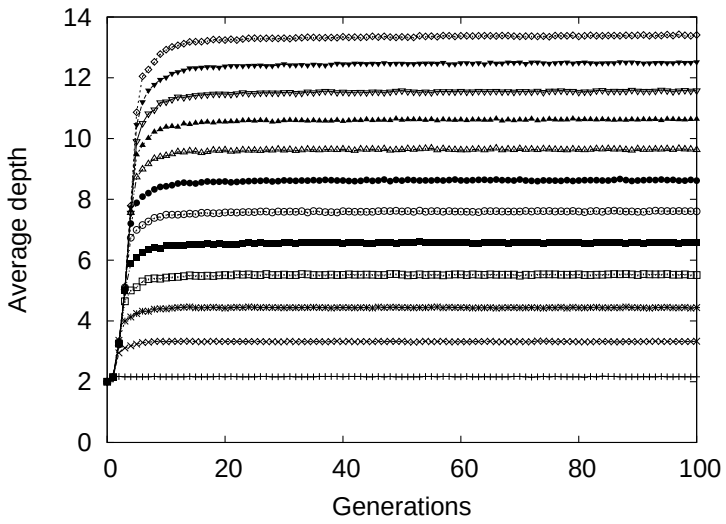


Figure 15: Median of the average depth of the GP trees in the population against generations.

Another indication of the different difficulty of the problem for different values of K is given by the plots of the median of the average number of nodes of the trees in the population, reported in Figure 16. The average number of nodes increases generation by generation for

all the considered values of K. The main difference between the various cases is that, for small values of K, the increase in size stops after some generations, while for high values of K the process continues up to the last studied generation. This curves show that even if the search process is quickly directed towards the "right tree depth", obtaining the optimum is still a process that increases in difficulty with K. In other words, while reaching the depth of the optimal tree is easy, producing an optimal tree of a given depth remains a difficult process for GP.

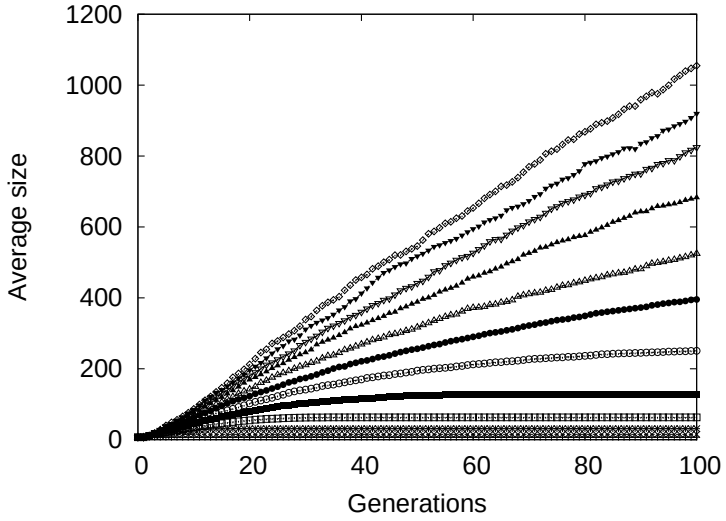


Figure 16: Median of the average number of nodes of the GP trees in the population against generations.

Given this information on the average size and the depth of the trees, it is interesting to explore the same information on the best element of the population. In Figure 17 the median of the depth of the best individual is presented. As it is possible to see, for each studied value of K, the depth quickly rises to the corresponding value of K and remains constant for all the considered generations. This is consistent with the previous observations and shows that the best individual in the population after few generations has a depth that is close to the depth of at least one global optimum.

Figure 18 reports the median of the number of nodes of the best individual at each generation. These results are consistent with the ones of the average number of nodes in the population reported so far. This means that the depth of the best individual, as the depth of many other individuals in the population, quickly rises to K, and then GP concentrates on finding the optimal number of nodes, focusing on the space of trees of depth close to K. Since at least one of the optima is a tree of a depth close to K that respects the property of Proposition 7.2.3, the number of nodes of that optimum increases exponentially with K, making the search process more difficult for high K values.

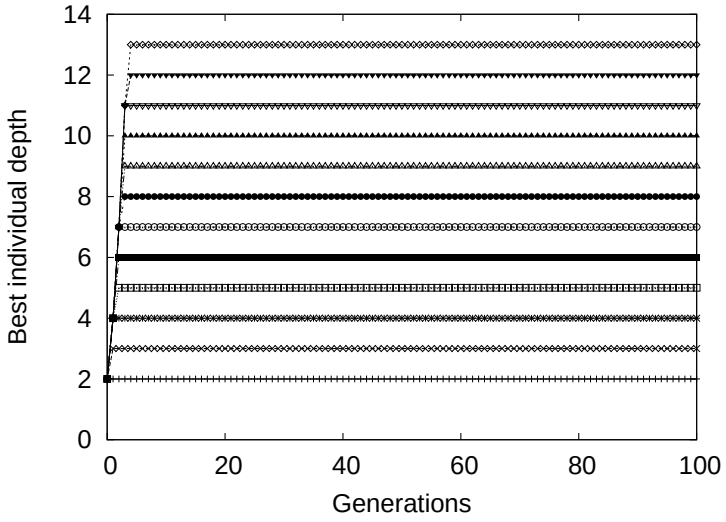


Figure 17: Median of the depth of the best GP tree at each generation.

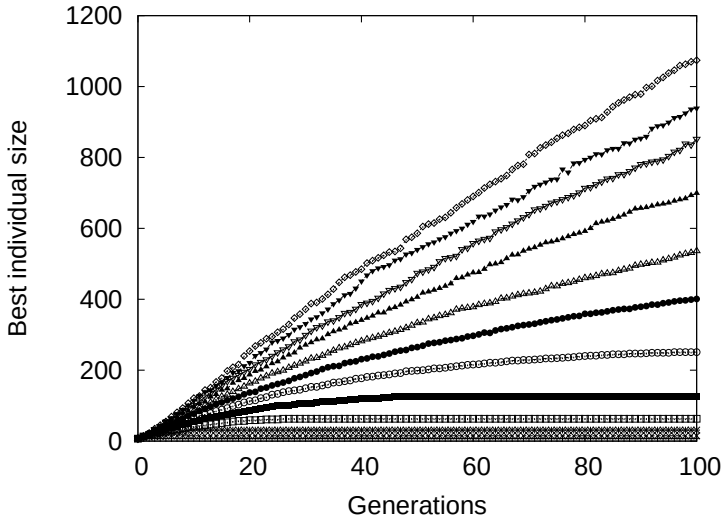


Figure 18: Median of the number of nodes of the best GP tree at each generation.

The fact that the difficulty of the problem increases with K is confirmed by Figure 19, where the median of the best fitness in the population at each generation is reported. Fitness values have been normalized by dividing them by the optimal fitness (that has been calculated using the algorithm reported at the end of Section 7.2). In this way, the global optimum has a normalized fitness equal to 1. The figure clearly shows that the difficulty increases with K, and this is visible since the very first generations. Furthermore, we point out that for $K = 2$, $K = 3$, $K = 4$ and $K = 5$ a global optimum has been reached by GP before generation 100 in the majority of the studied runs, while this is not the case for higher values of K. In particular, for values of K higher than 10 the displacement from a random individual (that has expected fitness equal to 0 because of the uniform random choice of

the ν function) is very low. This indicates that for values of K larger than 10, GP is behaving in a way that is comparable to random search, and the optimization process is extremely slow.

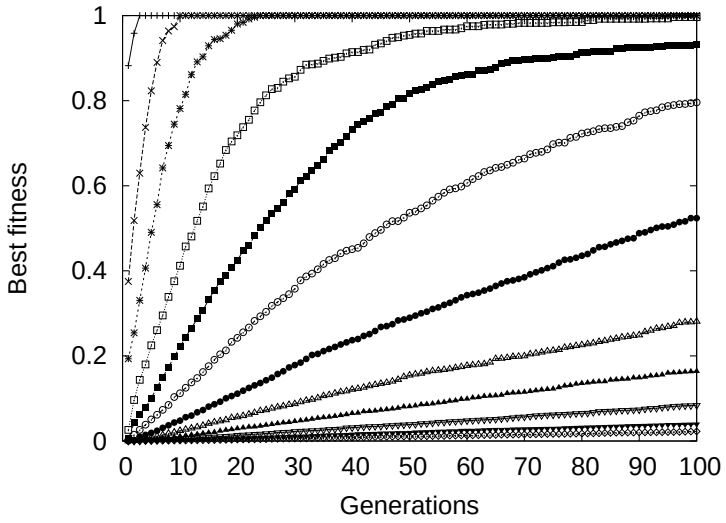


Figure 19: Median of the fitness of the best GP tree in the population at each generation. All fitness values have been normalized dividing them by the optimal fitness.

A more in-depth analysis has been done for the last studied generation of the runs. In the box plot in Figure 20 the average depth of the GP trees at the last generation is considered. It is interesting to note

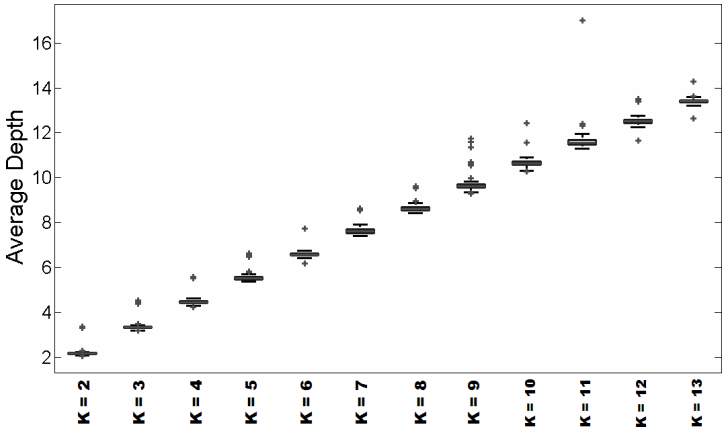


Figure 20: Average depth of the trees in the population at the last generation.

that the variabilities of the measured depths are low. This means that the search almost always remains near the desired value.

The box plot of the average number of nodes at the last generation is presented in Figure 21. It shows that the variability on the number of nodes increases with K. This behavior can be explained by the fact that the number of trees of a given depth increases with K.

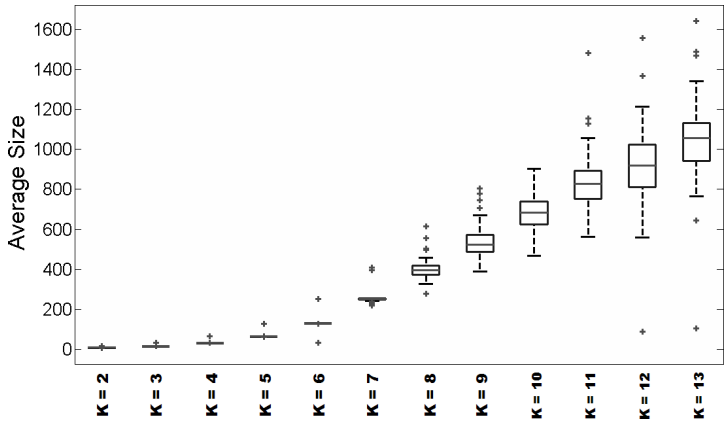


Figure 21: Average number of nodes of the trees in the population at the last generation.

The box plot of the best fitness at the last generation is presented in Figure 22. This figure shows that for different values of K there

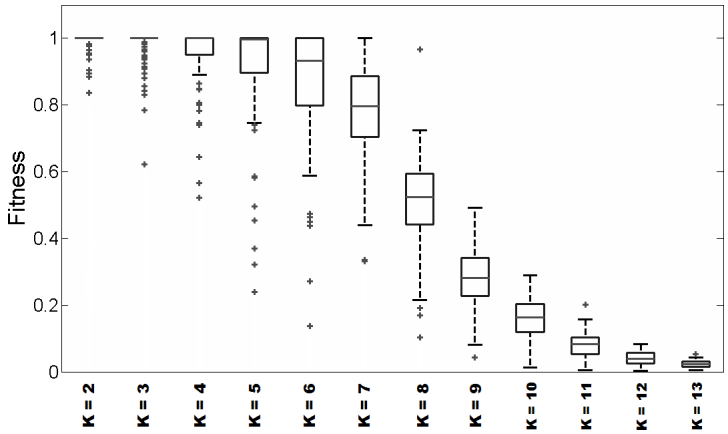


Figure 22: Fitness of the best individual in the population at the last generation.

are different behaviors: for low values of K the fitness remains near 1. For intermediate values of K the fitness decreases steadily when K increases. Finally, for high values of K the fitness still decreases but at a slower rate, since the fitness values approach 0, the expected fitness of a randomly generated GP tree.

We have also performed a t-test over the best fitnesses registered at the last studied generation for the performed 100 runs. The null hypothesis is that two means are identical. In Table 1 the p-values of the t-test are reported for all the different possible combinations of K. The combinations where the p-value is larger than 0.05 are written in italic. The results show that the differences between the recorded fitness values is almost always statistically significant for different values of K.

In conclusion, the presented experiments show that GP is able to find a global optimum in few generations for low values of K; increas-

k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=11	k=12	k=13	
.0193	.0001	.0000	*.1713*	.0000	.0000	.0000	.0000	.0000	.0000	.0000	**k=2**
	.0301	.0002	*.2728*	.0000	.0000	.0000	.0000	.0000	.0000	.0000	**k=3**
		.0365	*.5148*	.0000	.0000	.0000	.0000	.0000	.0000	.0000	**k=4**
			.9650	.0000	.0000	.0000	.0000	.0000	.0000	.0000	**k=5**
				.0115	.0000	.0000	.0000	.0000	.0000	.0000	**k=6**
					.0000	.0000	.0000	.0000	.0000	.0000	**k=7**
						.0000	.0000	.0000	.0000	.0000	**k=8**
							.0000	.0000	.0009	.0001	**k=9**
								.0000	*.2293*	*.0845*	**k=10**
									.7669	*.9045*	**k=11**
										.7645	**k=12**

Table 1: The p-values given by the t-test. The p-values greater that 0.05 are presented in *italic*.

ing K, the best normalized fitness found by GP gets worse until, for large values of K, the behavior of GP becomes comparable to the one of random search. Interestingly, the size of the individuals inside the population increases with K. This allows us to conclude that, for large values of K, we have a progressive code growth without a corresponding improvement in fitness. This is exactly the definition of bloat, as presented, for instance, in [156]. This means that the difficulty of the proposed problem can be effectively tuned by changing the value of K: increasing K we create problems in which GP is more and more unable to optimize and in which GP is more and more affected by bloat.

7.4 FURTHER REMARKS

An extension of the NK landscapes to tree based GP, simply called K landscapes, has been presented. In this benchmark, the epistatic interaction is quantified by the mutual influence on fitness of larger and larger structures in a tree as the value of the K parameter increases. The fact that the hardness of the problem increases with K has been experimentally shown. Furthermore, we have shown that GP produces more bloat as K increases.

A formal proof of the fact that the number of local optima increases with K is needed in the future. Furthermore, we plan to investigate the use of many landscape indicators, like fitness distance correlation, auto-correlation, density of states and many others, on the proposed benchmark, in order to further corroborate the hypothesis that a more and more rugged fitness landscape is induced by increasing K. Finally, we plan to extend the proposed benchmark, overcoming some of its major limitations, namely: (1) the current model does exhibit epistatic behavior under point mutation, but this is no longer true when considering subtree crossover; (2) the current benchmark cannot model dead code: every subtree contributes to the fitness; (3) an ideal solution is very repetitive in terms of used subtrees and this is very far from real world, where it is rarely the case that an ideal solution can be built from cloned and systematically arranged subtrees.

8

FAST SEMANTIC GENETIC PROGRAMMING

In this chapter we propose a new solution to the problem of exponential increase of the size of the trees in Semantic GP. We present a new GP system incorporating an implementation of geometric semantic genetic operators that not only makes them usable in practice, but even very efficient, without requiring any simplification of the individuals during the GP run. In this way, we are for the first time able to exploit the great potentialities of these operators, consisting in the fact that they induce unimodal fitness landscapes.

In order to experimentally validate our new GP system, we have applied it to a complex real-life problem in the field of pharmacokinetic: the prediction of human oral bioavailability of new potential drugs. The results we have obtained have been compared not only to the ones returned by standard GP, but also to the ones of several other state of the art Machine Learning methods reported in [9].

The chapter is organized as follows: Section 8.1 describes the geometric semantic operators, shows that the fitness landscapes induced by them are unimodal and outlines their limitations. In Section 8.2 we present our new GP system that overcomes the current limitations of geometric semantic operators, making them usable (and efficient) for real-life applications. Section 8.3 presents the test problem used, the experimental settings and the obtained results. Finally, Section 8.4 provides some further remarks and directions of research.

8.1 GEOMETRIC SEMANTIC OPERATORS

While the semantically aware methods cited in Section 5.2 often produced superior performances with respect to traditional methods, they are indirect: search operators act on the syntax of the parents to produce offspring, which are successively accepted only if some semantic criterium is satisfied. As reported in [142], this has at least two drawbacks: (i) these implementations are very wasteful as heavily based on trial-and-error; (ii) they do not provide insights on how syntactic and semantic searches relate to each other.

To overcome these drawbacks, in [142], using a formal geometric view on search operators and representations, the authors introduced a novel form of GP that directly searches the space of the underlying

semantics of the programs. This perspective provides new insights on the relation between program syntax and semantics, search operators and fitness landscapes, and allows principled formal design of semantic search operators for different classes of problems.

To explain the idea, let us first consider Genetic Algorithms (GAs), which are similar to GP with the major difference that the solutions are fixed length strings of characters and not computer programs. Let us consider a GA problem in which the target solution is known and the fitness of each individual corresponds to its distance to the target (our reasoning holds for any distance measure used). This problem is easy. In fact, for instance, if we use point mutation, any possible individual different from the global optimum has at least one neighbor (individual resulting from its mutation) that is closer than itself to the target, and thus fitter. So, there are no local optima: the fitness landscape is unimodal. This is also confirmed by the FDC that is clearly equal to 1, because fitness and distance to the goal are identical. Similar considerations hold for many types of crossover, including various kinds of geometric crossover [141].

Now, let us consider the typical GP problem of finding a function that maps sets of input data into known target ones. As already discussed, regression and classification are particular cases. The fitness of an individual for this problem is typically a distance between its calculated values and the target ones (error measure). Now, let us assume that we can find a transformation on the syntax of the individuals, whose effect is a random perturbation of one of their calculated values. In other words, let us assume that we are able to transform an individual G into an individual H whose output is like the output of G, except for one value, that is randomly perturbed. Under this hypothesis, we are able to map the considered GP problem into the GA problem discussed above. So, this transformation would induce a unimodal fitness landscape with FDC equal to 1 and every problem like the considered one (e.g. regressions and classifications) should be easily solvable by GP. The same also holds for transformations on pairs of solutions that correspond to GAs semantic crossovers.

Under this perspective, the objective of [142] was to find operators on the syntactic (or genotypic) space that map well-known operators on the semantic space. Here we report the definition of geometric semantic operators given in [142] for real functions domains, since these are the operators we will use in the experimental phase. For applications that consider other kinds of data, the reader is referred to [142].

Definition 8.1.1. (Geometric Semantic Crossover). Given two parent functions $T_1, T_2 : \mathbb{R}^n \to \mathbb{R}$, the geometric semantic crossover returns the real function:

$$T_{XO} = (T_1 \cdot T_R) + ((1 - T_R) \cdot T_2)$$

where T_R is a random real function whose output values range in the interval $[0, 1]$.

Reference [142] formally proves that this operator corresponds to geometric crossover on the semantic space, and thus induces a unimodal fitness landscape. To constrain T_R in producing values in $[0, 1]$ we use the sigmoid function: $T_R = \frac{1}{1+e^{-T_{rand}}}$ where T_{rand} is a random tree with no constraints on the output values.

Definition 8.1.2. (Geometric Semantic Mutation). Given a parent function $T : \mathbb{R}^n \to \mathbb{R}$, the geometric semantic mutation with mutation step ms returns the real function:

$$T_M = T + ms \cdot (T_{R1} - T_{R2})$$

where T_{R1} and T_{R2} are random real functions.

Reference [142] formally proves that this operator corresponds to a box mutation on the semantic space, and induces a unimodal fitness landscape.

We point out that at every step of one of these operators, offspring contain the complete structure of the parents and one or more random trees as subtrees, plus some arithmetic operators: the size of each offspring is thus clearly much larger than the one of their parents. The exponential growth of the individuals in the population (demonstrated in [142]) makes these operators unusable in practice: after a few generations the population becomes unmanageable and the fitness evaluation process becomes unbearably slow. The solution that is suggested in [142] as a future work consists in performing an automatic simplification step after every generation in which the programs are substituted by (hopefully smaller) semantically equivalent ones. However, this additional step adds to the computational cost of GP and is only a partial solution to the progressive program size growth. Last but not least, according to the particular language used to code individuals, automatic simplification can be a very hard task.

Due to all these limitations, it is important to make an effort in implementing a framework that will allow an efficient use of the geometric semantic operators. The objective is to present a GP implementation that overcomes this limitation, without performing any simplification step and without imposing any particular representation for the individuals (for example the traditional representation of GP individuals as trees can be used). This implementation is presented in the next section. For simplicity, from now on, GP using only geometric semantic crossover and mutation to explore the search space will be indicated as GS-GP (Geometric Semantic GP).

8.2 THE PROPOSED GP IMPLEMENTATION

The implementation we propose can be described by the following steps:

- We create an initial population of (typically random) programs, exactly as in standard GP (let P be the name of this population from now on).
- Given that geometric semantic crossover and mutation need the generation of random trees to be used, we create beforehand all the random trees that will be needed during the whole evolution (this pool of random trees will be called P_{mut} from now on).
- During the first generation, at the moment of evaluating the fitness of the individuals, we create two tables that we call T_{vp} and T_{vm}. T_{vp} (respectively T_{vm}) contains, for each individual in P (respectively in P_{mut}), the value resulting from the evaluation on all fitness cases (in other words, it contains the semantics of that individual). Hence, having a training set with k training instances and P (respectively P_{mut}) containing n (respectively m) individuals, results in a table T_{vp} (respectively T_{vm}) with k rows and n (respectively m) columns.
- For every generation $p > 1$, a new empty table T'_{vp} is created and, whenever a new individual T must be generated by crossover between selected parents T_1 and T_2, the following actions are performed:
  - T is represented by a triple $T = \langle \alpha(T_1), \alpha(T_2), \alpha(R) \rangle$, which is stored in an apposite structure (this structure is called $\mathcal{M}$ from now on), where R is one of the random trees in P_{mut} and, for any tree τ, $\alpha(\tau)$ is a *reference* (or memory pointer) to τ.
  - The first line that is still empty in T'_{vp} is successively filled with the values of the semantics of T, which can be easily obtained by calculating $(T_1 \cdot R) + ((1 - R) \cdot T_2)$ for each fitness case, according to the definition of geometric semantic crossover.
- Analogously, whenever at generation p a new individual T has to be obtained by applying mutation to an individual T_1, the following actions are performed:
  - T is represented by a triple $T = \langle \alpha(T_1), \alpha(R_1), \alpha(R_2) \rangle$ (this triple is also stored in $\mathcal{M}$), where R_1 and R_2 are two among the random trees in P_{mut}.
  - The first line that is still empty in T'_{vp} is this time filled with the values of the semantics of T which can be easily obtained by calculating $T_1 + ms \cdot (R_1 - R_2)$ for each fitness case, according to the definition of geometric semantic mutation.

- When generation p is completed, table T'_{vp} is copied into T_{vp} and erased.
- The process is iterated for the prefixed number of generations.

In synthesis, this algorithm is based on the idea that, when semantic operators are used, an individual can be fully described by its semantics (which makes the syntactic component much less important than in standard GP), a concept discussed in depth in [142]. In order to implement this idea, at every generation we update table T_{vp} by using the values contained in it and the ones in T_{vm}, without explicitly building the syntactic structures of the individuals, but incorporating all the information to do it in a second time.

We point out that:

1. This process of updating table T_{vp} can be performed efficiently and no evaluation of the whole tree is needed anymore. As a consequence, in this implementation the fitness calculation is rather efficient. Indeed the fitness evaluation process requires, for each individual and except for the first generation, a constant time, which is independent from the size of the individual itself. On the other hand, at the initial generation, the fitness evaluation requires a time that is dependent on the size of the individuals, as it is usual in GP.

2. Conceptually, population P evolves during a GP run, while P_{mut} does not change until the end of each run. This is implemented by continuously updating table T_{vp}, while T_{vm} stays unchanged during a run.

3. The structure $\mathcal{M}$ (that contains, for each individual, the triplet of references to the ancestors) increases in size during a GP run. However, given that this structure contains only pointers, we can manage it for several thousands of generations in a very efficient way.

4. The fact that all the random trees used by crossover and mutation are generated in one step before the beginning of the evolutionary process (instead of generating them at the moment they are needed) does not change the expected behaviour of the algorithm (there is no reason to imagine that a random tree should have different properties if generated in two different instants). Nevertheless, they still can be generated and stored in T_{vm} at the moment they are needed, instead of doing it all at once in the beginning, with no significant modification in the behavior of the algorithm.

5. Generating all the random trees that will be needed in one step before starting the evolutionary process and storing them in

P_{mut} is a procedure that can efficiently be managed from a computational viewpoint.

6. Tables T_{vp} and T_{vm} contain the values of the evaluation of the individuals on the fitness cases (i.e. the semantics of the individuals), not their fitness values. This information (and not the fitness) is the one that is needed to reconstruct the semantics of the individuals in the subsequent generations and iterate the process. It is nevertheless easy to calculate the fitness using the semantics and knowing the corresponding target values.

The final part of the algorithm has to be performed after the end of the last generation, in order to reconstruct the individuals. For doing that, we need to "unwind" our compact representation and make the syntax of the individuals explicit. In this way, we will still have the large trees that characterize the standard implementation of geometric semantic operators. However, all the evolutionary process can be performed efficiently and, if we are interested only in the best individual found by GP (which is the typical situation, where the best individual is interpreted as the model explaining data), we can perform the simplification of the expression on only one tree, instead of every tree in the population at each generation as proposed in [142]. Furthermore, the simplification is not performed during the evolution, but it can be done offline in a second step.

Excluding the time needed to simplify the best individual, the proposed implementation allowed us to evolve populations for thousands of generations with a speed up to at least 20 times higher than standard GP.

In the continuation of this section, we show a simple example that should clarify the functioning of the proposed algorithm.

8.2.1 *Example*

Let us consider the simple initial population P shown in Table 2 and the simple pool of random trees P_{mut} shown in Table 3 (usually P_{mut} contains a number of individuals much larger than P; in this example we consider both P and P_{mut} containing five individuals for simplicity). Besides the representation of the individuals in infix notation, these tables also contain an Id for each individual (T_1, T_2, T_3, T_4 and T_5 for the individuals in P and R_1, R_2, R_3, R_4 and R_5 for the individuals in P_{mut}). For simplicity, these Ids will be used from now on to address the different individuals, and individuals that will be created in the subsequent generations will be indicated using letter T follwed by progressive numbers (for example, the five individuals in the population at the second generation will be called T_6, T_7, T_8, T_9 and T_{10}).

Id	Individual
T_1	$x_1 + x_2 \cdot x_3$
T_2	$x_3 - x_2 \cdot x_4$
T_3	$x_3 + x_4 - 2 \cdot x_1$
T_4	$x_3 \cdot x_1$
T_5	$x_1 - x_3$

Table 2: The simple initial population P used in the example of Section 8.2.1. The leftmost column reports the Ids of the individuals. These Ids will be used in the text for simplicity.

Id	Individual
R_1	$x_1 + x_2 - 2 \cdot x_4$
R_2	$x_2 - x_1$
R_3	$x_1 + x_4 - 3 \cdot x_3$
R_4	$x_2 - x_3 - x_4$
R_5	$2 \cdot x_1$

Table 3: The individuals in the random pool P_{mut} used in the example of Section 8.2.1. The leftmost column reports the Ids of the individuals. These Ids will be used in the text for simplicity.

We now describe all the operations involved in the creation of the new population at the next generation, which we indicate as population P' from now on. Let us assume that the (non-deterministic) selection process imposes that T_6 is generated by crossover between T_4 and T_5. Analogously, let us assume that T_7 is generated by crossover between T_1 and T_4, T_8 is generated by crossover between T_1 and T_5, T_9 is generated by crossover between T_3 and T_4 and T_{10} is generated by crossover between T_3 and T_5. Furthermore, let us assume that to perform these five crossovers, individuals R_2, R_1, R_4, R_5 and R_3 of P_{mut} have to be used, respectively.

In our implementation, the individuals in P' are simply represented by the set of entries reported in Table 4, and stored in structure $\mathcal{M}$. In synthesis this table contains, for each new individual, a *reference* to the ancestors that have been used to generated it and the name of the operator used to generate it (either "crossover" or "mutation").

The only structures that we have to keep in memory during the GP run, besides the ones depicted in Tables 2, 3 and 4, are the two tables T_{vp} and T_{vm} that contain, at each generation, the values of the evaluation of the individuals in the current population and in P_{mut} for each fitness case. The size of the structure $\mathcal{M}$ reported in Table 4 grows during the GP run (in this example, five new entries are added to this table at each new generation, corresponding to the five new individuals in the population); however, it is very compact, because

Id	Operator	Entry
T_6	crossover	$\langle T_4, T_5, R_2 \rangle$
T_7	crossover	$\langle T_1, T_4, R_1 \rangle$
T_8	crossover	$\langle T_1, T_5, R_4 \rangle$
T_9	crossover	$\langle T_3, T_4, R_5 \rangle$
T_{10}	crossover	$\langle T_3, T_5, R_3 \rangle$

Table 4: How the individuals in the subsequent generations are stored in memory for the example of Section 8.2.1 (this structure is called $\mathcal{M}$ in the text). The leftmost column reports the Ids of the individuals. These Ids will be used in the text for simplicity. The central column reports the operation that has been used to generate the individual (it can be either "crossover" or "mutation". In this example, we use only crossover for simplicity). The rightmost column contains references to the ancestors used to generate the individual.

it only contains references, and thus we can manage it for several thousands of generations.

Let us assume that now we want to reconstruct the genotype of one of the individuals in P' (this typically happens only once, at the end of the run, for the best individual in the population). For instance, let us assume that the want to reconstruct T_8. Tables 2, 3 and 4 provide us with all the information we need to be able to do that. In particular, from Table 4 we learn that T_8 is obtained by crossover between T_1 and T_5, using random tree R_4. Thus, from the definition of geometric semantic crossover, we know that it will have the following structure: $(T_1 \cdot R_4) + ((1 - R_4) \cdot T_5)$. Table 2, that contains the syntactic structure of T_1 and T_5, and Table 3, that contains the syntactic structure of R_4, finally provide us with all the information we need to completely reconstruct the syntactic structure of T_8, which is:

$$((x_1 + x_2 \cdot x_3) \cdot (x_2 - x_3 - x_4)) + ((1 - (x_2 - x_3 - x_4)) \cdot (x_1 - x_3))$$

For simplicity, we have omitted mutation in this example and we have generated all the individuals in the new population using only crossover. Mutation works in a similar way, with the only differences that the central column in Table 4 contains the label "mutation" (and this information is useful because it tells us that, this time, we have to use the definition of geometric semantic mutation in order to reconstruct the individual) and the triplet associated to the newly generated individual this time contains one reference to an individual in P and two references to two individuals in P_{mut}.

It is important to remark that the time and space costs associated to this implementation of the geometric semantic operators are linear w.r.t. both the number of generations and the population size. In fact, at every generation we need $O(|P|)$ additional space (where $|P|$ is the population size). Thus, after g generations the space needed is $O(g|P|)$.

As for the time complexity, at every generation we need to produce $|P|$ new individuals, each one requiring a constant time. Thus, the time complexity for g generations is also $O(g|P|)$.

8.3 EMPIRICAL STUDY

8.3.1 *The Application*

The implementation provided so far makes the geometric semantic operators efficiently usable also on complex real-life applications. For this reason, for the first time, we are now able to validate those operators on one of those applications. We choose a real life problem in the field of pharmacokinetic.

As stated in [9], the availability of reliable pharmacokinetics prediction tools would permit to reduce the risk of late stage research failures in drug discovery and will enable to decrease the number of experiments and cavies used in pharmacological research, by optimizing the screening assays. Furthermore, predictive pharmacokinetic models would be of critical relevance for preventing Adverse Drug Reactions (ADRs), like those involved in the Lipobay-Baycol (cerivastatin) toxicity [194]. The potential of predictive modeling in terms of ADRs prediction is an hot research topic in medicine. Human oral bioavailability (indicated with %F from now on) is the parameter that measures the percentage of the initial orally submitted drug dose that effectively reaches the systemic blood circulation after the passage from the liver. This parameter is particularly relevant, because the oral assumption is usually the preferred way for supplying drugs to patients and because it is a representative measure of the quantity of active principle that can actuate its therapeutic effect. Being able to reliably predict the %F value for a potential new drug is outstandingly important, given that the majority of failures in compounds development from the early nineties to nowadays are due to a wrong prediction of this pharmacokinetic parameter during the drug discovery process [112, 109].

We have obtained a set of molecular structures and the corresponding %F values using the same data as in [221], using a public database of Food and Drug Administration (FDA) approved drugs and drug-like compounds [217]. The data has been gathered in a matrix composed by 359 rows and 242 columns. Each row (instance) is a vector of molecular descriptor values identifying a candidate new drug; each column (feature) represents a molecular descriptor, except the last one, that contains the known values of %F.

In our experiments, training and test sets have been obtained by randomly splitting the dataset: at each GP run, 70% of the molecules have been randomly selected with uniform probability and inserted into the training set, while the remaining 30% form the test set.

8.3.2 *Experimental Settings*

We tested the proposed implementation of GP with geometric semantic operators (GS-GP from now on) against a standard GP system (STD-GP). A total of 30 runs were performed with each technique considering different randomly generated partitions of the dataset into training and test set at each run. All the runs used populations of 100 individuals and the evolution stopped after 2000 generations. Trees initialization was performed with the Ramped Half-and-Half method [162] with a maximum initial depth equal to 6. The function set contained the four binary arithmetic operators $+$, $-$, $*$, and $/$ protected as in [162]. Fitness was calculated as the root mean squared error between outputs and targets (thus the lower the fitness, the better the individual). The terminal set contained 241 variables, each one corresponding to a different feature in the dataset. To create new individuals, STD-GP used standard (subtree swapping) crossover [162] and (subtree) mutation [162] with probabilities equal to 0.9 and 0.1 respectively. For GS-GP, crossover rate is 0.9, while mutation rate is 0.5. The motivation for this different mutation rate for the two GP systems is that a preliminary experimental study has been performed (independently for the two systems) for finding the parameter setting able to return the best results. Only the parameter settings that returned the best results for the two systems are presented here. Survival from one generation to the other was always guaranteed to the best individual of the population (elitism). No maximum tree depth limit has been imposed during the evolution.

In the next section, experimental results are reported using curves of the root mean square error on the training and test set. In particular, at each generation the best individual in the population (i.e. the one that has the smaller training error) has been chosen and the value of its error on the training and test has been stored. The reported curves finally contain the median of all these values collected at each generation. The median was preferred over the mean in the reported plots because of its higher robustness to outliers. The root mean square error on the training and test set, calculated as described above, will be in some cases informally indicated as training and test fitness, or training and test error, in the next section for simplicity.

8.3.3 *Experimental Results*

Figure 23 reports training and test error for STD-GP and GS-GP and clearly shows that GS-GP outperforms STD-GP on both training and test sets. In particular, GS-GP has a suitable behaviour: the curve of the error on the test set is quite "regular" and steadily decreasing during the whole evolutionary process. This behaviour on the test set gives us a hint of the fact that, contrarily to STD-GP, GS-GP does not

overfit training data for the considered application. Here we are using the number of generations as a the unit on the abscissa. While STD-GP and GS-GP generations are not the same in term of complexity, this does not give an advantage to GS-GP since, in term of execution time, GS-GP was always significantly faster than STD-GP.

Figure 24 reports a statistical study of the test fitness of the best individual, both for GS-GP and STD-GP, for each of the 30 performed runs. Let *IQR* be the interquartile range. The ends of the whiskers represent the lowest datum still within 1.5· *IQR* of the lower quartile, and the highest datum still within 1.5· *IQR* of the upper quartile. As it is possible to see, GS-GP produces solutions with a lower standard deviation with respect to the ones produced by STD-GP.

To analyze the statistical significance of these results, a set of tests has been performed on the median errors.

As a first step, the Kolmogorov-Smirnov test has shown that the data are not normally distributed and hence a rank-based statistic has been used. Successively, the Wilcoxon rank-sum test for pairwise data comparison has been used under the alternative hypothesis that the samples do not have equal medians. The p-values obtained are $6.0 \cdot 10^{-11}$ when test fitness of STD-GP is compared to test fitness of GS-GP and $7.1 \cdot 10^{-9}$ when training fitness of STD-GP is compared to training fitness of GS-GP. Therefore, when using a significance level $\alpha = 0.05$, we can clearly state that GS-GP produces fitness values that are significantly lower (i.e., better) than the STD-GP both on training and test data.

Besides comparing GS-GP with standard GP, we are also interested in comparing GS-GP with other well known state of the art Machine Learning methods, just to have an idea of the competitiveness of the results returned by GS-GP. Previous studies have appeared so far comparing several Machine Learning techniques for the prediction of the bioavailability of potentially new drugs. For instance, in [9] the following methods have been tested: linear regression, least square regression, multilayer perceptron, support vector machines regression with first degree polynomial and support vector machines regression with second degree polynomial kernel. In [9] all these methods are used with and without an explicit feature selection, performed on the original data as a preprocessing phase.

In [9] the feature selection methods used are principal component based feature selection and correlation based feature selection. Here, we take up exactly the same perspective, by using all these methods with and without these feature selection strategies on our dataset. As in [9] we used the implementations provided by the Weka public domain software [88] and, for each one of the used Machine Learning methods and feature selection strategies, we have used the default parameter setting of Weka. The results are reported in Table 5, where we can observe that the best performance was obtained by linear re-

gression with correlation based feature selection, that returned a root mean square error on the test set approximately equal to 27.52. Given that GS-GP, in the last performed generation, has returned a median test fitness equal to 30.44, and given that the best test fitness over the performed 30 runs was equal to 26.97, we state that GS-GP is able to find better, or at least comparable, results than the best one of the state of the art Machine Learning methods. We also point out that these results have been obtained by GS-GP without any explicit feature selection (given that GP is in general able to perform an automatic feature selection during the learning phase [143, 9]), while the best results of the state of the art methods have been obtained by explicitly selecting features by the correlation based technique. The explicit use of a preprocessing phase to select features has also been used so far in Evolutionary Computation in general [125], and in GP in particular [84], with excellent results. This should further improve GS-GP performances, and new experiments including explicit feature selection are part of our current research.

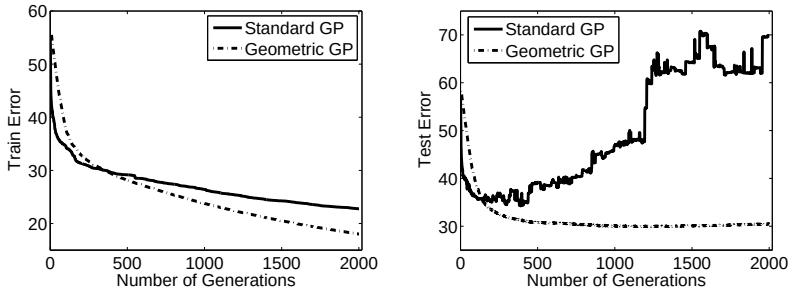


Figure 23: Median of train and test error for the consided techniques at each generation calculated over 30 independent runs.

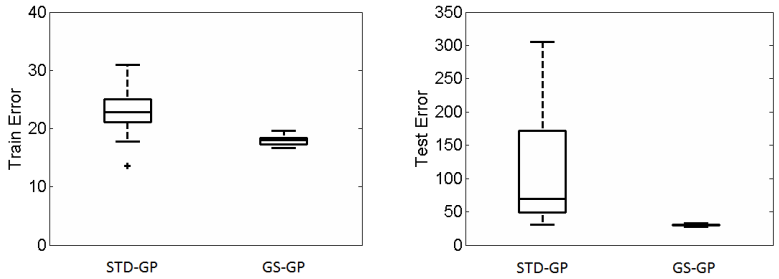


Figure 24: Train and test error of the best individual produced in each of the 30 runs at the last performed generation.

8.4 FURTHER REMARKS

New genetic operators, called geometric semantic operators, have been defined so far for genetic programming. They have the extremely

Method	Test RMSE
(a) No feature selection	
Linear regression	48.1049
Least square regression	37.2211
Multi layer perceptron	51.28
SVM regression-first degree polynomial kernel	34.804
SVM regression-second degree polynomial kernel	44.323
(b) Principal component based feature selection (PCFS)	
Linear regression	30.5568
Least square regression	40.4503
Multi layer perceptron	48.9771
SVM regression-first degree polynomial kernel	36.185
SVM regression-second degree polynomial kernel	42.3377
(c) Correlation based feature selection (CorrFS)	
Linear regression	27.5212
Least square regression	31.7826
Multi layer perceptron	32.5782
SVM regression-first degree polynomial kernel	28.8875
SVM regression-second degree polynomial kernel	29.7152

Table 5: Experimental comparison between different non-evolutionary Machine Learning techniques for oral bioavailability predictions. Error on the test reported for each technique.

interesting property of inducing a unimodal fitness landscape for any problem consisting in matching input data into known output ones (regression and classifications are instances of this general problem). This, at least at a theoretical level, should make all the problems of this kind easily solvable by genetic programming. Nevertheless, as demonstrated in the literature, these new operators, in their current definition, have a strong limitation, that makes them unusable in practice: they produce offspring that are larger than their parents, and this comports an exponential growth in the size of the individuals in the population.

We have overcome this limitation by proposing a new genetic programming system, in which geometric semantic operators are implemented in a very efficient way. The proposed implementation basically keeps in memory only the initial (randomly generated) population of programs, plus a set of randomly generated programs that will be used by the operators during the evolution. Furthermore, the implementation stores and maintains updated some tables containing pointers to those programs. The size of these tables grows linearly with generations, and thus managing those tables is quite feasible.

Thanks to this compact and efficient implementation, it is possible, for the first time, to employ the framework to solve complex prob-

lems, characterized by a large number of features. In particular, an important real life problem in the field of pharmacokinetic has been considered.

The presented experimental results demonstrate that the new system outperforms standard genetic programming and returns results that are better, or at least comparable to the best state of the art machine learning method for this application. Besides the fact that the new genetic programming system has excellent results on training data (which was expected, given that the fitness landscape is unimodal), we are positively surprised by its excellent generalization ability on the studied application, testified by the good results we have obtained on test data.

Part III

(EVOLUTIONARY) REACTION SYSTEMS

9

INTRODUCTION TO REACTION SYSTEMS

In this part of the thesis we introduce the concept of Reaction Systems and we study some combinatorial properties. Then, we will link the idea behind genetic programming with the representation for functions provided by reaction systems. This provide us with a new promising GP system: Evolutionary Reaction Systems. Since this GP system is new, after its introduction we will focus on its tuning, concluding with some further remarks and directions of future research.

9.1 REACTION SYSTEMS

In this section the notion of reaction system is introduced. We will recall some of the formal properties that they have and the ideas that led to their creation.

Reaction systems have been introduced by Ehrenfeucht and Rozenberg in 2004 as a formalism inspired by chemical reactions [51]. Its main aim was to be simple and easily extensible. In fact, in the following years the formalism was extended to also include, for example, the notion of time [53]. Extending the basic model is a necessity to allow the modelling of more complex phenomena, even if it makes a formal analysis more complex. This means that it is possible to have a trade-off between clarity, that simplifies the study of formal properties, and adherence to the modelled system.

9.1.1 *Basics of Reaction Systems*

The first concept to be defined is the concept of reaction that, being inspired to chemical reactions, comprises reactants, inhibitors and the products of the reaction.

Definition 9.1.1. A *reaction* $a = (R_a, I_a, P_a)$ is a triple of non-empty sets where R_a is called the set of reactants, I_a the set of inhibitors and P_a the set of products, with $R_a \cap I_a = \emptyset$. For any set S such that $R_a \subseteq S$, $I_a \subseteq S$ and $P_a \subseteq S$ we say that a is a reaction on S.

Given a set S, the set of all reactions on S is denoted by $rac(S)$. Given a set $T \subseteq S$ and a reaction $a \in rac(S)$, we say that a is enabled

in T iff $R_a \subseteq T$ and $I_a \cap T = \emptyset$. The result of a on T (denoted by $res_a(T)$) is P_a if a is enabled in T and $\emptyset$ otherwise. All these notions can be extended to sets of reactions. Given a set A of reactions, the reactant set is $R_A = \bigcup_{a \in A} R_a$, the inhibitor set is $I_A = \bigcup_{a \in A} I_a$ and the product set is $P_A = \bigcup_{a \in A} P_a$. The result set of A on T is $res_A(T) = \bigcup_{a \in A} res_a(T)$. Furthermore, we say that A is enabled on T iff for all $a \in A$, a is enabled on T.

Definition 9.1.2. A *reaction system* $\mathcal{A} = (S, A)$ is a pair where S is a finite set of symbols and $A \subseteq rac(S)$. The set S is called the background set of $\mathcal{A}$.

The result set of $\mathcal{A} = (S, A)$ on T is $res_{\mathcal{A}}(T) = res_A(T)$. The set of all enabled reactions of $\mathcal{A}$ on T (called the T-activity of A) is denoted by $en_{\mathcal{A}}(T)$. We will consider the *size* of a reaction system $\mathcal{A} = (S, A)$ as $|A|$.

9.1.2 *Dynamics of Reaction Systems*

An important characteristic of reaction systems is their dynamics, that is quite different from other system.

Definition 9.1.3. Let $\mathcal{A} = (S, A)$ be a reaction system. An interactive process $\pi = (\gamma, \delta)$ is a pair of sequences of subsets of S where $\gamma = C_0 C_1 \dots C_n$, $\delta = D_1 D_2 \dots D_n$ and for all $i \in \{2, \dots n\}$, $D_i = res_{\mathcal{A}}(D_{i-1} \cup C_{i-i1})$ and $D_1 = res_{\mathcal{A}}(C_0)$.

The sequence $D_1 \dots D_n$ is called the result sequence of π. It represents the set of results obtained at every steps from the components that are present in the system at the previous step. Note that even if a component is present at a certain time step if it is not produced again by a reaction then it is discarded at the next time step. This is a peculiarity of reaction systems, where the actions do not modify a state but are used to create a new state. So, if it is necessary to maintain an information for more that one time step then we need to have it recreated or inserted in the system at every time step.

The sequence $C_0 \dots C_n$ is called the context sequence of π. It represents the components that are introduced in the system at every time step. Note that C_0 represents the initial state. We may be also interested in systems where the intervention is limited at the generation of the initial state (i.e., where $C_i = \emptyset$ for all $i \geqslant 1$). These systems are called context-independent.

We will denote by W_i the set $W_i = D_i \cup C_i$ for all $i \in \{1, \dots, n\}$ and $W_0 = C_0$. The set of enabled reactions at a time step $i \in \{0, \dots, n-1\}$ is denoted as E_i and is $en_{\mathcal{A}}(W_i)$.

Example 9.1.1. Boolean functions can be easily represented by RS. As an example consider the *and* function with two inputs. It can be

represented as a reaction system with $S = \{x_1, x_2, \text{True}, i\}$, where x_1 and x_2 are the input variables, True is a constant that represents the output *true* of the system and i is a dummy inhibitor (a symbol that can only be in an inhibitor set and is never inserted in the system nor produced by other reaction). The set of reactions A contains only the reaction $\alpha = (\{x_1, x_2\}, \{i\}, \{\text{True}\})$. The outputs of the system with all possible inputs are the following:

$$\{x_1\} \longrightarrow \emptyset \qquad \{x_2\} \longrightarrow \emptyset \qquad \emptyset \longrightarrow \emptyset \qquad \{x_1, x_2\} \xrightarrow{\alpha} \{\text{True}\}$$

where on the left of the arrow there is the initial state, on the right the state at time 1 and the superscript over the arrow indicates the reactions that were enabled. This notation means that, if we have only x_1 in the system, no reaction is enabled and hence we obtain the empty set as a result. If only x_2 is present in the system then we also obtain the empty set as a result. Moreover, when we have no symbols in the system we do not generate other symbols. Finally, when both x_1 and x_2 are present in the system the reaction α is enabled (indicated by the superscript over the arrow) and we generate the symbol True. Denoting a true variable by inserting its corresponding symbol in the initial state of the system and a false variable by not inserting it, the reaction α clearly represents an *and* gate. Also notice that it is always possible to insert a dummy inhibitor (a symbol that is never present) and a dummy reactant (a symbol that is always present) in order to avoid the use of empty sets in the definition of reactions. Therefore, when using RS in practice (i.e., as a GP variant), we will allow empty sets either as reactant sets or as inhibitor sets since they can be easily simulated using dummy symbols.

9.1.3 *Equivalence of Reaction Systems*

In reaction systems it is important the study of equivalence between two reactions or two sets of reactions.

The first notion that is needed is the notion of functional equivalence. Two reactions $a, b \in rac(S)$ are said to be functionally equivalent (denoted by $a \sim b$) if for all $T \subseteq S$, $res_a(T) = res_b(T)$.

Ehrenfeucht and Rozenberg found the necessary conditions for obtaining the functional equivalence of two reactions [52]. They proved that two reactions $a, b \in rac(S)$ are functionally equivalent iff $R_a = R_b$, $I_a = I_b$ and $P_a = P_b$. The notion of functional equivalence can also be extended to sets of reactions. Two sets $A, B \subseteq rac(S)$ are functionally equivalent (denoted by $A \sim B$) iff for all $T \subseteq S$, $res_A(T) = res_B(T)$. It has been proved [52] that the problem of the functional equivalence between sets of reactions is coNP-complete.

It is possible to introduce a notion of partial ordering between reactions. Given two reactions $a, b \in rac(S)$, a covers b (denoted by $a \geqslant b$) iff for all $T \subseteq S$, $res_a(T) \supseteq res_b(T)$. It is immediate that

$a \sim b \Leftrightarrow a \geqslant b \wedge b \geqslant a$. It has been proved [52] that $a \geqslant b$ iff $R_a \subseteq R_b$, $I_a \subseteq I_b$ and $P_b \subseteq P_a$.

9.2 MOTIVATIONS FOR EVOLUTIONARY REACTION SYSTEMS

It is nowadays about fifty years since the very first computational experiments that originated Genetic Programming (GP) and about twenty years since John Koza named and popularised the method [114]. During the past two decades there has been a significant range and volume of development in the theory and application of GP and GP is nowadays recognized as a well established research field [162]. Large part of the efforts of researchers has been dedicated to the study of the evolution of several different computational formalisms, that can help practitioners to solve problems with different levels of expressiveness. Under this perspective, from the very earliest experiments in the automatic generation of executable structures [66] a variety of representations have been explored starting with binary string machine code [75], finite state automata [68], generative grammatical encodings [216] to the dominant tree-based form popularised by Koza [114]. To this day numerous alternative representations have been proposed including graph [191], strongly-typed [135], linear-tree [104], and linear-graph [105]. Among the many variants, particularly popular are the developments in grammar-based GP (see for instance [155]) and cartesian GP (see for instance [132]). Besides [162], the interested reader is referred to [156] for an in-depth discussion of the open issues opened by the several different GP representation models that have been proposed along the years.

This work is situated in this vast research field, and its aim is the one of proposing a new GP system, able to evolve programs expressed in a new and challenging computation formalism called *Reaction Systems* (RS) and recently introduced by Rozenberg and coworkers [51]. This new GP variant will be called *Evolutionary Reaction Systems* (EvoRS).

Why introducing a new GP variant, evolving another computational formalism, despite the many variants already defined so far? Many answers could be given to this question, justifying the fact that evolving RS is interesting and relevant. First of all, RS is a powerful and expressive computation formalism, that is particularly intuitive.

Another reason why the introduction of EvoRS is, in our opinion, relevant is that it lightens the final user from the burden of defining the set of functional symbols used to build up the evolved programs. This definition is clearly a crucial step in many of the most currently used GP variants, including tree based GP and grammar-based GP, since it has a direct impact on the ability of the GP system to find good solutions and it must be completely hand-defined by the final user.

Last but not least, RS is a bio-inspired computational formalism, and in [156], O'Neill and coworkers dedicate an entire section of their GP open issues chapter to "The Influence of Biology on GP", claiming that we currently do not use a sufficient set of features from biological evolution to embody its full potential in our artificial evolutionary process, and that in order to provide GP with new potentials and power we need to go back to the natural example of biology and to study what else can be learned from it.

Not only our objective is introducing EvoRS and discussing its functioning, but we also want to give an idea of the potentialities of this new evolutionary algorithm, by comparing its performances with the ones of other well known machine learning methods (including standard tree-based GP) and by discussing the expressiveness of its returned solutions.

9.3 MOTIVATIONS FOR PARAMETER TUNING

While the choice of individual representation, genetic operators and fitness function are fundamental ingredients for an Evolutionary Algorithm (EA) to achieve good results, a correct choice of these can still lead to poor results if the algorithm's parameters are not carefully set. Choosing the right parameter values, however, is a very difficult and time-consuming task, given their usually large number and their high level of inter-dependency. Thus, methods to address this problem are one of the priorities of the EA community. Although EvoRS produce results that are comparable to (and in some cases even better than) the ones produced by standard tree-based genetic programming and other well established machine learning techniques for a large set of problems, it is necessary to investigate the effect of its parameters values on the search process.

According to [55], it is possible to distinguish two major forms of setting parameter values: parameter tuning and parameter control. Parameter tuning is the commonly practiced approach that consists in finding good values for the parameters (for instance by means of a set of preliminary experiments) before the final runs of the algorithm and then running the algorithm using these values, which remain fixed during the run. On the other hand, parameters control consists in starting a run with initial parameter values which are dynamically changed during the run. Thus, the main difference is that while parameter control is done online (i.e., during the execution of the EA), parameter tuning is made *a priori*, with the parameters remaining static during the EA evolution [114].

As stated in [44], parameter tuning is a hard problem because there is a great number of possible parameter value combinations, which have to be set with very little available information regarding the effect that each parameter has on the EA performance [147]. Nowadays,

most EA users rely on some rules that emerged by the experience in using well-known EAs (e.g., crossover rates should be high and mutation rates should be low). But this empirical approach should not be followed when a new EA, like EvoRS, is proposed for the first time. In fact, while EvoRS presents several parameters that are commonly used in other evolutionary techniques, it is also characterized by its own parameters. Hence, an accurate analysis of their reciprocal interactions should be pursued.

9.4 PARAMETER TUNING: STATE OF THE ART

In this section, a literature review of parameter tuning in EAs is presented. As reported before, most EA users rely on rules of thumb and ad hoc choices of parameters [17]. However, in recent years, a few methods for making this task more systematic have been proposed. In [40] authors use a rigorous yet practical statistical methodology for the exploratory study of genetic algorithms. In their research, they examine the relationship between the statistical significance of interaction among crossover and mutation and increasing modality of a problem. They find that, as their test function increases in modality, the interaction between crossover and mutation becomes statistically significant. The effect of the interaction is striking when examining response curves, which illustrate distinct inflection. They conjecture that for highly modal functions the possibility of interaction between crossover and mutation must be considered. The practical implication of interaction is that when attempting to fine tune a genetic algorithm on a highly modal problem, the suitable/optimal rates for crossover and mutation cannot be obtained independently. All combinations of crossover and mutation, within given starting ranges, must be investigated in order to allow for the interaction effect.

In our work we evaluate the effects that different values of the parameters that characterize an EvoRS have on the search process. In doing this, we consider the fitness of the solutions and we try to draw some general consideration considering the statistical significance of the obtained results. It is important to underline that we do not study the interaction between different parameters. Our goal is to study the effect that a single parameter produces by itself on the search process.

In [146], authors underline how calibrating the parameters of an EA is a laborious task. In particular, authors focus on the fact that the highly stochastic nature of an EA typically leads to a high variance of the measurements. The standard statistical method to reduce variance is measurement replication, (i.e., averaging over several test runs with identical parameter settings). The computational cost of measurement replication scales with the variance and is often too high to allow for results of statistical significance. In their work, the authors study an alternative, that they call the REVAC method for Relevance

Estimation and Value Calibration, and they investigate how different levels of measurement replication influence the cost and quality of its calibration results. Two sets of experiments are reported: calibrating a genetic algorithm on standard benchmark problems, and calibrating a complex simulation in evolutionary agent-based economics. They find that measurement replication is not essential to REVAC, which emerges as a strong and efficient alternative to existing statistical methods.

Because our work represents a first attempt in tuning the parameters of an EvoRS, we decided to use existing statistical methods to perform this task. Nevertheless, REVAC method could represent a good choice for a future study. In fact the computational cost of measurement replication may be too high, especially if we decide to analyze the interaction between the different parameters.

In [147], an empirical study on the impact of different design choices on the performance of an EA is proposed. Four EA components are considered (parent selection, survivor selection, recombination and mutation) and, for each component, the authors study the impact of choosing the right operator, and of tuning its free parameter(s). They tune 120 different combinations of EA operators to 4 different classes of fitness landscapes, and measure the cost of tuning. They find that components differ greatly in importance. Typically, the choice of the operator for parent selection has the greatest impact, and mutation needs the most tuning.

Our work is similar to the one proposed in [147]. An empirical study on the impact that different parameters setting have on the performance of the final solution is proposed. As said before, the analysis we propose regards only the parameters that characterize an EvoRS. The aim is to find which parameter has the greatest effect on the quality of the solutions.

In [18], an approach for determining adequate parameters of optimization algorithms is proposed. The approach is called sequential parameter optimization (SPO). SPO is a heuristic that combines classical and modern statistical techniques to improve the performance of search algorithms. Although sequential parameter optimization relies on enhanced statistical techniques such as design and analysis of computer experiments, it can be performed algorithmically and requires basically the specification of the relevant algorithm's parameters. In their work, the authors demonstrate the usefulness of SPO for very different algorithms and optimization tasks and explain how it works. Moreover they discuss possible SPO use cases highlighting strengths and weaknesses of the method.

Regarding our work, SPO can be used in a future investigation. In fact, whenever parameters for an algorithm-problem combination have not been thoroughly searched before, application of SPO makes

sense if we want to have a competitive parameter set. Obviously, it is important to consider the limitations of SPO also reported in [18].

In [44], a new methodology is described for tuning parameters of genetic programming using factorial designs, one-factor designs and multiple linear regression [97]. The presented experiments show that factorial designs can be used to determine which parameters have the largest effect on the algorithm's performance. This way, parameter setting efforts can focus on them, largely reducing the parameter search space. The use of factorial design is also proposed in [65], where the authors evaluate the individual and combined effects of genetic programming parameters.

The use of factorial designs to determine which parameters have the largest effect on the algorithm's performance is related to our work, where we look for the parameter (between the ones that characterize an EvoRS) that has the greater impact on the search process. As stated before, in our work we use standard statistical analysis.

In [181] the most important issues related to tuning EA parameters are discussed, describing a number of existing tuning methods, and presenting an experimental comparison among them.

9.5 PARAMETER TUNING AND PARAMETER CONTROL

As already pointed out so far, there exist two major forms of setting parameter values: parameter tuning and parameter control. While parameter tuning is performed before starting the EA, parameter control is an online process and allows a dynamic change of the parameter values. This work is a first attempt to understand the role of the parameters that characterize EvoRS, thus a tuning of the parameter is sufficient for our purposes. Nonetheless we are aware that parameter tuning presents some drawbacks and better performances could be achieved using parameter control.

In [55], drawbacks of different tuning methods are discussed. The authors underline that a general drawback of the parameter tuning approach, regardless of how the parameters are tuned, is based on the observation that a run of an EA is an intrinsically dynamic, adaptive process. The use of rigid parameters that do not change their values is thus in contrast to this spirit. Additionally, it is intuitively obvious that different values of parameters might be optimal at different stages of the evolutionary process [12, 13, 92, 184]. For instance, large mutation steps can be good in the early generations, helping the exploration of the search space, and small mutation steps might be needed in the late generations to help fine tuning the suboptimal chromosomes. This implies that the use of static parameters itself can lead to inferior algorithm performance.

Thus, it is clear that parameter tuning is not the best choice if the main objective is to improve the performances of an EA. On the other

hand, when the aim is simply to study which parameters mainly affect the search process, a static parameter settings could produce a better understanding.

10

COMBINATORICS OF REACTION SYSTEMS

In this chapter a first study on the behaviour of large reaction systems has been performed. We used for the first time the concepts of extremal combinatorics in the field of reaction system.

10.1 COMBINATORICS

Combinatorial statements arise almost naturally in many fields. Particularly important are the statements about the properties that a certain kind of structure can have when its size increases. A famous statement of this kind is the *Ramsey theorem*, that states that for any $k \in \mathbb{N}$, any large enough complete graph that is two-colored has a monochromatic subgraph of k nodes. Other examples come from the most different mathematical structures. These statements can be presented in many ways. We can either give importance to the existence of a size bound after which a certain property holds (a more *Ramsey*-like view). Otherwise we can try to find the exact value of this bound or, at least, to obtain some information on its order of magnitude (an extremal combinatorics point of view). To prove statements of this kind a large body of work regarding proof techniques has been produced, from various counting techniques to combinatorial proofs to linear algebra methods. Since it is impossible to give a detailed account of all the results and the techniques, we refer the reader to specific books (see, for example, [103] for extremal combinatorics and [82] for Ramsey theory).

The properties that we want to study are the following:

- The minimal size after which a reaction system includes two reactions such that the chemicals necessary for one of them inhibit the other.
- The minimal size after which a reaction system necessarily has a reaction that produces the inhibitor of another reaction.
- The minimal size after which a reaction system can be substituted by a smaller reaction system without having an external observer noticing it.

These properties can clarify the limits in size and parallelism that are inherent in the definition of reaction systems. Many of the results obtained can also be restated as Ramsey-style statements (i.e., in terms of presence of an "ordered" substructure inside a large enough structure).

In Section 9.1 the basic notions regarding reaction systems are recalled. The results of this chapter are exposed in Section 10.2 and some further remarks and directions of future works are presented in Section 10.3.

10.2 PROPERTIES AND BOUNDS OF REACTION SYSTEMS

The behaviour of reaction systems has been studied focusing on the construction of reaction systems with the desired behaviour. Only recently some work on the study of the properties and the dynamical behaviour of a random reaction system has been carried on [54]. An important insight on the behaviour of a large group of reaction system can be obtained by studying the properties that any reaction system with a large enough size must have. In this section we will define three properties that a reaction system can have and we will prove that any large enough reaction system possess all these properties. Furthermore, since the minimal size to certainly have these properties is asymptotically smaller than the number of possible reaction systems when the number of symbols increases, we have that these properties are verified by the large majority of reaction systems.

When there exists a threshold after which a certain property surely happens we will denote this threshold by $R(P, n_1, \ldots, n_k)$ where P is the name of the property and $n_1, \ldots, n_k$ are parameters, usually positive integers.

The first property to be studied will be called is No-Concurrency (NC). This property states that in a set of reactions A there exist two reactions that cannot be executed at the same time step.

Definition 10.2.1. Given a reaction system $\mathcal{A} = (S, A)$, $\mathcal{A}$ has the property NC if there exist $a, b \in A$ such that for all $T \subseteq S$, $a \notin en_{\mathcal{A}}(T)$ or $b \notin en_{\mathcal{A}}(T)$.

Note that this property can also be stated as the existence of two reactions $a, b \in A$ such that $R_a \cap I_b \neq \emptyset$.

Proposition 10.2.1. *For any* $n \in \mathbb{N}_+$ *there exists* $R(NC, n) = (2^n - 1)\left(2^n - 2^{\lceil \frac{n}{2} \rceil} - 2^{\lfloor \frac{n}{2} \rfloor} + 1\right) + 1$, *such that for any reaction system* $\mathcal{A} = (S, A)$ *with* $|S| = n$, *if* $|A| \geqslant R(NC, n)$ *then* $\mathcal{A}$ *has the property NC.*

Proof. Let A be a set of reactions over S. When $A = rac(S)$ we have two reactions $a = (R_a, I_a, P_a)$ and $b = (R_b, R_a, P_a)$. This proves that $R(NC, n)$ always exists. Now it is necessary to prove that $R(NC, n) = (2^n - 1)\left(2^n - 2^{\lceil \frac{n}{2} \rceil} - 2^{\lfloor \frac{n}{2} \rfloor} + 1\right) + 1$. The set A does not have any pair

of reactions that satisfy property NC iff $R_a \cap I_a \neq \emptyset$. For the sake of argument, suppose otherwise. Then there exists a reaction a such that $R_a \cap I_A \neq \emptyset$. Furthermore, there exists a reaction b such that $R_a \cap I_b \neq \emptyset$. But this is property NC that we have assumed not to hold.

Let $|R_A| = k$ then there exist $2^k - 1$ possible subsets of R_A that can be reactant sets for a reaction in A. Since I_A is disjoint from R_A its cardinality is at most $n - k$, allowing a number of possible inhibitor sets of $2^{n-k} - 1$. Since we have no condition on P_A its cardinality can be as high as n, allowing $2^n - 1$ different product sets. Hence the maximum number of reactions in A is $(2^k - 1)(2^{n-k} - 1)(2^n - 1) = (2^n - 1)(2^n - 2^k - 2^{n-k} + 1)$. This is maximized when $k = \frac{n}{2}$ (if n is not even we either take the floor or the ceiling of $\frac{n}{2}$). All others reactions necessarily have a reactant set that intersects I_A or an inhibitor set that intersects R_A. Thus, $R(NC, n) = (2^n - 1)\left(2^n - 2^{\lceil \frac{n}{2} \rceil} - 2^{\lfloor \frac{n}{2} \rfloor} + 1\right) + 1$. □

A similar proposition regards the property of having at least one reaction that produces at least one of the inhibitors of another reaction (property Inh). In a system without this property the dynamics is given only by the reactants and the products, since the inhibitors play no role in the dynamics.

Definition 10.2.2. Given a reaction system $\mathcal{A} = (S, A)$, it has property Inh if there exist $a, b \in A$ such that $P_a \cap I_b \neq \emptyset$.

Proposition 10.2.2. *For any $n \in \mathbb{N}_+$ there exists $R(Inh, n) \in \mathbb{N}$ with $\left(2^{n-1} - 1\right)^2 < R(Inh, n) < (2^n - 1)\left(2^n - 2^{\lceil \frac{n}{2} \rceil} - 2^{\lfloor \frac{n}{2} \rfloor} + 1\right)$ such that for all reaction systems $\mathcal{A} = (S, A)$ with $|S| = n$, if $|A| \geqslant R(Inh, n)$ then $\mathcal{A}$ has property Inh.*

Proof. The existence of $R(Inh, n)$ is immediate since the system given by $(S, rac(S))$ has the desired property. We need to prove that at least one system with $\left(2^{n-1} - 1\right)^2$ reactions without property Inh exists. First of all, consider a system $\mathcal{A} = (S, A)$ such that $P_A \cap I_A = \emptyset$ and $R_A \cap I_A = \emptyset$ (note that this condition is stronger than the requirement that $R_a \cap I_a = \emptyset$ for all $a \in A$). Once fixed the cardinality of $|I_A| = n - k$ with $1 \leqslant k \leqslant n - 1$, the cardinality of P_A and R_A can be at most k. Hence, the number of reactions in $\mathcal{A}$ can be at most $\left(2^k - 1\right)^2 (2^{n-k} - 1)$. This value is maximized when $k = n - 1$, so that the maximum number of reactions in $\mathcal{A}$ is at most $\left(2^{n-1} - 1\right)^2$.

The last claim to prove is that $R(Inh, n)$ is strictly smaller than $(2^n - 1)\left(2^n - 2^{\lceil \frac{n}{2} \rceil} - 2^{\lfloor \frac{n}{2} \rfloor} + 1\right)$. Let $\mathcal{A} = (S, A)$ and such that $P_A \cap I_a = \emptyset$ with no restriction on $R_a \cap I_a$ for all $a \in A$ (i.e., we also count triples that are not reactions). When the cardinality of P_A is fixed to $0 < k < n$, the cardinality of I_A is $n - k$. Since we are not posing restrictions on R_A, its cardinality can be maximized to be n. Hence the number of triples is $(2^n - 1)(2^{n-k} - 1)(2^k - 1)$, a value that is maximized when $k = \lceil \frac{n}{2} \rceil$. Consequently the maximum number of triples that can be

in A without obtaining property Inh is $(2^n-1)\left(2^n-2^{\lceil\frac{n}{2}\rceil}-2^{\lfloor\frac{n}{2}\rfloor}+1\right)$. To prove that the inequality is strict is only necessarily to note that at least one triple in A is not a reaction (i.e., (I_a, I_a, P_a) for any I_a and P_a is not a reaction). □

Note that

$$\lim_{n\to+\infty} \frac{\left(2^{n-1}-1\right)^2}{(2^n-1)\left(2^n-2^{\lfloor\frac{n}{2}\rfloor}-2^{\lceil\frac{n}{2}\rceil}+1\right)} = \frac{1}{4}$$

hence the two bounds differs asymptotically only by a multiplicative factor.

Another property that can be interesting is the possibility for a system to be equivalent to a proper subset of itself.

Definition 10.2.3. Given a reaction system $\mathcal{A} = (S, A)$, it is *shrinkable* if there exists $B \subset A$ such that $A \sim B$.

In other words, a reaction system is shrinkable if its set of reactions contains reactions that have no impact on the dynamics of the system. It is useful to provide sufficient conditions to have a shrinkable reaction system. One of these conditions is linked to the presence of a maximal antichain inside the set of reactions.

Definition 10.2.4. Given a reaction system $\mathcal{A} = (S, A)$ it has property Comp if for all $a \in \mathrm{rac}(S)$, there exists $b \in A$ such that either $a \geqslant b$ or $b \geqslant a$.

In other words, a system has property Comp if it contains a maximal antichain of reactions.

Proposition 10.2.3. *Let $\mathcal{A} = (S, A)$ be a reaction system. If there exists $\mathcal{B} = (S, B)$ with $B \subset A$ and $\mathcal{B}$ has property Comp then $\mathcal{A}$ is shrinkable.*

Proof. First of all it is necessary to note that given two reactions $a, b \in A$ if $a \geqslant b$ then the system $\mathcal{A}' = (S, A \setminus \{b\})$ is equivalent to $\mathcal{A}$. Let $C \subseteq A$ be the set of maximal elements of A. The reaction system $\mathcal{C} = (S, C)$ is equivalent to $\mathcal{A}$ since no element of $A \setminus C$ is greater or incomparable to any element of A. It is only necessary to prove that $A \setminus C \neq \emptyset$. Since there exists a set $B \subset A$ that has property Comp, there exists $a \in B \setminus A$ such that there exists $b \in B$ with $a \geqslant b$ or $b \geqslant a$. Because of this at least one between a and b cannot be an element of C, hence $C \subset A$. □

By the previous proposition, it is useful to find a bound after which every system has property Comp. Then every reaction system having at least one reaction more than the bound is shrinkable.

Proposition 10.2.4. *For any $n \in \mathbb{N}_+$ there exists $R(Comp, n) \in \mathbb{N}$ with $\frac{n!(2^n-2)}{\lceil\frac{n}{2}\rceil!\lfloor\frac{n}{2}\rfloor!} \leqslant R(Comp, n) \leqslant \frac{n!(3^n-2^{n+1}+1)}{\lceil\frac{n}{2}\rceil!\lfloor\frac{n}{2}\rfloor!}$ such that for all reaction system $\mathcal{A} = (S, A)$ with $|S| = n$, if $|A| \geqslant R(Comp, n)$ then $\mathcal{A}$ has property Comp.*

Proof. The existence of $R(\text{Comp}, n)$ is immediate since the system whose set of reactions is the set of all possible reaction certainly satisfies Comp. To prove that $R(\text{Comp}, n) \geqslant \frac{n!(2^n-2)}{\lceil \frac{n}{2} \rceil ! \lfloor \frac{n}{2} \rfloor !}$ we have to show a reaction system with an antichain of $\frac{n!(2^n-2)}{\lceil \frac{n}{2} \rceil ! \lfloor \frac{n}{2} \rfloor !}$ reactions. Consider the set A all the reactions in the form $(R, S \setminus R, P)$ with $|P| = \lceil \frac{n}{2} \rceil$. These reactions are pairwise incomparable. Consider two distinct reactions a, b. If $P_a \neq P_b$ then the two reactions are not comparable since all the product sets are of the same cardinality and, hence, P_a and P_b are non-comparable. If $P_a = P_b$ then to have $a \geqslant b$ (the case $b \geqslant a$ is symmetric) we must have that $R_a \subseteq R_b$ and $S \setminus R_a \subseteq S \setminus R_b$, that is equivalent to $R_a \supseteq R_b$. This conditions hold only when $R_a = R_b$. Since we have supposed a distinct from b we have that they are non-comparable. It is now necessary to prove that the number of reactions in the form $(R, S \setminus R, P)$ is $\frac{n!(2^n-2)}{\lceil \frac{n}{2} \rceil ! \lfloor \frac{n}{2} \rfloor !}$. First of all, recall that for a fixed cardinality k there are $\binom{n}{k}$ subsets of S with that cardinality. This value is maximized when $k = \frac{n}{2}$ (or, when n is odd, either $\lceil \frac{n}{2} \rceil$ or $\lfloor \frac{n}{2} \rfloor$). Hence, for the product set there are $\binom{n}{\lceil \frac{n}{2} \rceil}$ possible sets. The set of inhibitors is completely determined by the set of reactants. The possible sets of reactants are $2^n - 2$ since all proper subsets of S are valid.

To prove that $R(\text{Comp}, n)$ is at most $\frac{n!(3^n-2^{n+1}+1)}{\lceil \frac{n}{2} \rceil ! \lfloor \frac{n}{2} \rfloor !}$ we are going to show a system with property Comp with this number of reactions such that any set of reactions with no comparable reactions has smaller size. Let B be a set of all reactions in the form (R, I, P) with $|P| = \lceil \frac{n}{2} \rceil$. By using the proof of Proposition 10.2.5 and the counting done for $|A|$, we can see that the cardinality of B is $\frac{n!(3^n-2^{n+1}+1)}{\lceil \frac{n}{2} \rceil ! \lfloor \frac{n}{2} \rfloor !}$. Consider a set of reactions C such that it contains no comparable reactions. Let $a = (R_a, I_a, P_a) \in C$. We claim that there exists an injective map f from C to B in the form $(R_a, I_a, P_a) \mapsto (R_a, I_a, P_{g(a)})$ (i.e., only P_a can have an image that is not itself). Suppose such a map does not exists. This means that there exists $a, b \in C$ with $a \neq b$ such that $(R_a, I_a, P_{g(a)}) = (R_b, I_b, P_{g(b)})$. This holds only when $R_a = R_b$, $I_a = I_b$ and $P_{g(a)} = P_{g(b)}$. Let $C_a = \{(R_a, I_a, P) \in C\}$ (equiv. B_a). We have that a map f such that its restrictions to C_a are injective for any C_a is injective. This holds because $f|C_a$ has as image as a subset of B_a and all B_a are disjoint. Since all C_a are also antichains, for all $a, b \in C_a$, P_a and P_b are not comparable. Thus, when we consider the set $\{P \mid (R_a, I_a, P) \in C_a\}$ we have that it contains only non-comparable sets. But the set $\{P \mid (R_a, I_a, P) \in B_a\}$ is a maximum antichain in $\mathcal{P}(S)$ (i.e., there is no antichain with more elemsnts), hence an injection from the first to the second set is possible, contradicting the existence of two distinct reactions in C with the same image. This means that $|C| \leqslant |B|$. □

The previous proposition can also be reformulated into a Ramsey-like statement. That is, every large enough reaction system contains a

maximal antichain of reactions. Or, with more emphasis on the property of being shrinkable, every large enough reaction system contains a subset of it that is equivalent to the whole system.

We can study how all the properties and bounds above relates with the number of possible reaction systems over a set of symbols of a fixed cardinality. The number of reaction systems is given by the following proposition.

Proposition 10.2.5. *For all* $n \in \mathbb{N}$*, given a set* S *of symbols with* $|S| = n$*,* $|rac(S)| = (2^n - 1)(3^n - 2^{n+1} + 1)$*.*

Proof. The number of reactions is given by the number of possible combinations of reactants, inhibitors and products. Let $|S| = n$. Then the number of possible combinations of reactants and inhibitors is:

$$\sum_{i=1}^{n-1} \binom{n}{i} \sum_{j=1}^{n-i} \binom{n-i}{j} = \sum_{i=1}^{n-1} \binom{n}{i} (2^{n-i} - 1) =$$
$$\sum_{i=1}^{n-1} \binom{n}{i} 2^{n-i} - \sum_{i=1}^{n-1} \binom{n}{i} = \sum_{i=1}^{n-1} \binom{n}{i} 2^{n-i} - 2^n + 2$$

Also, by the properties of of binomial coefficient we have that:

$$\sum_{i=1}^{n-1} \binom{n}{i} 2^{n-i} = -2^n - 1 + \sum_{i=0}^{n} \binom{n}{i} 2^{n-i} = -2^n - 1 + (2+1)^n$$

Obtaining a total of $3^n - 2^{n+1} + 1$. By multiplication with the number of possible product sets we obtain a total of $(2^n - 1)(3^n - 2^{n+1} + 1)$ reactions. □

It is easy to see that, as the number of symbols increases, the majority of reactions systems are not fully parallel, cannot sustain full parallelism and are equivalent to smaller reaction system.

10.3 FURTHER REMARKS

In this section the main results of this work are recalled and some directions of future research are illustrated.

This work provided a first application of extremal combinatorics in the study of large reaction systems. The results regard some basic properties of reaction systems. In particular the presence of reactions that cannot be enabled at the same time, the presence of a reaction that can inhibit another reaction and the presence of comparable reactions in a reaction system. All those properties are satisfied by large enough reaction systems. For all these properties bound on the minimal size to certainly have the desired properties have been provided.

Future works should focus on finding other properties that can aid in the analysis of reaction systems. A particular attention should be

taken in establishing a collection of proof techniques that can be easily and widely applied. In particular proof techniques that are common in extremal combinatorics and related field could be easily applied to reaction systems.

The studies on the extremal properties and the appearance of order for large enough reaction systems can provide insights on the behaviour of large reaction systems and, possibly, on the likeliness of having a certain property given a random reaction system.

11

EVOLUTIONARY REACTION SYSTEMS

This chapter introduces Evolutionary Reaction Systems and presents a comparison with known machine learning techniques and experiments on the parameter tuning of this new evolutionary algorithm. This Chapter is structured as follows: Section 11.1 presents EvoRS, illustrating the main ideas behind it, and its composing elements; Section 15.1.5 discusses the test problems that we have used to validate EvoRS and the experimental settings and obtained results. Section 11.3 describes the test problems and the experimental settings used for parameter tuning. Subsequently, experimental results concerning the role of the different parameters in EvoRS is proposed. Finally, Section 11.4 provides a summary of the presented work and proposes ideas for future research.

11.1 EVOLUTIONARY REACTION SYSTEMS

In this section an evolutionary version of reaction systems is presented. We will call it *Evolutionary Reaction Systems* (EvoRS). An EvoRS individual is a RS. A population is a set of RS. We will discuss only the phases of EvoRS that are different from others evolutionary algorithms. For example, selection, being based on the phenotype, it does not depend on the particular representation used and then could be performed using one of the standard algorithms (i.e., roulette-wheel, tournament, etc.).

11.1.1 *Input and Output for EvoRS*

One first aspect to note is that it is necessary to allow both input and output from a RS. Let $x_1, \ldots, x_m$ be the set of input variables. Suppose that every variable can assume only a finite number n_i of values: $x_i \in \{k_1, \ldots, k_{n_i}\}$. Fix $i \in \{1, \ldots, m\}$. To the variable x_i we will associate $n_i - 1$ input symbols, that represents the predicates $x_i = k_2, \ldots, x_i = k_{n_i}$. The predicate $x_i = k_1$ will be represented by the absence of an input symbol. Thus, there will be $\sum_{i=1}^{m}(n_i - 1)$ input symbols. An important characteristic of RS is that the symbols that are not explicitly preserved from one step to the other disappears (e.g., in 9.1.1 the symbol True is not preserved, hence we can have it at

$t = 1$ but not at $t = 2$). In fact, using a set of output symbols, if one of them appears at time t, we are not assured that in the subsequent time steps it will be preserved. Therefore, we decided to fix a parameter `execution length` that is a positive natural number. Suppose we have $o_0, o_1, \ldots, o_n$ possible output values. Choose o_0 as a default value. We will prevent it from being generated by any reaction. We run the RS for `execution length` steps and, if during the execution one of the output symbols $o_1, \ldots, o_n$ is produced then it is returned as output. Otherwise the default value o_0 is returned. In this way we are certain that a RS will always produce an output. Hence, a fitness evaluation can be performed.

11.1.2 *Initialization*

The initialization of an EvoRS is simply the creation of a population of randomly generated RS. Three parameters are needed: (1) the `population size`, (2) the `number of symbols` that can be used in the system (note that they need to be at least as many as the input symbols plus the output symbols); (3) the maximum `initial size` (when a RS is randomly generated, the number of reactions that it contains – i.e., its size – is chosen randomly but its value is bounded by the `initial size` parameter).

11.1.3 *Crossover*

Given the list of individuals $\mathcal{A}_1, \ldots, \mathcal{A}_n$ obtained by selection, for any pair $\mathcal{A}_{2i-1}, \mathcal{A}_{2i}$ with $1 \leqslant i \leqslant \frac{n}{2}$, there is a probability p_c that it will be subject to crossover. Given two RS, $\mathcal{A} = (S, A)$ and $\mathcal{B} = (S, B)$ a *crossover* of $\mathcal{A}$ and $\mathcal{B}$ is a stochastic operator that generates two reactions systems $\mathcal{A}'$ and $\mathcal{B}'$ in the following way: (1) let $C = A \cup B$; (2) let $k \in \{1, \ldots, |C| - 1\}$; (3) let A' be a subset of cardinality k of C (it can be randomly selected, or, if the element of C are ordered, it is possible to take the first k elements); we define: $\mathcal{A}' = (S, A')$ and $\mathcal{B}' = (S, C \setminus A')$. Note that since $A \cap B$ can be non-empty, we have that $|A| + |B| \geqslant |A \cup B| = |A'| + |B'|$, thus possibly reducing the total number of reactions.

11.1.4 *Mutation*

We have defined three kind of mutation for RS. One type of mutation is the *random reaction insertion*. Fix a RS $\mathcal{A} = (S, A)$. With probability p_{in} for any of the reactions in A, another randomly generated reaction is inserted. The second type of mutation is the *random reaction removal*. Let $\mathcal{A} = (S, A)$ be a RS. All reactions in A have probability p_{rm} of being removed. The last kind of mutation, called *renewal*, is, in fact, a random recreation of the RS with probability p_{ren}. The system that is

Parameter	Meaning
`population size`	Number of RS in the population
`number of symbols`	Number of symbols used in the creation of reactions
`initial size`	Initial size (i.e., number of reactions) of the RS
`execution length`	Number of iterations to perform during the fitness evaluation
p_c	Crossover probability
p_{in}	Probability of inserting a random reaction into a system
p_{rm}	Probability of removing a random reaction from a system
p_{ren}	Probability of regenerate randomly the current RS

Table 6: The parameters of an EvoRS system.

generated has a number of reactions chosen randomly and bounded by the `initial size` parameter.

11.1.5 *Minimization of Reaction Systems*

To simplify the individuals that are in the population, we introduce another genetic operator, that we call *minimization*, that reduces the number of reactions that comprises a RS without altering its behaviour, by eliminating reactions that have no impact on the results (equivalent of "dead code"). This operator is always applied and is based on the following observation: given a RS $\mathcal{A} = (S, A)$, the set $B \subseteq A$ of all maximal reactions in A can replace A without altering the behaviour of the system.

After the introduction of all the genetic operator we can recall all the parameters that are necessary for EvoRS (see Table 7).

There are four parameters related to the entire system (from the population size to the length of the execution of a RS). There is one parameter for crossover and three parameters for the three different types of mutation. The operators are applied as specified by Fig. 25. After the selection phase the more destructive kind of mutation, the *renewal*, is applied. The next operator to be applied is the crossover, followed by the two less destructive kind of mutation. The EvoRS cycle is ended by the application of the *minimization* operator.

11.1.6 *Properties of EvoRS*

Before describing the experimental results of EvoRS, it is interesting to note some of the advantages that they may have compared to other machine learning techniques. First of all, RS are not black boxes: their reactions, and the interactions between them, can be read and interpreted by humans. A second advantage with respect to other techniques, as GP, is that it is not necessary to define a set of functional and terminal symbols specific to the problem under exam. In fact, only the number of symbols used by the reactions is a necessary parameter. The operations are carried on by the reactions an the inter-

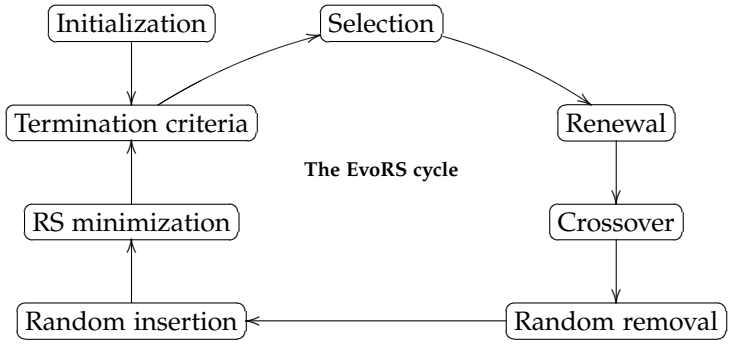


Figure 25: The execution cycle of EvoRS

actions between them. As currently defined, EvoRS also has a disadvantage: since for any input variable it is necessary to have a number of symbols comparable to the number of values that the variable can assume, we cannot use EvoRS on problems with continuous variable. An extension of EvoRS capable of handling this kind of problems is currently under study and will be presented in the future.

11.2 EXPERIMENTAL STUDY

To validate EvoRS, we compared it to some well-known machine learning methods on three different test problems. In this section we present the test problem used, then we will briefly introduce the other machine learning method used. Finally, the experimental settings are described.

11.2.1 *Test Problems*

We report the problems used in the experimental phase. All the considered test problems concern the classification of instances in two target classes.

The first problem is the well known k-even parity problem (see [114] for a definition of this problem). In the experimental phase we considered binary sequences of length from 2 to 8.

In the second test problem the task is to distinguish democrat votes from republican votes in the 1984 United States Congressional Voting Records. The data are the position taken by the representative on 16 key votes identifies by the Congress Quarterly Almanac. The dataset (available in WEKA [88]) has 435 instances and 17 attributes (where the last attribute is the target class). All the non-target attributes assume three possible values: *yes*, *no* and *unknown* (corresponding to a position that is neither *yes* or *no*). The target attribute assumes only two values: *republican* and *democrat*.

The last test problem regards diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images. Each of the patients is classified into two categories: *normal* and *abnormal*. The dataset has 267 instances that are described by 23 binary attributes (where the last attribute is the target class). Each row of the dataset represent an image of a different patient, where the attributes are the result of a processing that extracted 44 continuous features that describe the image, 22 of whom were selected in a subsequent phase. This dataset is available at the UCI Machine Learning Repository [73].

11.2.2 *Other studied techniques*

In the experimental phase, performances obtained with EvoRS have been compared with the results obtained considering different machine learning techniques. The machine learning techniques chosen to make the comparison are: feed-forward artificial neural networks trained with back-propagation (ANN), Bayesian networks (Bayes Net), naive Bayes classifier (Naive Bayes), radial basis function networks (RBF Net), and support vector machines using the sequential minimal optimization algorithm (SVM/SMO). For a complete description of these methods we refer to [33] for ANN, to [90] for Bayes Net, to [169] for Naive Bayes, to [158] for RBF net and to [38] and [160] for SVM/SMO. Furthermore, we compared EvoRS standard tree-based Genetic Programming (GP) [162].

11.2.3 *Experimental setting*

Due to the fact that both deterministic and non-deterministic machine learning methods are used in the experimental phase, it is important to explain how the experiments have been performed in order to produce a fair comparison of the results. Fitness was calculated as the number of correctly classified instances. We used the same set for both training and testing the algorithm. Hence, in these tests, we are not concerned with the issues of overfitting or generalization ability.

For all the non-evolutionary techniques we used the implementation of WEKA [88]. For the two Bayesan techniques (Bayes Net and Naive Bayes) only one run using the default WEKA's parameter setting has been performed. For ANN, RBF Net and SVM/SMO, 100 independent runs using the default WEKA's parameters have been performed. In each run we changed the seed used to generate random numbers in the algorithm.

For GP, 100 independent runs have been performed for each of the considered test problems. All the runs used populations of 100 individuals allowed to evolve for 100 generations. Tree initialization was performed with the Ramped Half-and-Half method [162]. The maximum initial depth was 4 for the SPECT and voting datasets, while it

was equal to k for k-even parity problems. The function set contained the three boolean operators *and*, *or*, and *not*. For the vote dataset they where interpreted as three-valued logic operators (i.e., the conjunction of a true or unknown value with an unknown value gives an unknown value, the negation of an unknown value remains unknown and the disjunction of a false or unknown value with an unknown value remains unknown). When the evaluation of a tree returned *unknown* instead of a specific class, its value was considered equal the one of the most represented class. The terminal set contained a number of variables equal to the number of attributes of each test problem. We have explicitly imposed functions and terminals to have the same probability of being chosen when a random node is needed. The reproduction (replication) rate was 0.1. Standard tree mutation and standard crossover (with uniform selection of crossover and mutation points) were used with probabilities of 0.1 and 0.9, respectively. The new random branch created for mutation has maximum depth 4. Selection for survival was elitist, with the best individual preserved in the next generation. The maximum tree depth is 17 except for the even parity problem, where the maximum tree depth is 2k.

For EvoRS, 100 independent runs allowed to evolve for 100 generations were performed. In all the run elitism was used. Hence, the individual with the best fitness was preserved across generations. For all the problems a population size of 50 individuals was used (half of the population size with respect to GP). The number of symbols was two times the number of input variables for the k-even parity problem and the SPECT problem. For the *vote* dataset we used two times the number of input variables plus 10 additional symbols. This variation was necessary since every input variable in the *vote* dataset can assume three values instead of two. The initial size of the RS was two times the number of input variables. For all the problems the execution length was 3. A crossover probability of 0.8 was chosen. The probability of a random insertion was fixed to 0.2. The probability of random removal was fixed to 0.2 and the probability of renewal was 0.1 for all the considered problems. Furthermore, elitism was used. It is important to note that a research of the best set of parameters has not been performed. Hence, these parameters need to be considered a guess based on a very limited number of test runs. A more detailed explanation of the parameters setting will be the focus of successive researches.

11.2.4 *Experimental Results*

In this section the results of the experimental phase are presented. Furthermore, an example of the structure of the solutions generated by EvoRS is presented. All the box plots presented have the end of the two whiskers representing one standard deviation above and below

the mean of the data. The cross represents the mean of the data. The fraction of successfully classified instances for GP and EvoRS is the one obtained after the last considered generation.

k-even parity. The results for the k-even parity problem are presented in Fig. 26. EvoRS and GP perform better than the other considered techniques for all the tested values of k. In particular, EvoRS is the best performer for values of k between 2 and 5, while GP performs better for values of k greater than 5. To test whether or not the differences in terms of fitness between the considered techniques are statistically significant, a test of statistical significance has been performed. First of all, a Kolmogorov-Smirnov (KS) test with a significance level of $\alpha = 0.05$ has been performed to test whether or not the fitness values are normally distributed. The (KS) test rejects the null hypothesis (hence the fitness values are not normally distributed) for all the k values and for all the non-deterministic techniques. Because the data are not normally distributed, a rank-based statistic has been used. The Wilcoxon rank-sum test for pairwise data comparison with a Bonferroni correction for the value of α is used under the alternative hypothesis that the samples do not have equal medians. The test has been performed by comparing EvoRS with the other techniques. We obtained that we can not reject the null hypothesis only in three cases: with $k = 2$ when comparing EvoRS with GP and RBF Net and also when $k = 5$ when comparing EvoRS with GP. In all the other cases the presented results have a statistically significant difference.

vote dataset. The results for the vote problem are presented in Fig. 27(a). In this case, ANNs is the best performer with the 99% of correctly classified instances. SVM/SMO produce a 97% of correctly classified instances followed by EvoRS and RBF Net with 94%. It is important to underline that EvoRS is a newly defined evolutionary technique, hence the tuning of its parameters is quite difficult. Nonetheless it produces results that are comparable with the ones produced by other well-known (and well studied) machine learning techniques. Also for this problem the same statistical tests as for the k-even parity have been performed. The null hypothesis cannot be rejected only when comparing EvoRS with RBF Net.

spect dataset. Results for the last dataset, i.e., the SPECT dataset, are presented in Fig. 27(b). In the SPECT problem EvoRS performance is lower than the other contenders except Naive Bayes. The same statistical tests performed for the other test problems have been considered for this problem. From the results of the Wilcoxon test we obtained that, for this problem, the null hypothesis has been always rejected. So difference in performance of the different studied methods is statistically significant. Nonetheless EvoRS performances are

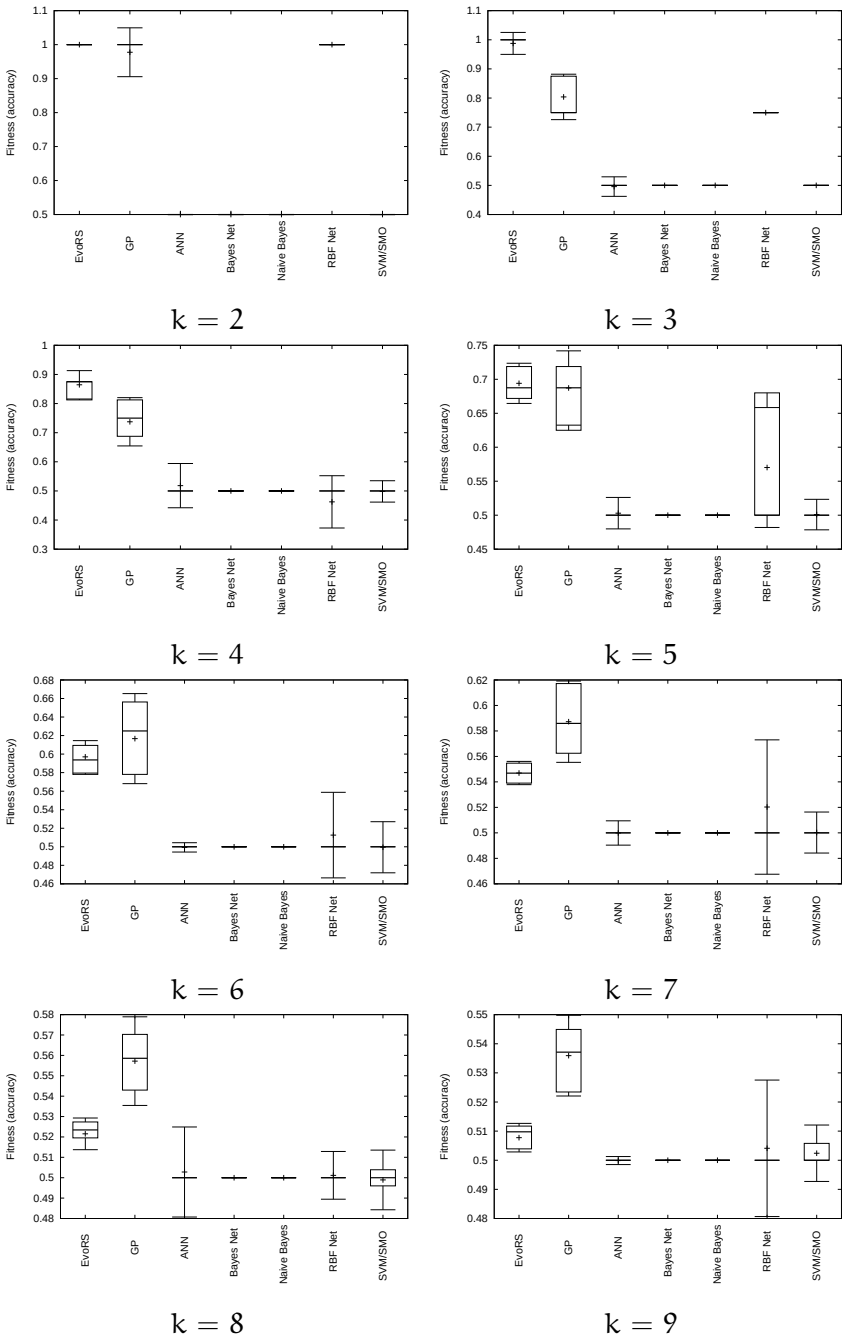
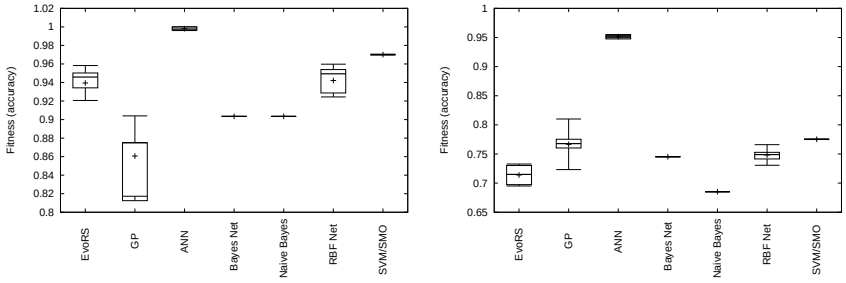


Figure 26: The results for the k-even parity problem. The two whiskers represent one standard deviation above and below the mean of the data.



Vote Dataset (a) Spect Dataset (b)

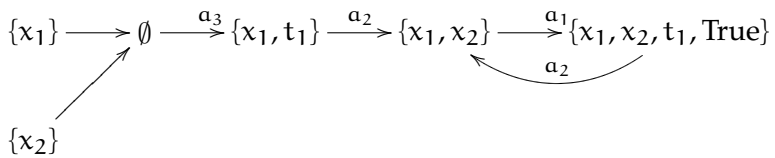
Figure 27: The results for the *vote* and SPECT datasets. The two whiskers represent one standard deviation above and below the mean of the data.

non very low compared to the other techniques. We do not consider this a negative result for a newly-developed and still-to-be-tuned evolutionary algorithm.

some individuals found by evors. Contrarily to many other machine learning techniques and similarly to GP, EvoRS provides models that are directly interpretable by humans. Let us consider, for instance, a solution generated by EvoRS for the even parity problem with $k = 2$. It is composed by the following three reactions:

$$a_1 = (\{x_1, x_2\}, \{t_1\}, \{x_1, x_2, t_1, \text{True}\})$$
$$a_2 = (\{t_1\}, \emptyset, \{x_1, x_2\}) \qquad a_3 = (\emptyset, \{x_1, x_2, t_1\}, \{x_1, t_1\})$$

Where x_1 and x_2 are the input symbols, True is the output symbols representing *true* and t_1 is a temporary symbol. The dynamic evolution of the system can be represented by the following graph, where the nodes are states and the transitions between them are represented by the edges labeled with the reactions that are activated:



Recall that the test were performed with an `execution length` of three. Hence, both $\{x_1, x_2\}$ and $\emptyset$ reach a state containing True in no more than three steps. But $\{x_1\}$ and $\{x_2\}$ reach it in four steps, too many to obtain True as output. This example shows how the solutions generated achieve the goal of producing the correct output not by rote memorization of the inputs but by producing a complex interactions between the different reactions.

11.3 PARAMETER TUNING

After having affirmed the suitability of EvoRS as an EA, it is necessary to study how its parameters influences its performances and behaviour. This is an essential study in every new EA techniques, since it allows to comprehend how the algorithm behave and it is the first step to allow a new EA to be effectively used.

11.3.1 *Experimental Settings*

The values of the EvoRS parameters used in our experiments are reported in Table 7.

Parameter	Values tested
`population size`	20
`number of symbols`	1, 1.5, 2, 2.5, and 3 times the number of input and output symbols
`initial size`	1, 1.5, 2, 2.5, and 3 times the number of input and output symbols
`execution length`	From 2 to 5
p_c	0.8
p_{in}	0.2
p_{rm}	0.2
p_{ren}	0.1

Table 7: The parameters of an EvoRS system and the values tested.

To study different combinations of the parameters that characterize an EvoRS we have considered different test problems:

- the n-multiplexer problem [113] with $n = 3$ and $n = 6$;
- the n-parity problem [114] with $n = 4$, $n = 5$ and $n = 6$;
- the n-majority problem [14] with $n = 3$ and $n = 5$;

For each problem, 100 different parameters settings have been considered and, for every setting 30 runs have been performed. Thus, considering 7 test problems instances, a total of 21000 runs of EvoRS have been performed.

The parameters that were studied are the ones that are typical of EvoRS, i.e. the ones that have no equivalent in other EAs, and thus there is no "rule of thumb" that can be inherited for suitably setting their values. While it is possible that the parameters common with others EA do not have the same influence on the behaviour of EvoRS, the fact that they behave quite similarly across many different EA is a good indication that the study should begin from parameters peculiar to EvoRS. Thus, we limited our study to the `number of symbols`, the `initial size` and the `execution length`. While the `number of symbols` can be thought as similar to the choice of functional or terminal symbols in GP, in EvoRS these symbols, except for input and

output symbols, do not carry any semantics. Only the `initial size` of RS can have a counterpart in the tree size of GP.

In all the tests, the populations size was set to 20 individuals (while the population size is small, a set of preliminary runs with size 50 showed that its influence on the results is limited), a tournament selection of size 2 was chosen as selection method. The crossover probability was set to 0.8, the random insertion and removal probabilities were set to 0.2 and the renewal probability to 0.1. The `initial size` was k times the number of input plus output symbols, where k varied between 1 and 3 with a step of 0.5. The same range was used for the `number of symbols`. The `execution length` was varied from 2 to 5 time steps. All the test were performed using a Python implementation of EvoRS.

11.3.2 *Experimental Results*

To show the influence of a particular parameter, we present the results in the following way: for a particular parameter, we find the best configuration of the other two parameters with respect to the average best fitness. By doing this, we find what is the best possible fitness obtainable with that particular value of that particular parameter. The results for the 5-majority problem are presented in Fig. 28 and the ones for 6-multiplexer are presented in Fig. 29. The results for 3-majority and 3-multiplexer are not outlined since in almost all the cases the algorithm finds the optimal solution. The results for 4-parity, 5-parity, and 6-parity are presented in Fig. 30, Fig. 31, and Fig. 32 respectively. In all figures, the box represents the 25% and the 75% percentile, while the whiskers represent the average fitness plus or minus the standard deviation.

Furthermore, we present the correlation between the different parameters and the fitness of the best solutions in Table 8. To statistically validate the results, we have performed a Mann-Whitney U-test with the alternative hypothesis that the probability of having an observation of one population exceeding an observation from the other population is not 0.5. This statistical test was chosen since the data obtained were not normally distributed, as confirmed by a Kolmogorov-Smirnov test. The p-values obtained are presented in Table 9.

11.3.3 *Discussion*

In this section we discuss the effect that different values of the considered parameters produce on the evolutionary process. In particular, for each problem, we discuss how the initial size, the number of symbols and the execution length affect the fitness of the solutions.

Test Problem	Initial Size	Number of Symbols	Execution Length
3-*majority*	0.15148	−0.34724	−0.11884
5-*majority*	0.15323	−0.48272	−0.15576
3-*multiplexer*	0.15451	−0.33660	−0.10644
6-*multiplexer*	0.18661	−0.50796	−0.16092
4-*parity*	0.15028	−0.73576	−0.09214
5-*parity*	0.13698	−0.77359	−0.12872
6-*parity*	0.10845	−0.79002	−0.11516

Table 8: The correlation coefficients between the three considered parameters on different problem and the fitness.

Test Problem	`symbols` = 1.5	`symbols` = 2	`symbols` = 2.5	`symbols` = 3
3-*majority*	0.657380	0.375044	0.375044	0.505859
5-*majority*	0.004128	0.000033	0.002439	0.000017
3-*multiplexer*	1.000000	0.657380	0.824496	0.375044
6-*multiplexer*	0.790147	0.594560	0.487138	0.496452
4-*parity*	0.000000	0.000000	0.000000	0.000000
5-*parity*	0.000002	0.000000	0.000000	0.000000
6-*parity*	0.000000	0.000000	0.000000	0.000000

Table 9: The p-values obtained by comparing the best results obtained with `symbols=1` with the best results obtained with the other values of `symbols` using the Mann-Whitney U-test.

11.3.3.1 *Initial Reaction System Size*

The initial reaction system size seems to have a weak effect on the performance of EvoRS. Considering the box plots related reported in Fig. 28 and Fig. 29, it is possible to see that using different values of the parameters, we produce comparable solutions, that present a difference in terms of fitness that is not statistically significant. The situation slightly changes considering the box plots reported in Fig. 30, in Fig. 31 and in Fig. 32. In this case, it seems that a value of the considered parameter (2.5) produces solutions that present a different fitness with respect to the fitness values obtained with the other values of the parameter, but this difference is not statistically significant. To summarize, the initial EvoRS size seems to have a faint effect on the fitness of the solutions. This fact is also strengthened by the values of the correlation coefficients reported in Table 8. The marginal effect produced by the initial EvoRS size is desirable; it suggests that, at least for the considered test problems, it is not necessary to have an EvoRS with a high number of reactions to find a solution with a "good" fitness value.

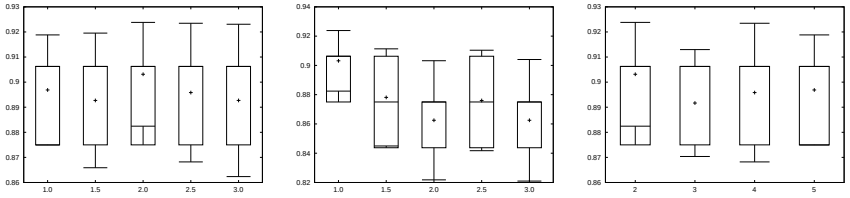
11.3.3.2 *Number of Symbols*

The results obtained considering the effects of different number of symbols is somewhat surprising. The values of the correlation coefficients reported in Table 8 show a negative correlation between the

number of symbols of the EvoRS and the fitness of the solutions. Because we have considered maximization problems, this fact suggests that better fitness values could be reached using a "small" number of symbols. While this result may be counterintuitive, it has a very simple explanation: using a large number of symbols enhances the number of possible reactions that may occur. In turn, this can lead to obtain the *true* value in a large number of fitness cases. But, obviously, the *true* value is not the correct value for all the fitness cases. The effect of the number of symbols used is particularly clear when the three instances of the n-parity problem are considered. As shown in Fig. 30, Fig. 31 and Fig. 32, the number of symbols has a strong impact on the fitness of the solutions, with a fitness that increases steadily with the decrease in the number of symbols. This difference in terms of fitness is statistically significant as showed in Table 9, and suggests that the EvoRS practitioners should pay attention in choosing the number of symbols of the system with parsimony. The impact that the number of symbols has on the fitness of the best solutions is also visible in Fig. 28 and in Fig. 29. As it is possible to see, the best solutions for the considered test problems have been obtained with "small" values of this parameter. Table 9 also reports the p-value obtained by comparing the best results obtained with `symbols=1` with the best results obtained with the other values of `symbols` for all the considered test problems.

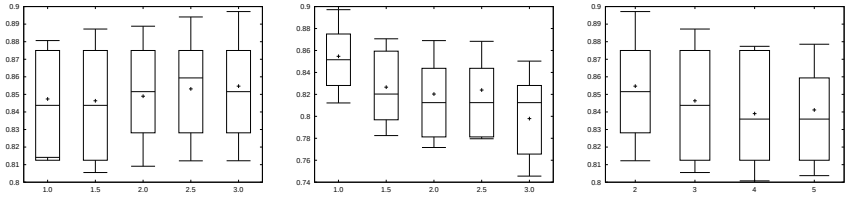
11.3.3.3 *Execution Length*

The execution length seems to have a weak effect on the performance of EvoRS. Considering the box plots reported from Fig. 28 to Fig. 32, it is possible to see that different values of the parameter result in comparable solutions, that present a difference in terms of fitness that is not statistically significant. This fact is also confirmed by the values of the correlation coefficients reported in Table 8. It is important to recall that the execution length is defined as the number of time steps in which the dynamic of an EvoRS is observed. In particular, if the symbol *True* appears in one of the observed states of the RS then the output of the system is *true*, otherwise (i.e., the symbol *True* has not appeared in the first `execution length` time steps), the output of the RS is *false*. From the definition, it seems that this parameter should have an important effect on the fitness of the best solution, because it allows a RS to reach (or not reach) the *True* value in a certain time steps. Results arising from the experimental phase should be taken with caution. The weak effect that the execution length has on the fitness of the final solution, may be due to the fact that, for the considered test problems, the system already reach the *True* value for "low" values of the parameter. Further investigations about the role of this parameter are needed. In particular, the effect of this parameter could emerge when more complex problems are considered.



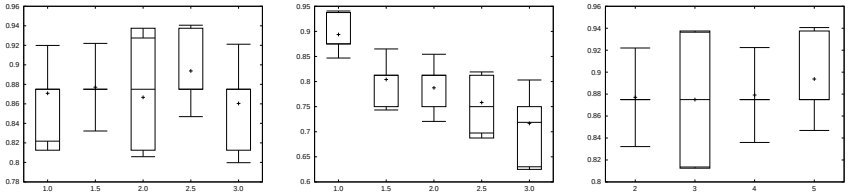
initial size (a) number of symbols (b) execution length (c)

Figure 28: The box plot of the results for the 5-majority problem



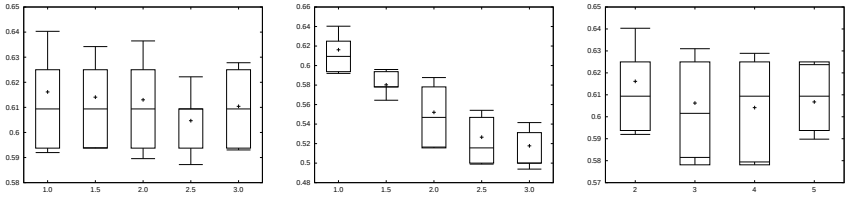
initial size (a) number of symbols (b) execution length (c)

Figure 29: The box plot of the results for the 6-multiplexer problem



initial size (a) number of symbols (b) execution length (c)

Figure 30: The box plot of the results for the 4-parity problem



(a) (b) (c)

Figure 31: The box plot of the results for the 5-parity problem

11.4 FURTHER REMARKS

In this chapter a new biologically inspired evolutionary algorithm, called evolutionary reaction systems (EvoRS), has been defined. It is based on reaction systems, an expressive and powerful computational formalism inspired by chemical reactions, recently defined by Rozenberg and coworkers. We have shown that the performances of EvoRS

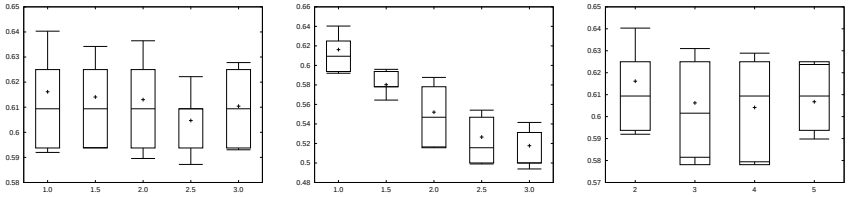


Figure 32: The box plot of the results for the 6-parity problem

are comparable, and in some cases even better, than the ones of other well known machine learning algorithms (including Bayesian methods, neural networks, support vector machines and standard genetic programming) on a set of case studies including real-life applications.

Since EvoRS is a new GP variant, parameter tuning plays an important role for its successful application. Given the stochastic nature of evolutionary techniques, a large literature has faced this problem. All the practitioners of evolutionary techniques have had to deal with this problem, and they know how the choice of parameters' values represents one of the most important and delicate phases prior to the execution of the algorithm itself. A wrong or inaccurate choice of the parameters' values may undermine the subsequent search process, leading the algorithm to find poor quality solutions.

Thus, the parameter tuning problem for EvoRS has been studied. This is an important step towards the application of this new evolutionary technique. The analysis involved the parameters that characterize an EvoRS: (1) the initial reaction system size, (2) the number of symbols, and (3) the execution length. With respect to the considered test problems, we can conclude that: (1) the initial reaction system size has a weak impact on the fitness of the best solutions returned by EvoRS. This point deserves a further analysis, in particular considering more test problems; (2) the number of symbols has an important effect on the fitness of the best individual: making a wrong choice for this parameter results in poor quality solutions. It is interesting that better solutions have been found when a "small" number of symbols has been used; (3) the execution length seems to show a weak effect on the best solution's fitness. This point needs further investigation, as we believe that this parameter could play an important role when more complex problems are considered. Once we achieve a complete understanding about the effects of all the parameters, our goal will be the development of a parameter control method, thus defining a more dynamical EvoRS framework, where the values of the parameters will be adapted during the evolution.

Part IV

ASYNCHRONOUS CELLULAR AUTOMATA

12

INTRODUCTION TO CELLULAR AUTOMATA

Cellular automata (CA) are a formal model used in many scientific fields [61, 34, 35]. They are composed of a grid of identical finite state automata, or cells, that update their own state according to a local rule that depends only on a fixed number of neighbour automata. The update of all the cells happens at the same time. The synchronicity is then a main characteristic of classical CA. Several formal properties concerning the dynamical behaviour of classical CA has been extensively studied in the recent past (for an up-to-date bibliography see, for instance, [120, 49, 2, 47, 48, 78, 1, 32, 31, 46]).

When CA are used for the modeling of various physical phenomena the assumption of synchronicity can be a problem. In fact, a synchronous behaviour is a quite rare event in nature. To more closely simulate real systems, the introduction of asynchronicity in models has been considered (think, for instance of the model from [6] used to simulate biochemical processes in living cells). In the last 20 years empirical studies focused on asynchronous updating schemes [21, 27, 62, 176, 145, 167]. According to which updating policy is chosen, the behavior of the ACA under consideration can be very different. This empirical observation motivated successive formal studies. However they were not as extensive as the ones for synchronous CA and focused mainly on particular examples [76, 166] or classes of probabilistic cellular automata [63, 64] (i.e., CA where every cell updates with a certain probability p).

In Chapter 13 we consider a fully asynchronous scheme in which, as in a continuous time process, two cells are never updated simultaneously (or, equivalently, one and only one cell is updated at each time step). A sequence of integers, named update sequence, specifies the cell which is updated at each time.

In classical CA the behaviour of the system is studied by some formal properties which give important information either on the CA global map (e.g., injectivity, surjectivity) or the CA dynamics (e.g., transitivity, sensitivity to initial conditions). These notions cannot be directly used for studying asynchronous CA. Here we suitably adapt them to the asynchronous case and we study them.

Motivated by the fact that update sequences are often ruled by real processes, we also study the stability of some properties with respect

to perturbation on the update sequence. Indeed, the computational models used to simulate these real processes may give update sequences with errors.

Another aspect of the study of fully asynchronous CA is the ability to perfrom fully asynchronous computation. This is quite interesting since, in some sense, it represents the worst case from the complexity point of view when a network of processors is considered. The ACA setting is also similar to token ring networks of which it represents the case of single token, lattice network and finite automata on nodes. A pletora of protocols and algorithms have been designed in the recent past for token ring networks, thus, we give the conditions on the token distribution which enable anetwork of finite automata (ACA) to cooperate to simulate step by step an universal Turing machine.

Furthermore, it is well-known that CA are capable of universal computation (see [154] for an up-to-date survey). Various mechanisms have been introduced to show how ACA can emulate classical CA or circuits [145, 148, 123, 218]. As a consequence of these results, Turing universality of ACA is also proved.

In Chapter 14 we focus on the way by which an ACA simulates a Turing Machine (TM for short). We analyse two different modes: strict simulation and scattered strict simulation. Roughly speaking, strict simulations require that the ACA exactly reproduces the steps of the TM (up to some encoding) admitting that at most a possibly unbounded amount time is wasted between two steps of the TM. The latter mode is essentially the same as strict simulation but considers the case that only a subset of cells can be used to perform the simulation, the others being "inactive" because of failure for example. We characterize all updating schemes allowing these two simulation modes (Theorems 14.1.3 and 14.1.5). Moreover, we show that in both cases, the time slowdown due to asynchronicity is quadratic w.r.t. the running time of the TM under simulation (Propositions 14.1.4 and 14.1.6).

The quadratic slowdown is essentially due to the fact that the token travels back and forth through the network passing through a lot of nodes (cells) that are not interested by the calculation (recall that we are on a lattice and that only cells having the token are allowed to update their state). Introducing some degree of randomness in the token distribution improves complexity to linear time at the price of low probability of obtaining the correct result (Proposition 14.2.1). This complexity vs. probability tradeoff seems to be necessary, indeed, maximising the probability of obtaining the correct results pumps up complexity to cubic time w.r.t. the time used by the original TM (Proposition 14.2.2).

The construction of sequences that allow quadratic simulations at a first glance might seem artificial, classifying our results in the domain of computability but of no use in practical simulations. Indeed,

for example, consider the program described in [6]. It is used to simulate biochemical processes in living cells. It is essentially based on an ACA (although the authors do not clearly state this) which represents proteins (and other chemicals) by particles. The current configuration is updated in two steps: diffusion and collision/reaction. In the diffusion step a particle is randomly chosen, a direction is randomly chosen and then the particle makes a move in this direction. This system can be seen as an ACA with the sequence of activations generated by a random walk. Since a random walk passes with probability 1 infinitely many time through every cell, the system described in [6] is capable of universal computation.

Finally, in Chapter 15, we study a more general setting relaxing the synchronicity constraint. Indeed, at each time step, the subset of cells to be updated is extracted using a probability measure μ on subsets of $\mathbb{Z}$. Clearly, one should put some conditions on μ to grant both a non-trivial behavior and substantial asynchronicity. We called these constraints "fairness conditions" (see Definition 15.1.2).

This new point of view asked to adapt the classical notions about dynamical behavior such as injectivity, surjectivity, expansivity, etc.. We proved a somewhat surprising result, namely, μ-almost surely surjective CA are μ-almost surely injective and *vice-versa*. Curiously, this result is much similar to the case of fully-ACA (ACA in which only one cell is allowed to be updated per time step [126], the ones studied in Chapters 13 and 14) and much different different from the classical CA setting where injective CA are surjective but the *vice-versa* is false.

We also proved that almost sure injectivity has also a combinatorial facet through the classical notion of diamond (Proposition 15.1.4). The existence of diamonds has also been related to almost sure expansivity (Proposition 15.1.5).

The study of $\mathfrak{m}$-ACA has just been started over and as usual there are more questions than answers. For instance, in the classical CA setting, it is well-known that a CA is either sensible to initial conditions or it has equicontinuity points. It would be interesting to investigate if this dichotomy is still true or which weakened form it can take.

The remaining part of this Chapter is focused on the introduction of some preliminary notions necessary to understand the remaining Chapters 13 throught 15.

12.1 PRELIMINARY NOTIONS

For all $i, j \in \mathbb{Z}$ with $i \leqslant j$ (resp., $i < j$) we denote by $[i, j]$ the set $\{k \in \mathbb{N} | \, i \leqslant k \leqslant j\}$ (resp., $[i, j) = \{k \in \mathbb{N} | \, i \leqslant k < j\}$). Also, $\forall i \in \mathbb{Z}$, let $(i, +\infty) = \{i, i+1, \ldots\}$ and $(-\infty, i) = \{i, i-1, \ldots\}$. The set of positive integers (resp., reals) is denoted by $\mathbb{N}_+$ (resp., $\mathbb{R}_+$). Given a set X, $\mathcal{P}(()X)$ denotes the collection of subsets of X. As usual, for a finite set X, $|X|$ is the number of its elements.

Let Σ be a finite alphabet. A *configuration* is a function from $\mathbb{Z}$ to Σ. As usual, for any two sets A and B, A^B is the set of all functions from B to A. The *configuration set* $\Sigma^{\mathbb{Z}}$ is usually equipped with the metric d defined as follows:

$$\forall x, y \in \Sigma^{\mathbb{Z}}\ d(x,y) = 2^{-n}$$
$$\text{where } n = \min\{i \in \mathbb{N} \mid x_i \neq y_i \text{ or } x_{-i} \neq y_{-i}\}$$

The set $\Sigma^{\mathbb{Z}}$ is a compact, totally disconnected and perfect topological space (i.e., it is a Cantor space) (see, for example, [108], [144] or [118] for an introduction to these topological concepts). For any pair $i, j \in \mathbb{Z}$, with $i \leqslant j$, and any configuration $x \in \Sigma^{\mathbb{Z}}$ we denote by $x_{[i,j]}$ the word $x_i \cdots x_j \in A^{j-i+1}$. Similarly, for every $u \in A^{\ell}$ and for every $i, j \in [0, \ell)$, $u_{[i,j]} = u_i \dots u_j$ is the portion of a word inside $[i, j]$. In both the previous notations, $[i, j]$ can be replaced by $[i, j)$, with the obvious meaning. A configuration x is said to be a-finite for some $a \in \Sigma$ if the number of positions i with $x_i \neq a$ is finite. In the first one it can also be replaced by either $[i, \infty)$, or (i, ∞), or $(-\infty, j]$ or $(-\infty, j)$ with the obvious meaning. By $|u|$ we will denote the length of any word u.

A 1D CA is a structure (Σ, λ, r) where Σ is the *alphabet* or *set of states*, $r \in \mathbb{N}$ is the *radius*, and $\lambda : \Sigma^{2r+1} \to \Sigma$ is the *local rule* of the automaton. The local rule λ induces a *global rule* $F : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ defined as follows:

$$\forall x \in \Sigma^{\mathbb{Z}},\ \forall i \in \mathbb{Z}, \qquad F(x)_i = \lambda(x_{i-r}, \dots, x_i, \dots, x_{i+r})\ .$$

Note that F is a uniformly continuous map with respect to the metric d. For any CA, the pair $(\Sigma^{\mathbb{Z}}, F)$ is a (discrete time) dynamical system. From now on, for the sake of simplicity, we identify a CA with the dynamical system that it induces or even with its global rule F.

The *orbit* of a configuration $x \in \Sigma^{\mathbb{Z}}$ is the sequence

$$\gamma_x = (x, F(x), (F \circ F)(x), \dots)$$

associating with each time step t the configuration $\gamma_x(t) = F^t(x)$ of the CA at that time.

A local rule $\lambda : \Sigma^{2r+1} \to \Sigma$ is *rightmost-permutive* (resp. *leftmost-permutive*) (resp. *center-permutive*) iff for any $w \in \Sigma^{2r}$ and any $b \in \Sigma$ there exists a unique $a \in \Sigma$ such that $\lambda(aw) = b$ (resp. $\lambda(wa) = b$) (resp. $\lambda(w_{[0,r)}\, a\, w_{[r,2r)}) = b$).

Let $\mathcal{T}$ be a monoid of continuous functions from $\Sigma^{\mathbb{Z}}$ to $\Sigma^{\mathbb{Z}}$.

The family $\mathcal{T}$ is *sensitive to initial conditions* (or, simply, *sensitive*) if there exists $\varepsilon > 0$ such that for any $x \in \Sigma^{\mathbb{Z}}$ and any $\delta > 0$, there is an element $y \in \Sigma^{\mathbb{Z}}$ with $d(x, y) < \delta$ such that $d(T(x), T(y)) \geqslant \varepsilon$ for some $T \in \mathcal{T}$.

The family $\mathcal{T}$ is said to be *positively expansive* (or, briefly, *expansive*) if there exists a constant $\varepsilon > 0$ such that for every pair of distinct elements $x, y \in \Sigma^{\mathbb{Z}}$, we have $d(T(x), T(y)) \geqslant \varepsilon$ for some $T \in \mathcal{T}$.

Also, $\mathcal{T}$ is said to be *equicontinuous* at the point $x \in \Sigma^{\mathbb{Z}}$ if $\forall \varepsilon > 0\ \exists \delta > 0$ such that $\forall y \in \Sigma^{\mathbb{Z}}$, $d(x,y) < \delta$ implies that $\forall T \in \mathcal{T}$, $d(T(x),T(y)) < \varepsilon$. The family $\mathcal{T}$ is equicontinuous if it is equicontinuous at every point.

A CA with global rule F is said to be sensitive (resp. expansive) (resp. equicontinuous) iff its family of functions $\{Id, F, F^2, \ldots\}$ is sensitive (resp. expansive) (resp. equicontinuous), where Id is the identity map on $\Sigma^{\mathbb{Z}}$.

A σ-algebra over a topological space X is a non-empty collection of subsets of X closed under countable union and complementation. The Borel σ-algebra on X is the smallest σ-algebra containing all the closed and the open sets of X. A measure μ is a function from a σ-algebra on X to $\mathbb{R}_+ \cup \{+\infty\}$ such that μ is countably additive. A measure μ is said to be a probability measure if $\mu(X) = 1$. For a given set X of outcomes, a σ-algebra on X represents the set of events and, once introduced a probability measure μ on the σ-algebra, the probability of an event E is $\mu(E)$.

For any $i \in \mathbb{Z}$, the *principal ultrafilter* $\mathcal{U}_i$ of i is the collection of all subsets of $\mathbb{Z}$ containing i. The complement of $\mathcal{U}_i$ is denoted by $\overline{\mathcal{U}_i}$. From now on, we consider $\mathcal{P}(\mathbb{Z})$ equipped with a topology for which the principal ultrafilters are clopen (i.e., closed and open) sets.

A *Turing Machine* $\mathcal{M}$ is a 7-tuple $(Q, \Sigma, \Gamma, b, \delta, q_0, F)$, where Q is the set of *states*, $\Sigma \subset \Gamma$ is the *input alphabet*, Γ is the *working alphabet* and $b \in \Gamma \setminus \Sigma$ is the *blank symbol*. The map $\delta : Q \times \Gamma \mapsto Q \times \Sigma \times \{L, R\}$ is the *transition function*, where L and R denote the left and right movements of the head, $q_0 \in Q$ is the *initial state* and $F \subseteq Q$ the *set of final states* (see [94], for an introduction on this subject). An *instantaneous configuration* c of $\mathcal{M}$ is a triple (T, q, p), where $T \in \Gamma^{\mathbb{Z}}$ is the content of the tape, $q \in Q$ is the current state of $\mathcal{M}$ and $p \in \mathbb{Z}$ is the position of its head.

Denote b^ω and ${}^\omega b$ the infinite repetition of b towards right and left, respectively. A *run* of $\mathcal{M}$ on the initial input $x \in \Sigma^*$ is the sequence $R_t = \{(T_t, q_t, p_t)\}_{t\in\mathbb{N}}$ where for $t = 0$ $(T_0, q_0, p_0) = ({}^\omega b x b^\omega, q_0, 0)$ and the first symbol of x is at position 0, while for any $t \in \mathbb{N}$, $(T_{t+1}, q_{t+1}, p_{t+1})$ is the instantaneous configuration of $\mathcal{M}$ at time $t+1$, where T_{t+1} is equal to T_t except that in position p_t in which the symbol $(T_t)_{p_t}$ is replaced by the symbol s with $(q_{t+1}, s, X) = \delta(q_t, (T_t)_{p_t})$, and $p_{t+1} = p_t + 1$ if $X = R$ and $p_{t+1} = p_t - 1$ if $X = L$. For any given run, note that if $q_t \in F$ at some time $t \in \mathbb{N}$, then $R_k = R_t$, for any $k > t$. In other words the computation halts on the instantaneous configuration R_t and the output is the word from the head of the TM to the first blank symbols on the tape, excluded.

13

FULLY-ASYNCHRONOUS CA

In this chapter the simplest case of asynchronous CA are introduced and their dynamical properties are studied. This kind of CA have the strongest possible asynchronicity: only one cell is updated at every time step.

The chapter is organized as follows. In Section 4.2 some basic notions are recalled. In Section 13.1 fully asynchronous CA and some of their properties are defined. In Section 13.2 the focus is on the study of the dynamical behaviour of asynchronous CA. Finally, in Section 13.3 some concluding remarks and future works are presented.

13.1 DEFINITION OF FULLY ASYNCHRONOUS CA

In this section we introduce *fully asynchronous cellular automata* (fully-ACA). Furthermore we reformulate the classical notions of surjectivity and injectivity for fully-ACA.

Let $\lambda : \Sigma^{2r+1} \to \Sigma$ be a local rule of radius r. We consider the following asynchronous updating for λ. At each time t the local rule λ is applied on one and only one cell. A sequence $(\tau_t)_{t>0}$ of integers specifies the index $\tau_t \in \mathbb{Z}$ of the cell which is updated at the time step $t > 0$.

Definition 13.1.1. A fully-ACA is a quadruple $(\Sigma, \lambda, r, \tau)$ where Σ is a finite alphabet, $\lambda : \Sigma^{2r+1} \to \Sigma$ is the local rule of radius $r \in \mathbb{N}$ and $\tau = (\tau_t)_{t>0}$, with $\tau_t \in \mathbb{Z}$ is a sequence of cell positions.

Every fully-ACA $C = (\Sigma, \lambda, r, \tau)$ induces a global behaviour described as follows. For any fixed $k \in \mathbb{Z}$, let $F_k : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ be the map such that:

$$\forall x \in \Sigma^{\mathbb{Z}},\ \forall i \in \mathbb{Z}, \quad F_k(x)_i = \begin{cases} \lambda(x_{i-r}, \ldots, x_i, \ldots, x_{i+r}) & \text{if } i = k \\ x_i & \text{otherwise} \end{cases}$$

For any $t \geqslant 0$, C transforms the generic configuration $x \in \Sigma^{\mathbb{Z}}$ at the time step t into the configuration $F_{\tau_t}(x)$ at the time step $t+1$. The dynamics of a fully-ACA is described by the family of functions $\mathcal{T}_C = \{\mathrm{Id}, F_{\tau_1}, F_{\tau_2} \circ F_{\tau_1}, \ldots, F_{\tau_t} \circ \cdots \circ F_{\tau_1}, \ldots\}$. Remark that all the elements from $\mathcal{T}$ are a continuous maps w.r.t. d. The *orbit* of a configuration

$x \in \Sigma^{\mathbb{Z}}$ is the sequence $\gamma_x = (x, F_{\tau_1}(x), (F_{\tau_2} \circ F_{\tau_1})(x), \ldots)$ associating with each time step t the configuration $\gamma_x(t) = (F_{\tau_t} \circ \ldots \circ F_{\tau_1})(x)$ of the fully-ACA at that time.

In many situations we are interested in properties that do not depend on the particular sequence τ. In those cases, we will refer to the class $C = (\Sigma, \lambda, r)$ of all fully-ACA in which the sequence τ is not fixed and we will call *uninstantiated fully-ACA* such a class. In the sequel, when no misunderstanding is possible, the term "uninstantiated" will be omitted.

Injectivity and surjectivity are important properties for classical CA. Their adaptation to fully-ACA takes into account the whole family of functions $\{F_k\}_{k\in\mathbb{N}}$.

Definition 13.1.2. A fully-ACA $C = (\Sigma, \lambda, r)$, is said to be *α-injective* if $\forall k \in \mathbb{N}, \forall x, y \in \Sigma^{\mathbb{Z}}, \quad x \neq y \Rightarrow F_k(x) \neq F_k(y)$

Definition 13.1.3. A fully-ACA $C = (\Sigma, \lambda, r)$, is said to be *α-surjective* if $\forall k \in \mathbb{N}, \forall x \in \Sigma^{\mathbb{Z}}, \quad F_k^{-1}(x) \neq \emptyset$

In other words, an uninstantiated fully-ACA is injective (resp. surjective) if every instantiated fully-ACA has all the global functions F_k injective (resp. surjective). Furthermore injectivity (resp. surjectivity) of all the global functions implies α-injectivity (resp. α-surjectivity).

In classical CA injectivity implies surjectity [127]. A stronger relation holds between α-injectivity and α-surjectivity:

Proposition 13.1.1. *Let $C = (\Sigma, \lambda, r)$ be a fully-ACA. Then, the following statements are equivalent:*

i) C *is α-injective.*

ii) C *is α-surjective.*

iii) f *is center permutive.*

Proof. i)⇔ii). Fix $k \in \mathbb{Z}$. Consider the function $h : \Sigma^{2r+1} \to \Sigma^{2r+1}$ defined as $h(u) = F_k(x)_{[k-r,k+r]}$ where x is any configuration such that $x_{[k-r,k+r]} = u$. Then, F_k injective $\Leftrightarrow$ h is injective and F_k surjective $\Leftrightarrow$ h is surjective. Since the domain and the codomain of h are equal and of finite cardinality h injective $\Leftrightarrow$ h is surjective.

i)⇔iii). Suppose that C is α-injective and fix $u, v \in \Sigma^r$ and $b \in \Sigma$. Since h is injective there exists only one $c \in \Sigma$ such that $h(ucv) = ubv$ and in particular $\lambda(ucv) = b$. Thus λ is center permutive. Vice versa, if λ is center permutive, then every block $w = ubv \in \Sigma^{2r+1}$ has an unique preimage $h^{-1}(w) = ucv$ for a certain $c \in \Sigma$. □

Remark. Unlike classical CA, fully-ACA defined by a rightmost/leftmost permutive local rule are not necessarily α-surjective. As an example consider the local rule $f : \{0,1\}^3 \to \{0,1\}$ such that $f(a, b, c) = a$. The rule is leftmost permutive but not center permutive, hence it is

not α-surjective. In fact, the set of preimages of the configuration $\ldots 0101010101 \ldots$ is empty (since $f(0,b,c) = 0$ and $f(1,b,c) = 1$, one application of any global function to any configuration will produce either 00 or 11 in its image).

In this chapter we also ask whether a property that holds for a certain fully-ACA $C_a = (\Sigma, \lambda, r, \tau)$ also holds for all fully-ACA defined by a sequence b "similar" to a. Expressing the similarity by the distance d may have some drawbacks. First of all, when changing the updating sequence the conservation of a property also depends on the value of δ such that $d(a,b) < \delta$. Moreover, the conservation depends only on a prefix of a. Here, we adopt the following similarity notion: a sequence b is similar to a iff $b \in \mathrm{Edit}(a)$, where $\mathrm{Edit}(a)$ is the set of all the sequences that can be obtained from a applying only a finite number of deletions, insertions and substitutions of elements. This is a direct generalization of the *Levenshtein distance*, an *edit distance* used in approximate string matching, for finite strings [124]. We can then define the stability of a property P.

Definition 13.1.4. A property P is *stable* for a fully-ACA $C_\tau = (\Sigma, \lambda, r, \tau)$ if P holds for all the fully-ACA $C_{\tau'} = (\Sigma, \lambda, r, \tau')$ with $\tau' \in \mathrm{Edit}(\tau)$.

In other words, a property P which holds for a fully-ACA C_a is stable iff it still holds in presence of a finite number of modifications in the sequence a.

Remark. Define the relation $\mathcal{R}$ as $\forall a, b \in \mathbb{Z}^{\mathbb{N}}$, $(a,b) \in \mathcal{R}$ iff $a \in \mathrm{Edit}(b)$. Clearly $\mathcal{R}$ is an equivalence relation.

13.2 DYNAMICAL PROPERTIES OF FULLY-ACA

The adaptation of CA dynamical properties to fully-ACA needs to take into account that there is an uncountable number of possible updating sequences. We will distinguish behaviours that can emerge for every sequence from the ones that can only appear for one particular sequence.

A sequence $(s_t)_{t\in\mathbb{N}}$ is ultimately periodic iff there exists a period $p \in \mathbb{N}_+$ and a preperiod $q \in \mathbb{N}$ such that $\forall i \in \mathbb{N}\, s_{p+q+i} = s_{q+i}$.

The dynamics of a fully-ACA is strictly related to the structure of the updating sequence. In particular the following property holds.

Proposition 13.2.1. *Let $C = (\Sigma, \lambda, r, \tau)$ be a fully-ACA. If a is ultimately periodic, then the orbit γ_x of every configuration $x \in \Sigma^{\mathbb{Z}}$ is ultimately periodic.*

Proof. Suppose that a is ultimately periodic with period p an preperiod q and let n be the number of the distinct values that appear in a. Fix a configuration $x \in \Sigma^{\mathbb{Z}}$. Let $B \subseteq \Sigma^{\mathbb{Z}} \times A^{\mathbb{N}_+}$ be the set of pair (c,b) where c is any configuration in γ_x and $b = a_{(m,+\infty)}$ for any $m \in \mathbb{N}$.

Since a is ultimately periodic and there are at most $|A|^n$ distinct configurations in γ_x, it holds that $|B| \leqslant (p+q)|A|^n$. Define $G : B \to B$ as follows: $\forall (c,b) \in B,\ G(c,b) = \left(F_{b_1}(c), b_{(1,+\infty)}\right)$. Thus, since B is finite there exist $p', q' \in \mathbb{N}$ such that $\forall (c,b) \in B,\ G^{p'+q'}(c,b) = G^{q'}(c,b)$. Since $\forall t \in \mathbb{N}$ the first component of $G^t(x,a)$ is $\gamma_x(t)$, the orbit γ_x is ultimately periodic. □

Remark. Note that the converse of Proposition 13.2.1 is not true. Indeed, let $C = (\Sigma, \lambda, r)$ be an uninstantiated fully-ACA where f is a constant function. Then, for any update sequence a such that $\exists k \in \mathbb{N}$ $\forall i \in \mathbb{N}, |a_i| \leqslant k$, all the orbits of the fully-ACA $C_a = (\Sigma, \lambda, r, \tau)$ are ultimately periodic.

Note that the property of having an ultimately periodic orbit is stable for a fully-ACA with an ultimately periodic sequence a. In fact, all the sequences obtained from a by a finite number of deletions, substitutions and insertions are still ultimately periodic.

The classical notion of sensitivity to initial conditions is adapted to both instantiated and uninstantiated fully-ACA.

Definition 13.2.1. An instantiated fully-ACA $C_\tau = (\Sigma, \lambda, r, \tau)$ is α-sensitive if its family $\mathcal{T}_{C_\tau}$ is sensitive. An uninstantiated fully-ACA $C = (\Sigma, \lambda, r)$ is α-sensitive if there exists a sequence $(\tau_t)_{t>0}$ such that the instantiated fully-ACA $C_\tau = (\Sigma, \lambda, r, \tau)$ is sensitive.

Remark. α-sensitivity means that at least one of the instantiated fully-ACA from the class C is α-sensitive to initial conditions. Requiring that all the instantiated fully-ACA are α-sensitive is a meaningless condition. Indeed, choose an integer $k > 0$ and consider the sequence $\tau = (k, k, k, \ldots)$. The orbits of two arbitrary configurations x and y such that $d(x,y) < 2^{-k}$ cannot separate by a distance greater than 2^{-k}.

Definition 13.2.2. An instantiated fully-ACA $C_\tau = (\Sigma, \lambda, r, \tau)$ is α-expansive if its family $\mathcal{T}_{C_\tau}$ is expansive. An uninstantiated fully-ACA $C = (\Sigma, \lambda, r)$ is α-expansive if there exists a sequence $a = (a_t)_{t>0}$ such that the instantiated fully-ACA $C_\tau = (\Sigma, \lambda, r, \tau)$ is α-expansive.

Like classical CA, α-expansivity implies α-sensitivity.

Proposition 13.2.2. *Let $C = (\Sigma, \lambda, r)$ be a fully-ACA with $r > 0$. If λ is either leftmost permutive or rightmost permutive then C is α-sensitive.*

Proof. Suppose that λ is rightmost-permutive. Set $\varepsilon = 2^{-r}$ and define the sequence τ as:

$$\tau = (\underbrace{0}, \underbrace{1,0}, \underbrace{2,1,0}, \underbrace{3,2,1,0}, \underbrace{4,3,2,1,0} \ldots) \tag{7}$$

Choose an arbitrary $x \in \Sigma^{\mathbb{Z}}$ and an integer $n \geqslant r$. Let $y \in \Sigma^{\mathbb{Z}}$ with $d(x,y) < 2^{-n}$ and $x_{n+1} \neq y_{n+1}$. There exists a first time $t_1 \in \mathbb{N}$ such

that $\tau_{t_1} + r = n + 1$. Since λ is rightmost permutive and $\gamma_x(t_1 - 1)_i = \gamma_y(t_1 - 1)_i$ for $i \in [-n, n]$, the smaller cell position in which $\gamma_x(t_1)$ and $\gamma_y(t_1)$ differ is $n - r + 1$. Repeat the previous argument k times with $k = \lfloor \frac{n+1}{r} \rfloor$. In this way, for any $1 \leqslant j \leqslant k$ there exists t_j such that $\gamma_x(t_j)$ and $\gamma_y(t_j)$ differ in position $n - jr + 1$ (this is possible since τ contains any positive integer infinitely many times). So, at a certain time t_k, the smallest cell position in which $\gamma_{t_k}(x)$ and $\gamma_{t_k}(y)$ differ will be smaller than r. In other words, there exists a time t_k such that $d(\gamma_x(t_k), \gamma_y(t_k)) < 2^{-r} = \varepsilon$, and so the fully-ACA $(\Sigma, \lambda, r, \tau)$ is α-sensitive.

If λ is leftmost permutive the proof is similar by considering the following sequence:

$$\tau' = (\underbrace{0}, \underbrace{-1, 0}, \underbrace{-2, -1, 0}, \underbrace{-3, -2, -1, 0}, \ldots) \tag{8}$$

□

Remark. Note that the sequence a defined in (7) is not the unique one which gives an α-sensitive instantiated fully-ACA. In fact, for every $n \in \mathbb{N}$ the sequence $\tau_{(n,+\infty)}$ can still be used in the proof. Moreover, for all $u \in \mathbb{Z}^{\mathbb{N}}_{-}$ all sequences obtained by interposing symbols of τ and u gives α-sensitivity.

Remark. As for classical CA, leftmost/rightmost permutivity is not a necessary condition for α-sensitivity. For example consider the alphabet $\Sigma = \{0, 1, 2\}$ and the function $\lambda : \Sigma^3 \to \Sigma$ defined as $\lambda(x_1, x_2, x_3) = 0$ if $x_3 = 0$, $\lambda(x_1, x_2, x_3) = 1$ otherwise. The fully-ACA $C = (\Sigma, \lambda, r, \tau)$ where λ is defined in (7) is sensitive but λ is neither leftmost nor rightmost permutive.

The property of being α-sensitive is stable for fully-ACA with a rightmost or leftmost permutive local rule and defined by the sequence λ from (7). Indeed, fix $\hat{\tau} \in \mathrm{Edit}(\tau)$. Since the number of insertions and deletions in τ is finite, there exist $k \in \mathbb{N}$ and $h \in \mathbb{Z}$ (depending on $\hat{\tau}$) such that $\tau_{i+h} = \tau_i$ for all $i \geqslant k$. For all $x \in \Sigma^{\mathbb{Z}}$ choose $y \in \Sigma^{\mathbb{Z}}$ with $x_{(p,+\infty)} \neq y_{(p,+\infty)}$ where $p = \max\{|\hat{\tau}_i| \mid i < k\}$. Then, by the same idea of the proof of Proposition 13.2.2, the difference between x and y is "pushed" in $[-r, r]$, and so the fully-ACA $(\Sigma, \lambda, r, \hat{\tau})$ is also α-sensitive.

Proposition 13.2.3. *Let $C = (\Sigma, \lambda, r)$ be a fully-ACA with $r > 0$. If λ is both leftmost permutive and rightmost permutive then C is α-expansive.*

Proof. We show that C is α-expansive with expansivity constant $\varepsilon = 2^{-r}$. Let

$$\tau'' = (0, 0, 1, 0, -1, 0, 2, 1, 0, -2, -1, 0, 3, 2, 1, 0, \ldots) \tag{9}$$

be the sequence obtained by interleaving the sequence τ and τ' defined in (7) and (8), respectively. We claim that the dynamics of the

fully-ACA $(\Sigma, \lambda, r, \tau'')$ "push" a difference between two arbitrary configurations into the window $[-r, r]$. Indeed, let $x, y \in \Sigma^{\mathbb{Z}}$ with $x \neq y$ and let $i \in \mathbb{Z}$ be a position such that $x_i \neq y_i$. Without loss of generality suppose that $i > r$. By a similar argument as in the proof of Proposition 13.2.2, the subsequence τ produces a dynamical evolution such that the difference is "pushed" in $[-r, r]$. Notice that the two subsequences τ and τ' operate distinctly on the positive and the negative positions, respectively. Hence the difference between two configurations cannot be cancelled if it is outside $[-r, r]$. □

The property of being α-expansive is stable for fully-ACA with a leftmost and rightmost local rule and defined by the sequence τ'' from (9).

Indeed, fix $\hat{\tau} \in \text{Edit}(\tau)$. Let $\varepsilon = 2^{-m}$ where $m = \max\{|\hat{\tau}_i| \mid i < \min\{k \in \mathbb{N} \mid \exists h \in \mathbb{Z} \ : \ \hat{\tau}_j = \tau_{j+h} \ \forall j \geqslant k\}\}$. Using the same idea of the proof of Proposition 13.2.3, for any two configurations $x, y \in \Sigma^{\mathbb{Z}}$ with $d(x, y) < 2^{-m} = \varepsilon$, it follows that there exist $t \in \mathbb{N}$ such that $d(\gamma_t(x), \gamma_t(y)) \geqslant \varepsilon$.

Remark that the α-expansivity constant ε depends on $\hat{\tau}$, and if we choose $\hat{\tau} \in \text{Edit}(\tau'')$, the fully-ACA $C_{\hat{\tau}} = (\Sigma, \lambda, r, \hat{\tau})$ may not be α-expansive with the same constant $\varepsilon = 2^{-r}$ as $C_{\tau''}$. Indeed, fix $k \in \mathbb{N}$ and consider $\hat{\tau} = (k, 0, 0, 1, 0, -1, \ldots)$. Let $C_{\hat{\tau}}$ be the fully-ACA defined by $\hat{\tau}$ and by $\lambda : \{0,1\}^3 \to \{0,1\}$ with $\lambda(c, d, e) = c \operatorname{xor} e$. Obviously λ is both leftmost and rightmost permutive. Consider $x, y \in \{0,1\}^{\mathbb{Z}}$ with $x_j = y_j \forall j \neq k$ and $x_k \neq y_k$. Then $\gamma_x(1) = \gamma_y(1)$. Hence $C_{\hat{\tau}}$ is not α-expansive with constant $\varepsilon \leqslant 2^{-k}$.

Proposition 13.2.4. *Let (Σ, λ, r) be a fully-ACA where f is both rightmost and leftmost permutive. Let τ'' be the sequence from (9). For any $\hat{\tau} \in \text{Edit}(\tau'')$ the fully-ACA $C_{\hat{\tau}} = (\Sigma, \lambda, r, \hat{\tau})$ is α-expansive with expansivity constant $\varepsilon = 2^{-r}$ iff λ is center permutive.*

Proof. Suppose that λ is not center permutive. Then, there exist $u, v \in \Sigma^{2r+1}$ such that $u_{r+1} \neq v_{r+1}$, $u_i = v_i \ \forall i \neq r+1$ and $\lambda(u) = \lambda(v)$. Choose $x, y \in \Sigma^{\mathbb{Z}}$ with $x_{[1,2r+1]} = v$ and $y_{[1,2r+1]} = u$. Let $C_{\hat{\tau}} = (\Sigma, \lambda, r, \hat{\tau})$ where $\hat{\tau} = (r+1, 0, 0, 1, 0, -1, \ldots) \in \text{Edit}(\tau'')$. Since $\gamma_x(1) = \gamma_y(1)$, $C_{\hat{\tau}}$ is not α-expansive with constant $\varepsilon = 2^{-r}$.

Conversely, assume now that f is center permutive. Fix $x, y \in \Sigma^{\mathbb{Z}}$ with $x \neq y$ and $\hat{\tau} \in \text{Edit}(\tau'')$. Then there exists $k \in \mathbb{N}$ and $h \in \mathbb{Z}$ such that $\tau_{i+h} = \hat{\tau}_i$ for all $i \geqslant k$. Since center permutivity is equivalent to α-surjectivity, it holds that $\gamma_x(k) \neq \gamma_y(k)$. Using the same technique of the proof of Proposition 13.2.2 with $\gamma_x(k)$ and $\gamma_y(k)$ as initial configurations and $\hat{\tau}_{[k,+\infty)}$ as the updating sequence, it follows that $C_{\hat{\tau}}$ is α-expansive with expansivity constant $\varepsilon = 2^{-r}$. □

Another interesting dynamical property of CA is transitivity. As with sensitivity and expansivity, the notion of transitivity can be adapted to fully-ACA.

Definition 13.2.3. An instantiated fully-ACA $C_\tau = (\Sigma, \lambda, r, \tau)$ is α-transitive if its family $\mathcal{T}_{C_\tau}$ is transitive. An uninstantiated fully-ACA $C = (\Sigma, \lambda, r)$ is α-transitive if there exists a sequence $\tau = (\tau_t)_{t>0}$ such that the instantiated fully-ACA $C_\tau = (\Sigma, \lambda, r, \tau)$ is α-transitive.

Lemma 13.2.5. *Let $C = (\Sigma, \lambda, r)$ be a fully-ACA with $r > 0$ and let λ be rightmost permutive. Consider the predicate $P(k)$ defined as follows:*

$$P(k) = \left(\begin{array}{c} \forall x \in \Sigma^{\mathbb{Z}},\ \forall b \in \Sigma,\ \forall m \in \mathbb{Z}\ \ \exists x' \in \Sigma^{\mathbb{Z}} \\ \text{s.t. } x_i = x'_i\ \forall i \neq m + rk \text{ and} \\ \left(F_m \circ F_{m+r} \circ \ldots \circ F_{m+r(k-2)} \circ F_{m+r(k-1)}\right)(x')_m = b \end{array} \right)$$

Then, $\forall k \in \mathbb{N}_+$, $P(k)$ is true.

Proof. We proceed by induction on k. When $k = 1$ fix $x \in \Sigma^{\mathbb{Z}}$, $b \in \Sigma$ and $m \in \mathbb{Z}$. Let $x' \in \Sigma^{\mathbb{Z}}$ with $x_i = x'_i\ \forall i \neq m + r$ and $x'_{m+r} = a$ where, by permutivity of λ, a is such that $\lambda(x_{[m-r,m+r)}a) = b$. It immediately follows that $F_m(x')_m = b$ and so $P(1)$ is true.

Suppose now that $P(k)$ is true for a certain $k \in \mathbb{N}_+$. Choose $x \in \Sigma^{\mathbb{Z}}$, $b \in \Sigma$ and $m \in \mathbb{Z}$. By induction hypothesis there exists $y \in \Sigma^{\mathbb{Z}}$ with $x_i = y_i\ \forall i \neq m + rk$ such that $\left(F_m \circ \ldots \circ F_{m+r(k-1)}\right)(y)_m = b$. Let $x' \in \Sigma^{\mathbb{Z}}$ be with $x'_i = x_i\ \forall i \neq m + r(k+1)$ and $x'_{m+r(k+1)} = a$ where, by permutivity of λ, a is the unique symbol such that $\lambda(x_{[m-r(k-1),m+r(k+1))}a) = y_{m+rk}$. Then,

$$\left(F_m \circ \ldots \circ F_{m+r(k-1)} \circ F_{m+rk}\right)(x')_m =$$
$$\left(F_m \circ \ldots \circ F_{m+r(k-1)}\right)(y_{(-\infty,m+r(k+1))}a y_{(m+r(k+1),+\infty)})_m = b$$

This proves that $P(k+1)$ is true. □

The previous lemma states that for any configuration x and for any position m there exists a configuration x' that differs from x in at most one position (arbitrarily far from m) and an updating sequence such that after a certain number t of time steps $\gamma_x(t)_m \neq \gamma_{x'}(t)_m$. We now generalize this result to more than one position.

Lemma 13.2.6. *Let $C = (\Sigma, \lambda, r)$ be a fully-ACA with $r > 0$ and let λ be rightmost permutive. Consider the predicate $Q(\ell)$ defined as follows:*

$$Q(\ell) = \left(\begin{array}{c} \forall x \in \Sigma^{\mathbb{Z}},\ \ \forall k \in \mathbb{N}_+,\ \ \forall w \in \lambda^\ell,\ \ \forall m \in \mathbb{Z} \\ \exists x' \in \Sigma^{\mathbb{Z}} \text{ s.t. } x_i = x'_i\ \ \forall i \notin [m + rk, m + rk + \ell) \text{ and} \\ \left(G_{\ell-1} \circ \ldots \circ G_0\right)(x')_{[m,m+\ell)} = w \end{array} \right)$$

where for all $j \in [0, \ell)$,

$$G_j = F_{m+j} \circ \ldots \circ F_{m+j+r(k-1)}$$

Then, $\forall \ell \in \mathbb{N}_+$, $Q(\ell)$ is true.

Proof. We proceed by induction on ℓ. The predicate $Q(1)$ is true by Lemma 13.2.5.

Assume now that $Q(\ell)$ is true for a certain $\ell \in \mathbb{N}_+$. Fix $x \in \Sigma^{\mathbb{Z}}$, $k \in \mathbb{N}_+$, $m \in \mathbb{Z}$ and $w \in \Sigma^{\ell+1}$. Then w can be written as $w = vb$ with $v \in \Sigma^{\ell}$ and $b \in \Sigma$. By induction hypothesis there exists $y \in \Sigma^{\mathbb{Z}}$ with $y_i = x_i \ \forall i \notin [m + rk, m + rk + \ell)$ such that:

$$(G_{\ell-1} \circ \ldots \circ G_0)(y)_{[m,m+\ell)} = v.$$

Let $z = (G_{\ell-1} \circ \ldots \circ G_0)(y)$. By Lemma 13.2.5 there exists z' with $z'_i = z_i \ \forall i \neq m + rk + \ell$ and

$$\left(F_{m+\ell} \circ \ldots \circ F_{m+\ell+r(k-1)}\right)(z')_{m+\ell} = b,$$

i.e., $G_\ell(z')_{m+\ell} = b$. Let $x' \in \Sigma^{\mathbb{Z}}$ be such that $x'_i = y_i \ \forall i \neq m + \ell + rk$ and $x'_{m+\ell+rk} = z'_{m+\ell+rk}$. Then:

$$(G_\ell \circ G_{\ell-1} \circ \ldots \circ G_0)(x') = G_\ell(z')$$

Since G_ℓ leaves $z'_{[m,m+\ell)}$ unchanged, we have that:

$$(G_\ell \circ G_{\ell-1} \circ \ldots \circ G_0)(x')_{[m,m+\ell+1)} = G_\ell(z')_{[m,m+\ell+1)} = vb = w.$$

This proves that $Q(\ell + 1)$ is true. □

Similar results hold for fully-ACA with a leftmost permutive local rule.

For any pair of configurations x, y, the previous Lemma assures the existence of a configurations x' in the neighbourhood of x and an update sequence individuating an orbit of the configuration x' which intersects a neighbourhood of y. This is not enough to conclude that rightmost permutive fully-ACA are α-transitive since the sequence of updates depends on the choice of the neighbourhoods of x and y.

In order to proceed, for a fixed $r > 0$, we will denote by $S(h, k)$ with $k, h \in \mathbb{N}$ the finite sequence:

$$\begin{gathered}
-h + r(k-1), -h + r(k-2), \ldots, \\
-h + r, -h, -h + 1 + r(k-1), \ldots, \\
r(k-1), r(k-2), \ldots, 0, 1 + r(k-1), \ldots, \\
h + r(k-1), h + r(k-2), \ldots, h + r, h.
\end{gathered}$$

Roughly speaking, with $m = -h$ and $\ell = 2h + 1$, $S(h, k)$ is the (finite) update sequence individuating the orbit $\gamma_{x'}$ from Lemma 13.2.6 which at time $k(2h + 1)$ has the desired word w between positions $-h$ and h. Define now $\mathrm{succ}(S(h, k))$ as the sequence $S(h', k')$ where $h' = h + rk$ and $k' = \min\{m \in \mathbb{N} \mid -h' + rm > h + rk\}$. Consider now the sequence:

$$\tau = v_0 v_1 \ldots \text{ where } v_0 = S(0, 1) \text{ and } \forall i \in \mathbb{N}_+, \ v_i = \mathrm{succ}(v_{i-1}) \quad (10)$$

Example 13.2.1. If $r = 1$ the sequence $\tau = \nu_0\nu_1\dots$ defined in (10) is the following:

$$\tau = (\underbrace{0}_{\nu_0 = S(0,1)}, \underbrace{1,0,-1,2,1,0,3,2,1}_{\nu_1 = S(1,3)}, \dots)$$

Furthermore, if $r = 2$ the sequence is:

$$\tau = (\underbrace{0}_{\nu_0 = S(0,1)}, \underbrace{2,0,-2,3,1,-1,4,2,0,5,3,1,6,4,2}_{\nu_1 = S(2,3)}, \dots)$$

Proposition 13.2.7. *Let $C = (\Sigma, \lambda, r)$ be a fully-ACA with $r > 0$. If λ is permutive then C is α-transitive.*

Proof. Suppose that λ is rightmost permutive (the proof for a leftmost permutive rule is similar). We claim that the sequence τ defined in (10) is such that the instantiated fully-ACA $C_\tau = (\Sigma, \lambda, r, \tau)$ is α-transitive.Choose $\varepsilon > 0$ and $x, y \in \Sigma^{\mathbb{Z}}$. Let $i \in \mathbb{N}$ be such that $\nu_i = S(h,k)$ for some $h, k \in \mathbb{N}$ with $2^{-h} < \varepsilon$. Let $z = \gamma_x(|\nu_0 \dots \nu_{i-1}|)$.

By Lemma 13.2.6 there exists z' with $z'_j = z_j$ $\forall j \notin [-h + rk, -h + rk + 2h + 1)$ such that $\gamma_{z'}(|\nu_i|)_{[-h,h]} = y_{[-h,h]}$.

Note that $x_{[-h+rk,-h+rk+2h+1)}$ is equal to $z_{[-h+rk,-h+rk+2h+1)}$. Define now x' as $x'_j = x_j$ $\forall j \notin [-h + rk, -h + rk + 2h + 1)$ and $x'_j = z'_j$ $\forall j \in [-h + rk, -h + rk + 2h + 1)$. We have that $d(x, x') \leqslant 2^{-h} < \varepsilon$ and $\gamma_{x'}(|\nu_0 \dots \nu_i|)_{[-h,h]} = \gamma_{z'}(|\nu_i|)_{[-h,h]} = y_{[-h,h]}$. Therefore C_τ is α-transitive. □

Proposition 13.2.8. *Let $C_\tau = (\Sigma, \lambda, r, \tau)$ be an instantiated fully-ACA with a defined in (10). If λ is either rightmost permutive or leftmost permutive then α-transitivity is stable for C_τ.*

Proof. Let $\hat{\tau} \in \mathrm{Edit}(\tau)$. Since the number of differences between τ and $s\hat{u}s$ is finite, there exists $i \in \mathbb{N}$ such that $s\hat{u}s = c\nu_i\nu_{i+1}\dots$ where $c = c_1, \dots, c_q$ and $\nu_i = S(h,k)$ for some $h, k \in \mathbb{N}$ with $\forall j \in [1, q]$ $c_j < -h + rk$. Fix $x, y \in \Sigma^{\mathbb{Z}}$. Let $z = \left(F_{c_q} \circ \dots \circ F_{c_1}\right)(x)$. Applying the same idea of the proof of Proposition 13.2.7 to z and y using $\nu_i\nu_{i+1}\dots$ as the updating sequence, we obtain that $C_{s\hat{u}s} = (\Sigma, \lambda, r, s\hat{u}s)$ is α-transitive. □

Remark. All the sequences that are needed to obtain α-sensitivity and α-transitivity have to be unbounded. Indeed for α-sensitivity, we need to consider differences that can be arbitrarily far from the center. For α-transitivity, the reason is similar: the update sequence has to contain arbitrarily far positions in order to define orbits reaching arbitrarily small neighbourhoods.

Another important notion regarding the dynamics of a CA is DPO. Before defining α-DPO, we need the notion of periodic point for a fully-ACA.

Definition 13.2.4. Let $C_\tau = (\Sigma, \lambda, r, \tau)$ be an instantiated fully-ACA. A point $x \in \Sigma^{\mathbb{Z}}$ is called periodic if there exists $p \in \mathbb{N}_+$ such that for all $n \in \mathbb{N}$, $\gamma_x(n) = \gamma_x(n+p)$.

Definition 13.2.5. Let $C_\tau = (\Sigma, \lambda, r, \tau)$ be an instantiated fully-ACA. C_τ has α-DPO if its set of periodic points is dense in $\Sigma^{\mathbb{Z}}$. An uninstantiated fully-ACA $C = (\Sigma, \lambda, r)$ has α-DPO if there exists a sequence $\tau = (\tau_t)_{t>0}$ such that the instantiated fully-ACA $C_\tau = (\Sigma, \lambda, r, \tau)$ has α-DPO.

Proposition 13.2.9. *Let $C = (\Sigma, \lambda, r)$ be a fully-ACA. If C is α-surjective then it has α-DPO.*

Proof. Consider the sequence $\tau = (k, k, k, \ldots)$ for any $k \in \mathbb{N}$. Fix $x \in \Sigma^{\mathbb{Z}}$ and consider the possible values that the orbit $\gamma_x(t)_k$ of the fully-ACA C_τ can assume. For every $b = F_k(x)_k$ there is exactly one $b' \in \Sigma$ such that $F_k(x_{(-\infty,k-1]} b' x_{[k+1,+\infty)})$ is equal to b. Because only the cell in position k changes, neither an aperiodic orbit (indeed the number of elements in the orbit of x is finite) nor an ultimately periodic orbit exist (indeed every configuration has exactly one preimage). This means that x is a periodic point for C_τ, hence the fully-ACA has α-DPO. □

Remark that α-DPO is not a stable property. Indeed, let $C_\tau = (\{0,1\}, 0, \lambda, \tau)$ with $\lambda(c) = 1 - c \ \forall c \in \{0,1\}$ and $\tau = (0, 0, \ldots)$. Let $s\hat{u}s = (1, 0, 0, \ldots) \in \mathrm{Edit}(\tau)$. Clearly, there are no configurations that are periodic points for $C_{s\hat{u}s}$.

Remark. We remark that α-DPO is a property meaningful also for uninstantiated fully-ACA (i.e.,, there exists an uninstantiated fully-ACA that does not have α-DPO). In fact, let $\lambda : \{0,1\} \to \{0,1\}$ be defined as $\lambda(c) = 0 \ \forall c \in \{0,1\}$ and fix $\tau = (\tau_1, \tau_2, \ldots)$. Choose $x \in \{0,1\}^{\mathbb{Z}}$ such that $x_{\tau_1} = 1$. It is immediate that the orbit of x is not periodic, also, any $y \in \{0,1\}^{\mathbb{Z}}$ such that $d(x,y) < 2^{|\tau_1|}$ cannot have a periodic orbit. This means that $C = (\{0,1\}, \lambda, 1)$ does not have α-DPO.

Remark. There exists an α-injective uninstantiated fully-ACA with a leftmost and center permutive local rule, i.e., a fully-ACA that is α-surjective, α-sensitive, α-transitive and has α-DPO. Let $\Sigma = \{0,1\}$ and $\lambda : \{0,1\}^3 \to \{0,1\}$ defined as $\lambda(a,b,c) = a \,\mathrm{xor}\, b$. The local rule is both leftmost permutive and center permutive. Then, by Proposition 13.1.1, 13.2.2, 13.2.7 and 13.2.9 the fully-ACA $(\{0,1\}, \lambda, 1)$ is α-surjective, α-sensitive, α-transitive and has α-DPO.

An other important property of classical CA is equicontinuity.

Definition 13.2.6. An instantiated fully-ACA $C_\tau = (\Sigma, \lambda, r, \tau)$ is α-equicontinuous if its family $\mathcal{T}_{C_\tau}$ is equicontinuous. An uninstantiated fully-ACA $C = (\Sigma, \lambda, r)$ is α-equicontinuous if for every sequence $\tau = (\tau_t)_{t>0}$ the instantiated fully-ACA $C_\tau = (\Sigma, \lambda, r, \tau)$ is α-equicontinuous.

Trivially, any α-equicontinuous fully-ACA is not α-sensitive. All constant local rules of any radius as well as all local rules of radius 0 define α-equicontinuous fully-ACA.

It is useful to remark that, unlike other properties as α-sensitivity, α-equicontinuity requires that every sequence τ defines a fully-ACA C_τ with an equicontinuous family $\mathcal{T}_{C_\tau}$. Indeed, for any uninstantiated fully-ACA C there always exists a sequence $\tau = (0, 0, 0, \ldots)$ such that the family $\mathcal{T}_{C_\tau}$ is equicontinuous. In particular, for any $\varepsilon > 0$, the equicontinuity condition is satisfied with $\delta = 2^{-r}$.

Recall that for classical CA equicontinuity is equivalent to ultimate periodicity [119] (i.e., all the configurations are ultimately periodic). On the contrary, the equivalence between α-equicontinuity and fully-ACA-ultimate periodicity is not true. Indeed, consider the fully-ACA $C = (\{0, 1\}, 0, \lambda)$ with λ defined as $\forall c \in \{0, 1\}$, $\lambda(c) = 0$. Clearly, C is α-equicontinuous. However, the orbit of $x = (\ldots, 1, 1, 1, \ldots)$ w.r.t. the updating sequence $\tau = (0, 1, 2, \ldots)$ is not ultimately periodic.

13.3 FURTHER REMARKS

In this chapter fully asynchronous cellular automata have been studied. The classical notions of surjectivity and injectivity have been adapted to the new setting where there is not a single global map. We have proved that their fully-ACA counterparts are equivalent. The study of the dynamics of asynchronous cellular automata has been performed with the adapted notions of transitivity, sensitivity, expansivity, DPO and equicontinuity. We found that leftmost/rightmost permutivity of the local rule is strictly linked to many of these properties. In fact, leftmost/rightmost permutivity implies both α-transitivity and α-sensitivity.

14

COMPUTATIONAL POWER OF FULLY ASYNCHRONOUS CA

In this chapter we focus on the way in which an ACA simulates a Turing Machine. We provides three different constructions, depending on the way the TM is simulated.

14.1 SIMULATION OF TURING MACHINES

It is well-known that CA are a universal computational model according to different notions of universality (for a survey see [154]). The main point to prove Turing universality is to simulate a TM. Of course, one can apply similar ideas and constructions to prove computational universality or computational capability of ACA (see for example [145, 148, 123]). In this section we would like to precise the computational cost of such simulations. While Turing universality of fully-ACA is expected, it still provides an indication of the possibility of using many natural systems that are "natural ACA" as a way to perform computation.

The basic idea when simulating a TM using a fully-ACA is to act by "extracting" first the information about the state of the TM from the current configuration of the fully-ACA, and then to operate the TM transition saving information on the fully-ACA configuration again. The way of saving the TM state and the way we extract it from the current configuration lead to the two following notions of simulation.

NOTATION. Given a collection of finite and pairwise disjoint sets $A_1, A_2, \ldots, A_n$, for each $i \in [1, n]$ define the projection map $\Pi_{A_i} : A_1 \times A_2 \times \ldots \times A_n \to A_i$ as

$$\forall (a_1, a_2, \ldots, a_n) \in A_1 \times A_2 \times \ldots \times A_n$$
$$\Pi_{A_i}(a_1, a_2, \ldots, a_i \ldots, a_n) = a_i$$

These projection maps can be naturally extended to work with configurations. Indeed, for any $i \in [1, n]$ and any configuration $c \in (A_1 \times \ldots \times A_n)^{\mathbb{Z}}$ with a little abuse of notation we denote by $\Pi_{A_i}(c)$ the configuration defined as $\Pi_{A_i}(c)_j = \Pi_{A_i}(c_j)$, for all $j \in \mathbb{Z}$.

Given a configuration $c \in \Sigma^{\mathbb{Z}}$ and a function $\psi : \mathbb{Z} \to \mathbb{Z}$, c^{ψ} is the configuration defined as $c^{\psi}_i = c_{\psi(i)}$ for all $i \in \mathbb{Z}$.

Definition 14.1.1. Let $\mathcal{M} = (Q, \Sigma, \Gamma, b, \delta, q_0, F)$ be a TM and $C = (A, \lambda, r, \tau)$ be a fully-ACA. C *strictly simulates* $\mathcal{M}$ iff $A = \Gamma \times B$ for some finite set B and there exists a strictly increasing function $\zeta : \mathbb{N} \to \mathbb{N}$ such that for any input $x \in \Sigma^*$ of $\mathcal{M}$, there exists a configuration $c \in \Sigma^{\mathbb{Z}}$ satisfying the conditions:

1. $\Pi_\Gamma(c) = {}^\omega bxb^\omega$ and $\Pi_B(c) = {}^\omega sus^\omega$ for some $u, s \in B$;

2. for any time $t \in \mathbb{N}$,

$$\Pi_\Gamma(f^{(\zeta(t))}(c)) = T_t$$

In other words, a fully-ACA C strictly simulates a TM $\mathcal{M}$ if its configurations can represent in a direct way the tape of $\mathcal{M}$, possibly using an additional amount of information (stored in the alphabet B) and some additional time. Relaxing the condition on the representation of the tape, the following weaker notion of simulation is obtained.

Definition 14.1.2. Let $M = (Q, \Sigma, \Gamma, b, \delta, q_0, F)$ be a TM. Let $C = (A, \lambda, r, \tau)$ be a fully-ACA. C *scattered strictly simulates* $\mathcal{M}$ iff $A = \Gamma \times B$ for some finite set B, and there exists a strictly increasing function $\zeta : \mathbb{N} \to \mathbb{N}$ such that for any input $x \in \Sigma^*$ of $\mathcal{M}$ there exist a strictly increasing function $\psi : \mathbb{Z} \mapsto \mathbb{Z}$ and a configuration $c \in \Sigma^{\mathbb{Z}}$ satisfying the following conditions:

1. a) $\Pi_B(c^\psi) = {}^\omega sus^\omega$, for some $u, s \in B$;
   b) $\forall i \in \mathbb{Z} \setminus \psi(\mathbb{Z}), \Pi_B(c_i) = q$, for some $q \in B$;
   c) $\Pi_\Gamma(c^\psi) = {}^\omega bxb^\omega$;

2. for any time $t \in \mathbb{N}$,

$$\Pi_\Gamma((f^{(\zeta(t))}(c))^\psi) = T_t$$

A scattered strict simulation assumes that only a subset of cells participates to the simulation and the others are somehow inactive. For this reason, in the fully-ACA configurations there can be an offset made for example by bs between the symbols of the TM tape content. Note that when the function ψ is $\psi(i) = i$, then scattered strict simulation and strict simulation coincide.

According to the above definitions, even if a fully-ACA can simulate a TM on a fixed input x, it might not be able to simulate the same TM on a different input simply because of an inappropriate updating sequence τ.

14.1.1 *Construction 1.*

Given a TM $\mathcal{M} = (Q, \Sigma, \Gamma, b, \delta, q_0, F)$ build a family of fully-ACA $C_\tau = (A, \lambda, 1, \tau)$ such that $A = \Gamma \times Q \times D \times C$, where $D = \{L, R\}$, $C = \{\textit{ready}, \textit{active}, \textit{inactive}\}$, and the local rule $\lambda : A^3 \to A$ is defined as follows

$$\lambda(u, v, z) = \begin{cases} (\sigma, q, m, \textit{ready}) & \text{if } u = (\sigma_u, q_u, R, \textit{active}),\ v = (\sigma_v, q_v, m_v, \textit{inactive}) \text{ and } \delta(q_u, \sigma_v) = (q, \sigma, m) \\ (\sigma_v, q_v, R, \textit{inactive}) & \text{if } v = (\sigma_v, q_v, R, \textit{active}),\ z = (\sigma_z, q_z, m_z, \textit{ready}) \\ (\sigma_v, q_v, m_v, \textit{active}) & \text{if } u = (\sigma_u, q_u, R, \textit{inactive}),\ v = (\sigma_v, q_v, m_v, \textit{ready}) \\ (\sigma, q, m, \textit{ready}) & \text{if } z = (\sigma_z, q_z, L, \textit{active}),\ v = (\sigma_v, q_v, m_v, \textit{inactive}) \text{ and } \delta(q_z, \sigma_v) = (q, \sigma, m) \\ (\sigma_v, q_v, L, \textit{inactive}) & \text{if } v = (\sigma_v, q_v, L, \textit{active}),\ u = (\sigma_u, q_u, m_u, \textit{ready}) \\ (\sigma_v, q_v, m_v, \textit{active}) & \text{if } z = (\sigma_z, q_z, L, \textit{inactive}),\ v = (\sigma_v, q_v, m_v, \textit{ready}) \\ v & \text{otherwise.} \end{cases}$$

Every cell of the fully-ACA contains the symbol of the corresponding cell on the TM tape, the state of the TM, the direction of movement of the TM head, and a value $\xi \in C$ to control the simulation. At TM time t, the cell i with $\xi = \textit{active}$ is the one where the TM head is positioned at time $t-1$, i.e., $p_{t-1} = i$ (with $p_{-1} = -1$). During the fully-ACA evolution, at most one cell in whole configuration has $\xi = \textit{active}$. If the updating sequence allows the cell $i+1$ (resp., $i-1$) to be updated and the cell i has $m = R$ (resp., $m = L$), then the cell $i+1$ (resp., $i-1$) changes its state according to the TM rule and its

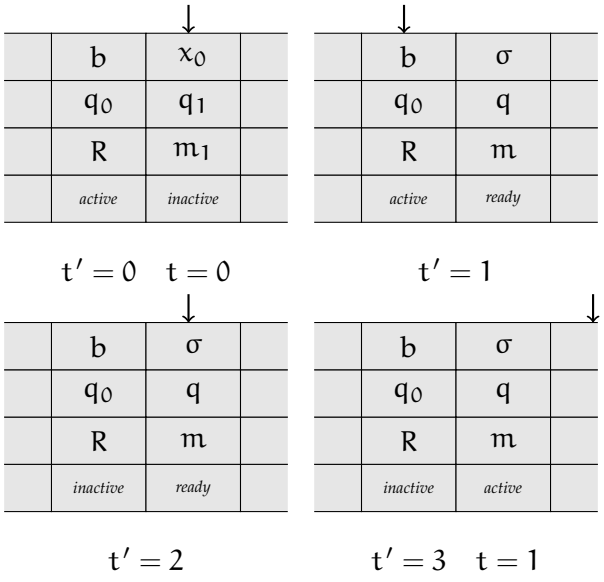


Figure 33: Simulation of the first step of a TM using a fully-ACA built by construction 1 with updating sequence $\tau = (0, -1, 0, 1, \ldots)$. The fully-ACA and TM times are denoted by $t' = \zeta(t)$ and t, respectively. The arrow points at the current active cell of the fully-ACA. The transition function of the simulated TM is such that $\delta(q_0, x_0) = (q, \sigma, m)$.

own value of ξ is set to *ready* to indicate that the information about the head position has to be moved to this cell. To perform it, at subsequent times fully-ACA will set the cell with $\xi =$ *active* to *inactive* and the cell with $\xi =$ *ready* to *active*. An example of this behavior is shown in Figure 33.

In order to (strictly) simulate a TM on input $x = x_0 \ldots x_{n-1} \in \Sigma^*$, fully-ACA given by the above construction have to start on the following configuration c

$$\forall i \in \mathbb{Z},\ c_i = \begin{cases} (x_i, q, m, \textit{inactive}) & \text{if } 0 \leqslant i < |x|. \\ (b, q_0, R, \textit{active}) & \text{if } i = -1. \\ (b, q, m, \textit{inactive}) & \text{otherwise,} \end{cases}$$

where q and m are an arbitrarily chosen state and movement (since they will not be used in the simulation, their choice can be arbitrary). The last point to precise is which updating sequences can be used. Of course, that depends on the TM to simulate but there are sequences that can be used in "all occasions", they are called universal.

Roughly speaking, a sequence is universal on the set $K \subseteq \mathbb{Z}$ if any cell of K is updated infinitely many times. When no misunderstanding is possible, we will simply refer to a universal sequence as a sequence which is universal on the whole $\mathbb{Z}$.

Definition 14.1.3. An updating sequence τ is *universal* on the set $K \subseteq \mathbb{Z}$ iff it holds that $|\{i \in \mathbb{N}, \tau_i = k\}| = \infty$ for every $k \in K$.

Lemma 14.1.1. *Consider a fully-ACA* $C = (A, \lambda, 1, \tau)$ *given by Construction* 1 *and let* τ *be universal. Then,* C *is Turing universal.*

Proof. Consider a TM $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, b, q_0, F)$ and a fully-ACA $C = (A, f, 1, \tau)$ with τ universal. For any $x \in \Sigma^*$ input of $\mathcal{M}$, let c be the initial configuration built in construction 1. Let us prove that C strictly simulates $\mathcal{M}$.

Let $R_t = (T_t, q_t, p_t)$ be the configuration of the $\mathcal{M}$ at time t. We claim that for all $t \in \mathbb{N}$ there exists $t' \in \mathbb{N}$ such that the configuration $c' = f^{(t')}(c)$ of C has the following properties

1. $\Pi_\Gamma(c') = T_t$.
2. $\Pi_Q(c'_{p_{t-1}}) = q_t \quad (p_{-1} = -1)$.
3. $\Pi_C(c'_{p_t}) =$ *active* and $\Pi_C(c'_i) =$ *inactive*, for all $i \in \mathbb{Z} \setminus \{p_t\}$.
4. $\Pi_D(c'_{p_t}) = R$ if $p_t > p_{t-1}$; L if $p_t < p_{t-1}$.

We proceed by induction. For $t = 0$, the claim is true by construction ($t' = 0$ and $c' = c$).

Assume now that the claim is true for $t \geqslant 0$, i.e.,, there exists t' such that the configuration $c' = f^{(t')}(c)$ satisfies the four stated properties. Remark that $\Pi_C(c'_{p_t}) =$ *active* and hence the only cells that can change their value are at positions $p_t + 1$ or $p_t - 1$, depending on the value of $\Pi_D(c'_{p_t})$. Suppose that $\Pi_D(c'_{p_t}) = R$ (the other case is similar).

Marking of the ready cell. Since τ is universal, there exists $t'' > t'$ such that $\tau_{t''} = p_t + 1$ and for any other $\bar{t} \in \mathbb{N}$ either $\tau_{\bar{t}} \neq p_t + 1$ or $\bar{t} > t''$. According to the definition of f, at time t'' the cell $p_t + 1$ will become $(\sigma, q, m, \textit{ready})$, where $(\sigma, q, m) = \delta(\Pi_Q(c''_{p_t}), \Pi_\Gamma(c''_{p_t+1}))$ and $c'' = f^{(t'')}(c)$. Moreover, no other cell can change its content between time $t' + 1$ and $t'' - 1$. Therefore $\Pi_\Gamma(c'') = T_{t+1}$ and $\Pi_Q(c''_{p_t+1}) = q_{t+1}$, i.e.,, c'' satisfies the first and second properties.

Deactivation of the active cell. Again, since τ is universal, there exists $t''' > t''$ such that $\tau_{t'''} = p_t$ and for any other $\bar{t} \in \mathbb{N}$ either $\tau_{\bar{t}} \neq p_t + 1$ or $\bar{t} > t'''$. According to f, the forth component of the cell at position $\tau_{t'''}$ is set to *inactive*. Once more, remark that no changes in the configuration of the fully-ACA occur between time $t'' + 1$ and $t''' - 1$.

Activation of the ready cell. Finally, by the universality of τ, there exists $\tilde{t} > t'''$ such that $\tau_{\tilde{t}} = p_t + 1$ and for any other $\bar{t} \in \mathbb{N}$ either $\tau_{\bar{t}} \neq p_t + 1$ or $\bar{t} > \tilde{t}$. From the definition of f, one deduces that the only possibility is that the fourth component of the cell at position $p_t + 1$ in $f^{(\tilde{t})}(c)$ is set to *active*. Since no changes in the configuration of the fully-ACA occur between time $t''' + 1$ and $\tilde{t} - 1$, the claim is proved. □

Lemma 14.1.2. *Let* $C = (A, \lambda, 1, \tau)$ *be the fully-ACA given by Construction* 1*. If* C *is Turing universal, then* τ *is universal.*

Proof. Consider the TM $\mathcal{M} = (\{q_R, q_L\}, \{0,1\}, \{0,1,b\}, b, \delta, q_R, \emptyset)$ in which the function δ is defined as follows

(q,σ)	(q_R, b)	$(q_R, 0)$	$(q_R, 1)$
$\delta(q,\sigma)$	$(q_L, 1, L)$	$(q_R, 1, R)$	$(q_R, 1, R)$

(q,σ)	(q_L, b)	$(q_L, 0)$	$(q_L, 1)$
$\delta(q,\sigma)$	$(q_R, 0, R)$	$(q_L, 0, L)$	$(q_L, 0, L)$

On any input, $\mathcal{M}$ writes a symbol 1 on the cell 0, then it writes 1s towards the right until a blank symbol is reached. When a blank is reached, it moves left writing a symbol 0 until a blank is encountered at this point it starts moving right writing 1s and so forth. It is clear that the head of $\mathcal{M}$ passes through any cell of the tape infinitely many times. Therefore any fully-ACA given by Construction 1 needs an universal updating sequence to strictly simulate it. □

According to Lemma 14.1.1 and Lemma 14.1.2 we have the following.

Theorem 14.1.3. *A fully-ACA* $C = (A, \lambda, 1, \tau)$ *given by Construction* 1 *is Turing universal if and only if* τ *is universal.*

The previous result proves that the class of fully-ACA given by Construction 1 are computational universal but it seems that requiring an universal updating sequence involves a considerable time loss (see the proof of the Theorem 14.1.3). The following proposition shows that there exist (carefully chosen) updating sequences such that the time loss is acceptable (quadratic).

Proposition 14.1.4. *Given a TM* $\mathcal{M}$ *that executes in time* $T(n)$*, there exists a fully-ACA* C *given by Construction* 1 *that simulates* $\mathcal{M}$ *in time* $O\left(T(n)^2\right)$.

Proof. For any $i \in \mathbb{N}$, let s_i be the finite sequence $s_i = (-i, -i+2, \ldots, i-1, i)$. Define the sequence τ by the concatenation of the s_i sequences as follows:

$$\begin{aligned} \tau &= \underbrace{s_0(-1)s_0}\ \underbrace{s_1 s_0 s_1}\ \underbrace{s_2 s_1 s_2}\ldots \\ &= \underbrace{0,-1,0,}\underbrace{-1,1,0,-1,1,}-2,0,2,-1,1,\ldots \end{aligned}$$

Clearly, τ is universal. Consider now the fully-ACA $C = (A, \lambda, 1, \tau)$ where λ and A are as in Construction 1. It is easy to verify that the every segment of the fully-ACA evolution individuated by the block $s_i s_{i-1} s_i$ simulates one step of $\mathcal{M}$. The size of the block increases by a (multiplicative) constant 3 for every i. The total length of the simulation is then bounded by $1 + |s_0| + \sum_{i=1}^{T(n)} 2 \cdot |s_i| + |s_{i-1}|$ $= 2 + \sum_{i=1}^{T(n)} 3i + 2 = O\left(T(n)^2\right)$. □

Remark. The time $O\left(T(n)^2\right)$ is the best asymptotic limit for a sequence if we do no restrict the kind of TM that can be simulated. Indeed, consider a TM that reads an input of length n that is the description of the next n head movements (i.e.,, it stops in $T(n) = 2n + 1$ steps). Since the updating sequence is the same for all the inputs, it needs to account for all the possible head movements. The position of the head at time $n + 1$ is n, either $n + 1$ or $n - 1$, at time $n + 2$, and so on. Let us introduce the graph $G = (V, E)$ where $V = \{n + 1, \ldots, 2n + 1\} \times \{0, \ldots, 2n\}$ and $E = \{((t, a), (t + 1, b)) \mid t \in \{n + 1, \ldots, 2n\}$ and $|a - b| = 1\}$. Any path starting from $(n + 1, n)$ and ending to $(2n + 1, b)$ with $b \in \{0, \ldots, 2n\}$ represents a possible sequence of a TM head. In order to simulate all the head movements, all the nodes of the graph must be visited at least once by these paths. Since the graph has $O(T(n)^2)$ vertexes, the simulation time must be $O(T(n)^2)$.

Remark. Actually, for any a TM $\mathcal{M}$ executing in $O(T(n))$, there are uncountably many fully-ACA given by Construction 1 that simulate $\mathcal{M}$ in time $O(T(n)^2)$. An infinite set of them is individuated by the sequences obtained from the one illustrated in the Prop. 1 "inserting" in it any integer in one or more positions.

The slowdown in the simulation of TM using fully-ACA given by Construction 1 is essentially given by the fact that we want a strict simulation and we must keep track (among other things) of the position of the head of the TM. Relaxing this last constraint brings to a different notion of simulation and construction, like, for example, in Construction 2.

14.1.2 *Construction 2*

Construction 1 needs three steps in order that both the movement of the TM head and the update of a the state are effectively simulated. Since the asymptotic time needed for the simulation is already optimal, the goal is now to reduce the multiplicative constant $\frac{3}{2}$ in the asymptotic time from Proposition 14.1.4 without loss of generality. Next construction is an example of how this constant can be reduced down to $\frac{1}{2}$, but only by tying the construction to a specific updating sequence. This shows that there is a trade-off between a general construction (i.e.,, in which every universal sequence can be used) and one producing a faster simulation. If this gap actually exists or if it can be eliminated remains an open question.

Given a TM $\mathcal{M} = (Q, \Sigma, \Gamma, b, \delta, q_0, F)$ build a family of fully-ACA $C_\tau = (A, \lambda, 1, \tau)$ such that $A = \Gamma \times Q \times D \times C$, where $D = \{L, R\}$, $C = \{\textit{active}, \textit{inactive}\}$ and $\lambda : A^3 \to A$ is defined as follows

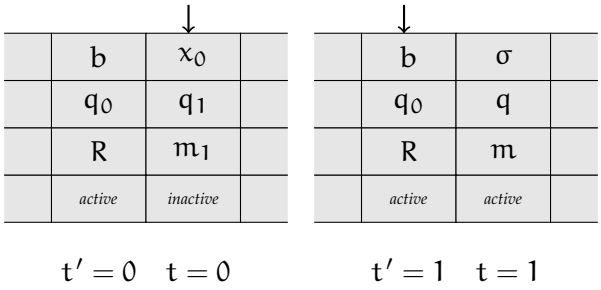


Figure 34: Simulation of the first step of a TM using a fully-ACA built by construction 2. The fully-ACA and TM times are denoted by $t' = \zeta(t)$ and t, respectively. The arrow points at the current active cell of the fully-ACA. The transition function of the simulated TM is such that $\delta(q_0, x_0) = (q, \sigma, m)$.

$$\lambda(u,v,z) = \begin{cases} (\sigma, q, m, \textit{active}) & \text{if } u = (\sigma_u, q_u, R, \textit{active}),\ v = (\sigma_v, q_v, m_v, c_v),\ z = (\sigma_z, q_z, m_z, \textit{inactive}) \text{ and } \delta(q_u, \sigma_v) = (q, \sigma, m) \\ (\sigma, q, m, \textit{active}) & \text{if } u = (\sigma_u, q_u, m_u, \textit{inactive}),\ v = (\sigma_v, q_v, m_v, c_v),\ z = (\sigma_z, q_z, L, \textit{active}) \text{ and } \delta(q_z, \sigma_v) = (q, \sigma, m) \\ (\sigma_v, q_v, m_v, \textit{inactive}) & \text{otherwise} \end{cases}$$

The initial configuration used for the simulation is the same as the one given for Construction 1. Such fully-ACA C_τ can simulate $\mathcal{M}$ only for sequences τ which are able to suitably store a piece of information concerning the position of the TM head. Indeed, for the specific updating sequence $\tau = s_0 s_1 s_2 ...$, the fully-ACA C_τ simulates $\mathcal{M}$. Moreover, using similar techniques as in Theorem 14.1.3, one can prove that C_τ simulates $\mathcal{M}$ on any input with a total running time $\sum_{i=0}^{T(n)} |s_i| = \sum_{i=0}^{T(n)} (i+1) = O\left(T(n)^2\right)$,

where $T(n)$ is the running time of $\mathcal{M}$ (see Figure 34 for the simulation of one $\mathcal{M}$ step) and the multiplicative constant is $\frac{1}{2}$ instead of $\frac{3}{2}$.

14.1.3 *Construction 3*

Construction 1 and 2 assume that potentially all cells of the fully-ACA cooperate to the simulation of the TM. Assume now that only a subset of cells participates to the simulation and the others are somehow inactive. In this section, we are going to show that the fully-ACA can still (scattered strictly) simulate any TM whenever the updating sequence has some specific properties.

A set $S \subset \mathbb{Z}$ is *syndetic* if there exists some finite $E \subset \mathbb{Z}$ such that $\cup_{n\in E}(S-n) = \mathbb{Z}$, where $(S-n) = \{k \in \mathbb{Z} \mid k+n \in S\}$. Syndetic sets have bounded gaps i.e., there exists $g \in \mathbb{N}$ (which depends on S) such that for any $h \in \mathbb{Z}$, $\{h, h+1, \ldots, h+g\} \cap S \neq \emptyset$. Given a sequence $\alpha = \{\alpha_i\}_{i\in\mathbb{N}}$, the *support* of α is the set $supp(\alpha) = \cup_{i\in\mathbb{N}}\{\alpha_i\}$.

NOTATION. To shorten up the notation in what follows, given an ordered sequence of states $u^{(-r)}, \ldots, u^{(0)}, \ldots, u^{(r)}$ and $k \in C$, denote by $E_R(k)$ the set $\left\{i \in [1, r] \mid \Pi_C(u^{(i)}) = k\right\}$ and, similarly, by $E_L(k)$ the set $\left\{i \in [-r, -1] \mid \Pi_C(u^{(i)}) = k\right\}$. Finally, denote $j_R^{(k)} = \min E_R(k)$ if $E_R(k) \neq \emptyset$ and $j_L^{(k)} = \max E_L(k)$ if $E_L(k) \neq \emptyset$.

Given a TM $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, F)$ build a family of fully-ACA $C = (A, \lambda, r, \tau)$ such that $A = \Gamma \times Q \times D \times C$, $D = \{L, R\}$, and C is the set $\{ready, active, inactive, disabled\}$. The local rule $\lambda : A^{2r+1} \to A$ is defined as follows (when the movement of the head is to the right):

$$\lambda(u^{(-r)}, \ldots, u^{(0)}, \ldots, u^{(r)}) =$$

$$\begin{cases} & \text{if } E_L(active) \neq \emptyset, \\ (\sigma, q, m, ready) & u^{(j_L^{(active)})} = (\sigma_u, q_u, R, active), \\ & u^{(0)} = (\sigma_v, q_v, m_v, inactive), \\ & \text{and } \delta(q_u, \sigma_v) = (q, \sigma, m) \\ & \text{if } u^{(0)} = (\sigma_v, q_v, R, active), \\ (\sigma_v, q_v, R, inactive) & E_R(ready) \neq \emptyset, \\ & \text{and } u^{(j_R^{(ready)})} = (\sigma_z, q_z, m_z, ready) \\ & \text{if } E_L(inactive) \neq \emptyset, \\ (\sigma_v, q_v, m_v, active) & u^{(j_L^{(inactive)})} = (\sigma_u, q_u, R, inactive), \\ & u^{(0)} = (\sigma_v, q_v, m_v, ready) \end{cases}$$

	b	x_0	b	b	x_1	b	x_2	
	q_0	q	q	q	q	q	q	
	R	m	m	m	m	m	m	
	active	*inactive*	*disabled*	*disabled*	*inactive*	*disabled*	*inactive*	

Figure 35: The initial configuration of a fully-ACA that scattered strictly simulates a TM. The dark cells are the disabled ones (i.e.,, they do not contribute to the simulation).

Similarly, when the movement of the head is to the left:

$$\lambda(u^{(-r)},\dots,u^{(0)},\dots,u^{(r)}) = \begin{cases} & \text{if } E_R(\textit{active}) \neq \emptyset, \\ (\sigma, q, m, \textit{ready}) & u^{(j_R^{(\textit{active})})} = (\sigma_z, q_z, L, \textit{active}), \\ & u^{(0)} = (\sigma_v, q_v, m_v, \textit{inactive}) \\ & \text{and } \delta(q_z, \sigma_v) = (q, \sigma, m) \\ & \text{if } u^{(0)} = (\sigma_v, q_v, L, \textit{active}), \\ (\sigma_v, q_v, L, \textit{inactive}) & E_L(\textit{ready}) \neq \emptyset, \\ & u^{(j_L^{(\textit{ready})})} = (\sigma_u, q_u, m_u, \textit{ready}) \\ & \text{if } E_R(\textit{inactive}) \neq \emptyset, \\ (\sigma_v, q_v, m_v, \textit{active}) & u^{(j_R^{(\textit{inactive})})} = (\sigma_z, q_z, L, \textit{inactive}), \\ & u^{(0)} = (\sigma_v, q_v, m_v, \textit{ready}) \end{cases}$$

In all the other cases $\lambda(u^{(-r)},\dots,u^{(0)},\dots,u^{(r)}) = u^{(0)}$.

In order to be able to (scattered strictly) simulate a TM on input $x_0 \dots x_{n-1} \in \Sigma^*$, fully-ACA given by the above construction have to be started on the following configuration c

$$\forall i \in \mathbb{Z},\ c_i^{\alpha} = \begin{cases} (b, q_0, R, \textit{active}) & \text{if } i = \alpha_{-1} \\ (x_i, q, m, \textit{inactive}) & \text{if } i \in \{\alpha_0, \dots, \alpha_{n-1}\} \\ (b, q, m, \textit{inactive}) & \text{if } i \in \textit{supp}(\alpha) \setminus \{\alpha_0, \dots, \alpha_{n-1}\} \\ (b, q, m, \textit{disabled}) & \text{otherwise} \end{cases}$$

where α is a subsequence of τ such that $\alpha_0 < \alpha_1 < \dots < \alpha_n$, and $q \in Q$, $m \in D$ are arbitrarily chosen. An example of an initial configuration can be found on Figure 35. Clearly, the whole construction (and hence the simulation) depends on τ and its subsequence α. The following result characterizes them.

Theorem 14.1.5. *A fully-ACA $C = (A, \lambda, r, \tau)$ given by Construction 3 scattered strictly simulates any TM on any input if and only if τ contains*

a subsequence α such that $supp(\alpha)$ is a syndetic set and α is universal on $supp(\alpha)$.

Proof. For any TM $\mathcal{M} = (Q, \Sigma, \Gamma, b, \delta, q_0, F)$ consider a fully-ACA $C = (A, \lambda, r, \tau)$ given by Construction 3. Assume that C scattered strictly simulates $\mathcal{M}$. First of all, let us prove that $|supp(\tau)| = \infty$. Indeed, if $|supp(\tau)| < \infty$, only a finite number of cells can be used for simulation and therefore, according to Condition 2 of Definition 14.1.2 only a finite portion of the tape can be simulated. Choose a subsequence α of τ such that $\alpha_0 < \alpha_1 < \ldots < \alpha_n$ (this is possible since $|supp(\tau)| = \infty$). By contradiction, assume that no sequence α is universal on an infinite set. This means that the set of cells that can be updated infinitely many times is finite or empty. Without loss of generality assume that it has finite cardinality and $j > 0$ is the maximal of its elements (the case $j \leqslant 0$ is similar). Let k be the index of last occurrence of j in α. Consider the TM $\mathcal{M}$ from the proof of Theorem 14.1.3 on the empty input. Since, for all $t > k$, $f^{(t)}(c^\alpha)_j = c^\alpha_j$, Condition 2 of Definition 14.1.2 is violated.

Now, always by contradiction, assume that there exist subsequences of τ but none of them has a syndetic support set on which it is universal. This means that there are larger and larger sets $[a, b] \subset \mathbb{N}$ not contained in $supp(\alpha)$. Choose one of them such that $b - a > r$ and let $\mathcal{M}$ be the TM which on the empty input writes $2b$ symbols 1 as an output. Set $h = \min_{j \in supp(\alpha)}\{b < j\}$. According to the definition of λ, for all $t \in \mathbb{N}$, $f^{(t)}(c^\alpha)_h = c^\alpha_h$. Hence Condition 2 of Definition 14.1.2 is false.

Therefore if C scattered strictly simulates $\mathcal{M}$, τ has to contain a subsequence α with a syndetic support on which α is universal.

On the other hand, assume that τ contains an subsequence α with a syndetic support on which α is universal. Then, there exists a subsequence α' with $\alpha'_0 < \alpha'_1 < \ldots < \alpha'_n$, where n is the length of the input of $\mathcal{M}$. Build the initial configuration $c^{\alpha'}$ as described in Construction 3. Since $supp(\alpha)$ has bounded gaps, let p be the longest one and set the radius $r = p$. The rest of the proof is essentially the same as the one given for Theorem 14.1.3 with α' playing the role of τ. □

Proposition 14.1.6. *For any TM $\mathcal{M}$ that executes in time $T(n)$, consider a fully-ACA C given by Construction 3 and an updating sequence τ containing a subsequence α with syndetic support on which α is universal. Then, C scattered strictly simulates $\mathcal{M}$ in time $O\left(T(n)^2\right)$.*

Proof. Consider $C = (A, \lambda, p, \tau)$ where λ and A are as in Construction 3. For $i \in \mathbb{N}$, let s_i be as in the proof of Proposition 14.1.4. Let α be the subsequence of τ given in the hypothesis and let $p \in \mathbb{N}$ be the minimal gap. Consider the subsequence $\alpha' = s_{2p}s_{4p}\ldots s_{2ip}\ldots$ Similarly to the proof of Proposition 14.1.4, $s_{2ip}s_{4ip}s_{6ip}$ can be used to encode the simulation of a step of $\mathcal{M}$. The total length of the simulation is

then bounded by $\sum_{i=1}^{T(n)} |s_{2ip}| + |s_{4ip}| + |s_{6ip}| = \sum_{i=1}^{T(n)} 12ip + 3 = O\left(T(n)^2\right)$. □

14.2 UPDATING SEQUENCES GENERATED BY RANDOM WALKS

The quadratic slowdown in the simulation time observed in the previous section is essentially due to the signalling for the active cell (token) has to go back and forth through lots of cells that are not interested by the current calculation. TM heads are the place where the calculation takes place and they can move only one cell at a time. This simple remark induced us to modify the definition of fully-ACA introducing some randomness in the distribution of the token i.e., the token can go right with probability p and left with probability q for example. In this way a certain degree of non-determinism is introduced producing a time speed-up.

More formally, consider a countable number of independent identically distributed random variables X_i with values in $\{+1, -1\}$. The position of the token (active cell) at time $t \in \mathbb{N}$ can be represented by $\tau_t = \sum_{i=0} X_i$ which is a random variable with values in $\mathbb{Z}$. Therefore, practically speaking, the updating sequence is a random walk (see [111] for more on random walks). In order to simplify the exposition, in the sequel, we will assume that all X_i have uniform Bernoulli distribution. The generalisation of the results to the non-uniform case is strightforward. The following definition adapts the notion of fully-ACA to this new setting.

Definition 14.2.1. A *random-walk fully-ACA* (or *rw-ACA* for short) is the class of all fully-ACA (A, λ, r, τ) where τ is generated by a simple random walk starting at position $\tau_0 = 0$.

Remark. The definition of C*rw* is consistent with our purposes. Indeed, consider any TM $\mathcal{M}$. Since 1D random walks pass through all sites infinitely many times with probability 1, by Theorem 14.1.3, the the rw-ACA C*rw* with alphabet and local rule given by Construction 1 strictly simulates $\mathcal{M}$ with probability 1.

Concerning simulation time, the following proposition shows that any TM executing in time $T(n)$ can be simulated by rw-ACA in $O(T(n))$ steps. This means that a simulation using rw-ACA can be faster than a deterministic one performed by fully-ACA. Unfortunately, the probability of such a simulation decreases exponentially with $T(n)$.

Proposition 14.2.1. *For every TM $\mathcal{M}$ halting in $T(n)$ steps there exists a rw-ACA C*rw *simulating $\mathcal{M}$ in time $3T(n)$ with probability $2^{-3T(n)}$.*

Proof. Using Construction 1, the simulation of one TM step requires at least 3 cell updates by the fully-ACA and then a TM halting in $T(n)$ steps requires at least $3T(n)$ applications of the fully-ACA local

rule. Since the involved cells are adjacent, random walks are potentially able to produce the sequence where the first $3T(n)$ elements are exactly the cells that need to be updated. The number of possible sequences of length $3T(n)$ generated by a random walk is $2^{3T(n)}$. So, the probability (favorable cases over total number of cases) of obtaining the right sequence simulating the TM is $2^{-3T(n)}$. □

14.2.1 *Bounded Random Walks*

In the previous section we have seen that the simulation of a TM by a fully-ACA can be speed-up to linear time at the cost of low probability in getting the correct result. On the other hand, if one allows infinite simulation time then we obtain the correct result with probability one (see Remark 14.2). There is also another possible tradeoff: simulation space.

The new idea is to use update sequences generated by a random walk *bounded* between $-n$ and n. Since 1D random walks pass infinitely many times through every cell with probability 1, the probability of obtaining a bounded sequence is 0. Nonetheless bounded sequences are interesting since when modeling real systems the used space needs to be large enough to complete the computation but is certainly not infinite.

Note that the considered bounded sequences can be generated by random walks on finite linear graphs (i.e.,, in which the next node in the random walk is uniformly selected among the neighbors of the current node). Recall that the *expected cover time* for a random walk is the expected time after which every node in the graph has been visited. The expected cover time for a linear graph is $O(|V|^2)$ (see [24] for example).

Proposition 14.2.2. *For any TM $\mathcal{M}$ halting in $T(n)$ steps, there exists an rw-ACA Crw working on the bounded sequences between $-T(n)$ and $T(n)$, simulates $\mathcal{M}$ in an expected time $O\left(T(n)^3\right)$.*

Proof. If a random walk $\{z_i\}_{i\in\mathbb{N}}$ is bounded between $-T(n)$ and $T(n)$ then it can be considered as a random walk on a linear graph with $2T(n)+1$ vertexes and $2T(n)$ edges. The expected cover time of a linear graph is bounded by $O((2T(n)+1)^2) = O(T(n)^2)$. Consider the local rule given by Construction 1. The simulation of $\mathcal{M}$ requires $3T(n)$ fully-ACA applications and one graph cover is performed for each fully-ACA application assuring the involved cells are updated. By linearity of the expectation operator, the rw-ACA Crw simulates $\mathcal{M}$ in expected time $\sum_{i=1}^{3T(n)} O(T(n)^2) = O(T(n)^3)$. □

15

M-ASYNCHRONOUS CA

In this chapter, we introduce a model of ACA (m-ACA) in which there is partial synchronicity given by a *fair* probability measure over the subsets of cells that can be updated. That is, a measure that, intuitively, gives to any cell the possibility to be updated and limit the difference between the most and the lest frequently updated cells.

15.1 M-ACA

In this section we introduce the notion of m-ACA as a generalization both of classical CA and of fully-ACA. The basic idea is to augment the classical CA model by a measure μ. Updating sequences will be generated using μ. In this manner, it is possible to precisely define when a property holds for almost all updating sequences or only for a negligible set of them. Differently from classical CA and fully-ACA, the definition of m-ACA can capture, depending on the probability measure used, many different updating scheme (including fully-ACA and classical CA). Thus, when proving the presence of a property for a particular class of measures, we are including many different possible updating schemes. Hopefully, this approach would reveal itself more general than many ad-hoc methods tied to a particular updating scheme.

Definition 15.1.1 (m-ACA)**.** An m-ACA $\mathcal{C}$ is a quadruple $(\Sigma, r, \lambda, \mu)$ where A is a finite alphabet, $r > 0$ is the *radius*, $\lambda : \Sigma^{2r+1} \to \Sigma$ is the *local rule* and μ is a probability measure on the Borel σ-algebra on $\mathcal{P}(\mathbb{Z})$.

Given the local rule λ and a set $U \in \mathcal{P}(\mathbb{Z})$ define $F_U : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ as follows

$$\forall x \in \Sigma^{\mathbb{Z}}, \forall i \in \mathbb{Z} \quad F_U(x)_i = \begin{cases} \lambda(x_{i-r}, \ldots, x_i, \ldots, x_{i+r}) & \text{if } i \in U \ , \\ x_i & \text{otherwise } . \end{cases}$$

Given a sequence $\upsilon \in \mathcal{P}(\mathbb{Z})^{\mathbb{N}}$ extracted using μ (all extractions are considered as independent), the *dynamics* of the m-ACA is described by the family of functions $\mathcal{T}_\upsilon = \{id, F_{\upsilon_1}, F_{\upsilon_2} \circ F_{\upsilon_1}, \ldots\}$. Remark that all the elements from $\mathcal{T}_{\gg}$ are continuous functions w.r.t.

the metric d. The *orbit* of a configuration $x \in \Sigma^{\mathbb{Z}}$ is the sequence $\gamma_{x,v} = (x, F_{v_1}(x), (F_{v_2} \circ F_{v_1})(x), \ldots)$ associating with each time step $t \in \mathbb{N}$ the configuration $\gamma_{x,v}(t) = (F_{v_t} \circ \ldots \circ F_{v_1})(x)$ of the m-ACA at that time.

Remark. The notion of m-ACA includes both classical CA and fully-ACA. Indeed, the former case is obtained choosing a measure μ_1 such that $\mu_1(A) = 1$ if $\mathbb{Z} \in A$, 0 otherwise. The latter is obtained by choosing μ_2 such that $\mu_2(\mathcal{U}_i) > 0$ for $i \in \mathbb{Z}$ and $\mu_2(\mathcal{U}_i \cap \mathcal{U}_j) = 0$ for $i \neq j$. Note that in the last case we cannot have a shift-invariant measure.

In order to study the core behavior of the model, the "extremal" cases reported in Remark 15.1 should be avoided. This goal can be reached, for instance, by adding some further requirements to the measure μ used to produce the updating sequences. Therefore, the following "fairness" requirements are put on μ:

1. at any time step (extraction), each single cell is updated or non updated with a positive probability. This means that events like "always updated" or "always non updated" for the same cell happen with probability 0.
2. for any cell the probability of being updated is independent from the probability of being updated of any other cell.
3. any event that fix the update/non update condition of an infinite number of cells has zero probability.

The following definition formalizes the above requirements.

Definition 15.1.2. A probability measure μ on the Borel σ-algebra on $\mathcal{P}(\mathbb{Z})$ is *fair* iff:

1. $\forall i \in \mathbb{Z}, 0 < \mu(\mathcal{U}_i) < 1$.
2. $\forall A \subseteq \mathbb{Z}$ with A finite, $\mu\left(\bigcap_{a \in A} U_a\right) = \prod_{a \in A} \mu(U_a)$, where each U_a can be either $\mathcal{U}_a$ or $\overline{\mathcal{U}}_a$.
3. $\forall A \subseteq \mathbb{Z}$ with A infinite, $\mu\left(\bigcap_{a \in A} U_a\right) = 0$, where each U_a can be either $\mathcal{U}_a$ or $\overline{\mathcal{U}}_a$.

Remark. The class of α-asynchronous CA are an example of m-ACA. Indeed, each cell i is updated with a fixed probability $\alpha > 0$. This is equivalent to take $\mu(\mathcal{U}_i) = \alpha$ for every $i \in \mathbb{Z}$. Note that m-ACA also allow the non-shift invariant case of a probability of updating a cell that depends on its position.

Remark. For both classical CA and fully-ACA it is impossible to define a fair measure to generate the correct updating sequences. In fact, for classical CA we need that $\mu(\mathcal{U}_i) = 1$ for every $i \in \mathbb{Z}$. Also, for fully-ACA we need that $\mu(\mathcal{U}_i \cap \mathcal{U}_j) = 0$ whenever $i \neq j$.

The following lemma illustrates a first consequence of the fairness requirements on the measure μ: for a given cell, the probability of being updated or not can be arbitrarily close neither to 0 nor to 1.

Lemma 15.1.1. *Consider a fair measure μ on the Borel σ-algebra on $\mathcal{P}(\mathbb{Z})$. There exist $h, k \in \mathbb{R}$ with $0 < h \leqslant k < 1$ such that $\forall i \in \mathbb{Z}$, $h \leqslant \mu(\mathcal{U}_i) \leqslant k$.*

Proof. Let $h = \inf\{\mu(\mathcal{U}_i) \mid i \in \mathbb{Z}\} \geqslant 0$. We claim that $h > 0$. By contradiction, assume $h = 0$. Then, there exists a sequence $\{a_i\}_{i\in\mathbb{N}}$ of elements of $\mathbb{Z}$ such that $\mu(\mathcal{U}_{a_i}) \leqslant \frac{1}{(i+2)^2}$. Remark that $\prod_{i\in\mathbb{N}} \mu(\overline{\mathcal{U}}_{a_i}) \geqslant \prod_{i=2}^{+\infty}\left(1-\frac{1}{i^2}\right) = \frac{1}{2}$. This fact contradicts the assumption that every event concerning the update/non update condition of infinitely many cells has probability 0 (i.e., $\prod_{i\in A} \mu(\overline{\mathcal{U}}_{a_i}) = 0$ for every infinite subset $A \subseteq \mathbb{Z}$).

Let $k = \sup\{\mu(\mathcal{U}_i) \mid i \in \mathbb{Z}\} \leqslant 1$. We claim that $k < 1$. Again, by contradiction, assume that $k = 1$. Then, there exists a sequence $\{a_i\}_{i\in\mathbb{N}}$ of elements of $\mathbb{Z}$ such that $\mu(\mathcal{U}_{a_i}) \geqslant \frac{(i+2)^3-1}{(i+2)^3+1}$. Remark that $\prod_{i\in\mathbb{N}} \mu(\mathcal{U}_{a_i}) \geqslant \prod_{i=2}^{+\infty} \frac{i^3-1}{i^3+1} = \frac{2}{3}$ contradicting the assumption that μ is fair. □

Remark. As a direct consequence of Lemma 15.1.1, the measure of the complements of ultrafilters is also bounded between $1-k$ and $1-h$. Moreover, there exists $0 < p \leqslant q < 1$ such that for every finite set $A \subseteq \mathbb{Z}$, $p^{|A|} \leqslant \mu(\bigcap_{a\in A} \mathcal{U}_a) \leqslant q^{|A|}$.

Denote by $\mathcal{S}$ the set of all updating sequences. In the model proposed in this chapter, μ is used to extract the subset of $\mathbb{Z}$ indicating which cells are allowed to be updated. At each time step, a new extraction is performed and we made the hypothesis that extractions are independent. Therefore, it is natural to consider the product measure μ_s of the measure μ to measure sets of updating sequences i.e., subsets of $\mathcal{S}$ (μ_s always exists and is unique, see [89, Thm. B, pag. 157]).

In the sequel, the notion of control pattern will play a central role in the proofs concerning injectivity and surjectivity properties.

A *control pattern* of length n is an updating mask for n contiguous cells. More formally,

Definition 15.1.3. A *control pattern* of length $n \in \mathbb{N}_+$ is a pair $B = (B_1, B_2)$ of sets such that $\{B_1, B_2\}$ is a partition of $[0, n)$.

Definition 15.1.4. Given a control pattern $B = (B_1, B_2)$ and $\tau \in \mathcal{P}(\mathbb{Z})$, B is *represented in τ at position k* iff $\forall b \in B_1$, $b + k \in \tau$ and $\forall c \in B_2$, $c + k \notin \tau$. B is *represented at least m times in τ* if there exist m distinct positions at which B is represented. If B is represented at least m times for any $m \in \mathbb{N}_+$, then B is said to be represented infinitely many times.

Lemma 15.1.2. *For any fair measure μ, the following statements hold.*

1. *For every* $n \in \mathbb{N}_+$ *and for every control pattern* B *of length* n*, the set of all* $\tau \in \mathcal{P}(\mathbb{Z})$ *in which* B *is represented infinitely many times has measure* 1.

2. *Every countable family of subsets of* $\mathbb{Z}$ *has measure* 0.

Proof.

1. Consider a control pattern B of length n. For any $t \in \mathbb{N}$, let E_t be the set of all subsets of $\mathbb{Z}$ in which the pattern is represented at position tn. Note that $\mu(E_t \cap E_q) = \mu(E_t)\mu(E_q)$ whenever $t \neq q$. Each E_t has a positive measure and, by Lemma 15.1.1 there exists $p > 0$ such that for all $t \in \mathbb{N}$, $\mu(E_t) \geqslant p^n$.
Since $\sum_{i=0}^{+\infty} \mu(E_t) \geqslant \sum_{i=0}^{+\infty} p^n = \infty$, by the second Borel-Cantelli Lemma [28], it follows that
$$\mu(\limsup_{t\to\infty} E_t) = \mu\left(\bigcap_{i=1}^{\infty}\bigcup_{j=i}^{\infty} E_j\right) = 1$$
Hence, B is represented almost surely infinitely many times.

2. Consider $A \subseteq \mathbb{Z}$ then $\mu(\{A\}) = \mu\left(\cap_{a\in A} U_a \bigcap \cap_{a\notin A} \overline{U}_a\right) = p$. By Condition 3. from Definition 15.1.2, we have $p = 0$. Finally, the thesis follows by countable additivity of μ.

□

By Lemma 15.1.2, for every control pattern B, the set of $\tau \in \mathcal{P}(\mathbb{Z})$ in which B is represented infinitely many times has full measure. Moreover, for every control pattern B and every position $i \in \mathbb{Z}$, the set of $\tau \in \mathcal{P}(\mathbb{Z})$ in which B is represented infinitely many times at positions greater (or smaller) than i has full measure too. Furthermore, from item 2. of Lemma 15.1.2, one can deduce that the set of all $\tau \in \mathcal{P}(\mathbb{Z})$ for which the number of updated cells is finite or cofinite (i.e., with finite complement) has null measure.

Definition 15.1.5. An m-ACA $C = (\Sigma, \lambda, r, \mu)$ is surjective (resp. injective) iff $\forall U \in \mathcal{P}(\mathbb{Z})$, F_U is surjective (resp. injective). The m-ACA C is μ-*almost surely* surjective (resp., injective) iff

$$\mu(\{U \in \mathcal{P}(\mathbb{Z}),\ F_U \text{ is surjective (resp. injective)}\}) = 1\ .$$

From Definition 15.1.5, it trivially follows that if an m-ACA $C = (\Sigma, \lambda, r, \mu)$ is surjective, the corresponding CA (Σ, λ, r) is surjective. The *shift* CA $(\{0,1\}, \sigma, 1)$ where $\sigma(a, b, c) = c$ for all $a, b, c \in \{0,1\}$ is bijective but its corresponding m-ACA $(\{0,1\}, \sigma, 1, \mu)$ is not μ-surjective for any fair μ.

Recall, from Chapter 13, that for fully-ACA, injectivity is equivalent to surjectivity [126]. Indeed, a similar result holds also for m-ACA as illustrated by the following proposition.

Proposition 15.1.3. *Consider an* $\mathfrak{m}$*-ACA* $\mathcal{C} = (\Sigma, \lambda, r, \mu)$*, where* μ *is fair. Then,* $\mathcal{C}$ *is* μ*-almost surely surjective iff* $\mathcal{C}$ *is* μ*-almost surely injective.*

Proof. Consider a control pattern $B = (\emptyset, [0, r))$ and let $\tau \in \mathcal{P}(\mathbb{Z})$ be generated by μ. By Lemma 15.1.2, B is represented infinitely many times in τ. Hence, τ can be decomposed in intervals separated by gaps of non updated cells of length at least r. Call $\mathcal{F} = \{[h_i, k_i]\}_{i\in\mathbb{Z}}$ the sequence of such intervals, namely $\forall i, j \in \mathbb{Z}$, $i < j$ implies $k_i + r < h_j$ (i.e., the intervals are ordered and the distance between the extremes is at least r) and $\forall j \in \mathbb{Z}$, if $j \notin \bigcup_{i\in\mathbb{Z}}[h_i, k_i]$ then $\forall x \in \Sigma^{\mathbb{Z}}$, $F_\tau(x)_j = x_j$.

Note that F_τ is injective iff $\forall i \in \mathbb{Z}$ and $\forall x, y \in \Sigma^{\mathbb{Z}}$, $x_{[h_i,k_i]} \neq y_{[h_i,k_i]}$ implies $F_\tau(x)_{[h_i,k_i]} \neq F_\tau(y)_{[h_i,k_i]}$. This is equivalent to the following condition: $\forall i \in \mathbb{Z}$ and $\forall w \in \Sigma^{(k_i-h_i+1)}$, $\exists x \in \Sigma^{\mathbb{Z}}$ such that $F_\tau(x)_{[h_i,k_i]} = w$, that in its turn is equivalent to the surjectivity of F_τ. Thus, F_τ is injective iff it is surjective. □

Remark. By the proof of Proposition 15.1.3, it follows that every $\mathfrak{m}$-ACA that is μ-almost surely injective has a center-permutive local rule for any fair measure μ. Indeed, this property holds since the set of all τ in which the control pattern $B = (\{r\}, [0, r) \cup (r, 2r + 1))$ occurs has full measure for any fair measure. When B occurs at some position in τ, injectivity or surjectivity of F_τ holds only if the local rule is center-permutive.

Example 15.1.1. Consider the $\mathfrak{m}$-ACA $\mathcal{C} = (\{0, 1\}, \lambda, 1, \mu)$ where μ is fair and $\forall a, b, c \in \{0, 1\}$, $\lambda(a, b, c) = a$ xor b. Since $\forall n \in \mathbb{N}_+$ and $\forall x, y \in \Sigma^{\mathbb{Z}}$, $x_{[0,n)} \neq y_{[0,n)}$ implies $F(x)_{[0,n)} \neq F(y)_{[0,n)}$, $\mathcal{C}$ is almost surely injective and surjective. Figure 36 gives an example of evolution of $\mathcal{C}$.

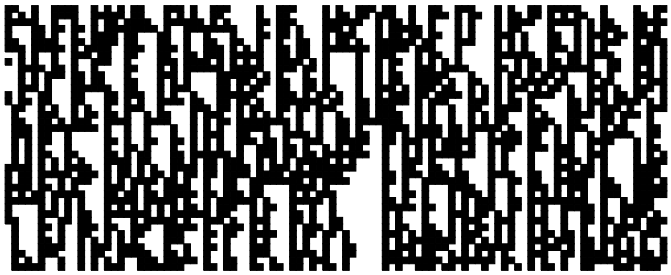


Figure 36: The space-time diagram of the probabilistic xor CA of Example 15.1.1. A black (resp., white) box stands for a 1 (resp., 0). μ is the uniform measure over $\{0, 1\}^{\mathbb{Z}}$. Time goes downward.

Similarly to classical CA, we introduce the concept of *diamond* and relate it to injectivity/surjectivity properties [91].

Definition 15.1.6. An $\mathfrak{m}$-ACA $\mathcal{C} = (\Sigma, \lambda, r, \mu)$ has a *diamond* if there exist a control pattern $B = (B_1, B_2)$ of length $n \in \mathbb{N}_+$ and words

$z, w \in \Sigma^n$, with $z \neq w$, $u, v \in \Sigma^r$ such that $\lambda_B(uwv) = \lambda_B(uzv)$ where $\lambda_B : \Sigma^{n+2r} \to \Sigma^n$ is defined as

$$\forall \alpha \in \Sigma^{n+2r}, \quad \forall i \in [0, n), \qquad \lambda_B(\alpha)_i = \begin{cases} \lambda(\alpha_{[i,i+2r)}) & \text{if } i + r \in B_1 \\ \alpha_{i+r} & \text{otherwise} \end{cases}$$

NOTATION. For notational convenience, for any $\tau \subseteq \mathbb{Z}$ and for all $h, k \in \mathbb{N}$ with $h < k$, let $\tau_{[h,k)}$ be the pattern (B_1, B_2) where $B_1 = \{i \in [0, k-h) \mid i + h \in \tau\}$ and $B_2 = \{i \in [0, k-h) \mid i + h \notin \tau\}$.

Proposition 15.1.4. *Consider an* m*-ACA* $\mathcal{C} = (\Sigma, \lambda, r, \mu)$ *with* μ *fair. Then the following statements are equivalent:*

1. $\mathcal{C}$ *is not* μ*-almost surely injective.*
2. $\mathcal{C}$ *has a diamond.*

Proof.

1. $\Rightarrow$ 2. If $\mathcal{C}$ is not almost surely injective then there exists a collection A of subsets of $\mathbb{Z}$ with positive measure such that $\forall \tau \in A$, $\exists x, y \in \Sigma^{\mathbb{Z}}$, with $x \neq y$ and $F_\tau(x) = F_\tau(y)$. Since A has positive measure, there exists at least a set $\tau \in A$ in which the pattern $B = (\emptyset, [0, r))$ is represented infinitely many times. Let $x, y \in \Sigma^{\mathbb{Z}}$ be two distinct configurations such that $F_\tau(x) = F_\tau(y)$ and let $i \in \mathbb{Z}$ be a position such that $x_i \neq y_i$. Let h, k be two positions such that $h + r < i < k - r$ and B is represented in τ at positions h and $k - r$. It is immediate that $x_{[h,h+r)} = y_{[h,h+r)}$, $x_{[k-r,k)} = y_{[k-r,k)}$, $x_{[h+r,k-r)} \neq x_{[h+r,k-r)}$, and $\lambda_{\tau_{[h,k)}}(x_{[h,k)}) = \lambda_{\tau_{[h,k)}}(y_{[h,k)})$. Thus, there exists a diamond.

2. $\Rightarrow$ 1. Suppose that there exists a diamond. Let $B = (B_1, B_2)$ and $z, w \in \Sigma^n$, $u, v \in \Sigma^r$ be the control pattern of length n and the words, respectively, that define the diamond. Let $B' = (\emptyset, [0, r))$. An integer $i \in \tau \subseteq \mathbb{Z}$ is said to have property I if it is a multiple of $n + 2r$ such that B' is represented in τ at positions i and $i + n + r$ and B is represented in τ at position $i + r$. Let $\tau \in \mathcal{P}(\mathbb{Z})$ be a set with property I for an integer i. Consider the configurations $x, y \in \Sigma^{\mathbb{Z}}$ with $x_j = y_j$ for all $j \notin [i, i + n + 2r)$, $x_{[i,i+n+2r)} = uwv$, and $y_{[i,i+n+2r)} = uzv$. Then, $F_\tau(x) = F_\tau(y)$. Thus F_τ is not injective. Let A be the collection of all subsets of $\mathbb{Z}$ having at least an integer with property I. By the same idea used in the proof of Lemma 15.1.2, one can show that $\mu(A) = 1$. Thus $\mathcal{C}$ is not μ-almost surely injective.

□

In this section, we adapt the notion of expansivity to m-ACA and we show that it is related to surjectivity in a similar way to classical CA [60, 119].

Definition 15.1.7. An m-ACA $\mathcal{C} = (\Sigma, \lambda, r, \mu)$ is υ-*expansive*, for a sequence $\upsilon \in \mathcal{P}(\mathbb{Z})^{\mathbb{N}}$ extracted by μ, if the family $\mathcal{T}_\upsilon$ is expansive. $\mathcal{C}$ is said to be *almost surely expansive*, if the set of sequences $\upsilon \in \mathcal{P}(\mathbb{Z})^{\mathbb{N}}$ such that $\mathcal{C}$ is υ-expansive has measure 1 (w.r.t. the measure μ_s).

Proposition 15.1.5. *Let $\mathcal{C} = (\Sigma, \lambda, r, \mu)$ be an almost surely expansive m-ACA. Then, $\mathcal{C}$ is μ-almost surely injective.*

Proof. Suppose that $\mathcal{C}$ is not μ-almost surely injective. By Proposition 15.1.4, $\mathcal{C}$ has a diamond. So, there exist a control pattern B of length n, and words $w, z \in \Sigma^n$, $u, v \in \Sigma^r$ such that $\lambda_B(uwv) = \lambda_B(uzv)$. By Lemma 15.1.2, for any $k \in \mathbb{N}$, B is represented in almost all sets $\tau \subseteq \mathbb{Z}$ infinitely many times at positions greater than k. For every $k \in \mathbb{N}$, define now the non empty set $A_k = \{(x, y) \in \Sigma^{\mathbb{Z}} \times \Sigma^{\mathbb{Z}} \mid \exists h > k,\ \forall i \notin [h, h + 2r + n)\ x_i = y_i,\ x_{[h,h+2r+n)} = uwv,\ y_{[h,h+2r+n)} = uzv\}$. Since for almost all $\tau \subseteq \mathbb{Z}$, there are infinitely many pairs $(x, y) \in A_k$ such that $F_\tau(x) = F_\tau(y)$, the set of sequences υ having τ as first component has non null measure (w.r.t. μ_s). This means that $\mu_s\{\upsilon \in \mathcal{P}(\mathbb{Z})^{\mathbb{N}} \mid \mathcal{C} \text{ is not } \upsilon\text{-expansive}\} \neq 0$. By additivity of μ_s, it follows that $\mu_s\{\upsilon \in \mathcal{P}(\mathbb{Z})^{\mathbb{N}} \mid \mathcal{C} \text{ is } \upsilon\text{-expansive}\} \neq 1$, i.e., $\mathcal{C}$ is not almost surely expansive. □

Example 15.1.2. Let $\mathcal{C} = (\Sigma, \lambda, r, \mu)$ be an m-ACA with $\lambda : \{0, 1\}^3 \to \{0, 1\}$ defined as $\lambda(a, b, c) = a$ xor c. The local rule λ is not center-permutive, hence $\mathcal{C}$ is not almost surely surjective. Note that, when we start from the two finite configurations 000 and 010, $0\lambda(0,0,0)0$ is equal to $0\lambda(0,1,0)0 = 000$ (i.e., $\mathcal{C}$ has a diamond). Thus, $\mathcal{C}$ is not almost surely expansive. Remark that, in the case of fully-ACA, λ is the local rule of an α-expansive ACA. Figure gives an example of evolution of $\mathcal{C}$.

Figure 37: Probabilistic xor CA of Example 15.1.2. A black (resp., white) box stands for a 1 (resp., 0). μ is the uniform measure over $\{0, 1\}$. Time goes downward.

15.2 FURTHER REMARKS

In this chapter we introduced a new model of asynchronicity for CA where the set of cells where the local rule is applied is given by a prob-

ability measure with some fairness conditions. This model represents a first step in trying to unify different approaches to asynchronicity under a common framework. We adapted to this new setting the usual notions of injectivity, surjectivity, diamond and expansivity. We found that both injectivity and surjectivity are almost always equivalent and they are related to a generalized version of the classical notion of diamond. Moreover, an almost always expansive CA is also almost always injective. This fact is a bit surprising since in the classical case, expansive CA are never injective.

Part V

FINAL REMARKS

16

CONCLUSIONS AND FUTURE WORKS

As final remarks, we summarize the contribution of this thesis and we state some future directions of research and open problems.

16.1 CONTRIBUTIONS

The contributions of this thesis must be placed into two different categories. One is about the single areas where new results have been found. The other one is about the different cross-pollination between the areas.

16.1.1 *Genetic Algorithms*

In the field of Genetic Algorithms theory, a new study of the dynamics of the crossover operator has been performed. With the introduction of concepts of topology, it has been possible to answer in an efficient manner the following question:
Given a population and in individual, how many generations are needed to generate the individual from the population using only crossover operations?
Also important are the tools used to answer this question. Topology has been used in different ways to study the dynamics of different evolutionary algorithms. In particular, we think that the methods used to answer this question can be extended to answer other similar questions both in Genetic Algorithms and in other evolutionary techniques.

16.1.2 *Genetic Programming*

There are multiple contributions in the field of Genetic Programming. First of all, two different measures to quantify the learning ability of Genetic Programming. These measures provide a way to find the training points that are difficult to learn, thus allowing to modify the learning process to either focus or ignore these points.

A second step has been the definition of a tunable benchmark, the K-landscapes, inspired by NK-landscapes for Genetic Algorithm. This

benchmark allows to better explore the process that allows Genetic Programming to build solutions.

Finally, a fast way of implementing Semantic GP has been devised. Semantic GP is a recently introduced GP variant that proved to be very effective. However, previous implementations could have an exponential increase in the space needed to store solutions and were computationally expansive. A new implementation that avoids these problems has been developed, thus allowing a greater applicability of Semantic GP.

16.1.3 *Reaction Systems*

The young formalism of Reaction Systems has been explored in two ways. In the first case we studied some combinatorial properties of Reaction Systems with a focus on properties that necessarily holds for "large enough" Reaction Systems.

As a second step, we used the ability of Reaction System to represent boolean functions in an effective way to develop an evolutionary version of them, called Evolutionary Reaction Systems. This new evolutionary algorithms proved to have performances comparable or superior to many other well-established machine learning techniques. These results provide a first step to make this new algorithm a widely used evolutionary algorithm.

16.1.4 *Cellular Automata*

Some dynamical and computational properties of asynchronous Cellular Automata have been studied.

For fully-asynchronous Cellular Automata a link has been found between properties of the local rule, in particular permutivity, and the global dynamical behaviour - sensitivity to initial conditions, expansivity, and transitivity. For the same class of asynchronous Cellular Automata we have studied the ability to simulate step-by-step a Turing Machine. We have found that this simulation can be performed with only a limited (polynomial) slowdown, thus showing that some biological systems that are a natural analogue of this kind of Cellular Automata could be worth studying with the aim of performing computation with them.

We have also defined and studied m-Asynchronous Cellular Automata, in which the asynchronicity is given by a probability measure over the subsets of the cells of the automaton. We studied some formal properties of these automata, proving, for example, that in almost all cases either the update of the automaton is a bijection or it is neiter injective nor surjective. This class of automata seems promising since it encompass other classes of Asynchronous Cellular Automata while avoiding to be too general to be effectively studied.

While the single improvements and discoveries made in the different areas are, by themselves, interesting, it is important to note the interactions between the different areas. This is particularly prominent in the case of Reaction Systems, in which both combinatorial techniques and evolutionary algorithms were used in their study. In all cases a - albeit limited - exchange of ideas and techniques has been carried on between the different ares. In the future, we want to extend and make it more prominent.

16.2 OPEN PROBLEMS

While there are many directions for future research and open problems, here we want to illustrate the most important ones, while referring for other open problems to the specific chapters.

16.2.1 *Genetic Algorithms*

Future research should focus on the extension of the techniques currently used for one-point crossover of fixed-length strings to other types of representation and crossover. It is important to devise an extension that is not ad-hoc for the particular representation or crossover, otherwise no general structure or comparison of algorithm would be possible.

The current model makes some simplifying assumption. It would be interesting to remove them in order to be more faithful to the real behaviour of Genetic Algorithms. For example, we are currently modeling only the presence of a particular genetic material, but not his quantity, even if this is an important aspect in the determination of Genetic Algorithms dynamics. Thus, an important part in the extension of the current model is the introduction of more and more previously non-considered aspects of Genetic Algorithms.

16.2.2 *Genetic Programming*

There are many possible directions of future research for Genetic Programming. In particular, the study and the evaluation of the performances of Genetic Programming will greatly benefit from the definition of a standard set of benchmarks. A first step in this direction has been recently made [129], with an analysis of the current benchmarks and a the formation of a community[1] to reach this goal. However, many steps remain to be taken.

Another direction of research is about Semantic Genetic Programming. This new method should be more extensively compared with traditional Genetic Programming and its formal properties should be

1 see http://gpbenchmarks.org/.

studied. In particular, some first steps toward its runtime analysis are currently being made. With the elimination of the performance penalty of Semantic Genetic Programming with respect to classical GP, we expect a wider and wider application of it. However, the current representation, while providing many advantages, does not give an easy understandable individual (because of its size when unfolded). To attaining this a simplification process must still be performed. Thus, for domains in which this aspect of GP is required, we need to study a way to balance the ability to efficiently perform thousands of generations with the need to successively perform a simplification.

16.2.3 *Reaction Systems*

There are two main directions of research for Reaction Systems. The first one is about a more extensive use of combinatorial techniques to explore and study their behaviour. This studies, if successful, will have the opportunity to establish these techniques as a powerful tool to understand Reaction Systems. Following this ideas, a new study on the minimum complexity (number of reactants, inhibitors and products) needed in a reaction to remain able to simulate any reaction systems with "less-complex" reaction systems. This study, currently in progress, is expected to give a complete characterization of the "minimum-complexity" needed in a reaction system. Hopefully, it will be possible to concentrate the study on less complex reaction systems, knowing that, after a certain size or complexity, no new dynamics is possible.

As an evolutionary algorithms, Evolutionary Reaction Systems need to be extended to continuous domains and to be more extensively tested. Only this kind of study can effectively bring this new technique among the other widely used and well established evolutionary algorithms. Furthermore, while we stressed the importance of the parallel nature of Reaction Systems, a corresponding implementation taking advantage of it is still to be done. Since more and more computationally intensive workload is being moved to GPU architectures, this point is essential in understanding the real applicability of Evolutionary Reaction Systems.

16.2.4 *Cellular Automata*

The research of Asynchronous Cellular Automata is currently fragmented due to the different ways of introducing asynchronicity. The definition and study of m-Asynchronous Cellular Automata represented a first step in providing a subdivision of the different updating scheme in classes that are large enough to cover many different schemes but still particular enough to be obtain interesting results. In

the future, we plan to provide an axiomatic framework to work with asynchronicity in Cellular Automata. By providing a set of axioms that a probability measure must respect in order to obtain a property we immediately have that any new updating scheme that is defined and that respects that axioms has the desired property, without the need to perform ad ad-hoc study.

A particularly interesting result that has been obtained is that surjectivity and injectiviy are the same in all the asynchronous models studied. Thus, we want to study the largest possible conditions in which this equality remains true. Furthermore, we are interested in discovering if there is a sharp boundary between the cases in which the equality holds and the ones in which it does not hold (i.e., if we immediately pass from an "almost always" case to an "almost never" one without nothing in between).

In conclusion, there are many possibilities to continue the research in the different topics introduced in this thesis. However, what would be also interesting is a convergence of the methods used across the different areas. In this way, it would be possible to allow a fruitful exchange of ideas between different communities.

BIBLIOGRAPHY

[1] L. Acerbi, A. Dennunzio, and E. Formenti. Shifting and lifting of cellular automata. In *CiE*, volume 4497 of *LNCS*, pages 1–10. Springer, 2007. ISBN 978-3-540-73000-2. (Cited on page 133.)

[2] L. Acerbi, A. Dennunzio, and E. Formenti. Conservation of some dynamcal properties for operations on cellular automata. *Theoretical Computer Science*, 410:3685–3693, 2009. (Cited on page 133.)

[3] H. E. Aguirre and K. Tanaka. Genetic algorithms on NK-landscapes: effects of selection, drift, mutation, and recombination. In *Proceedings of the 2003 international conference on Applications of evolutionary computing*, EvoWorkshops'03, pages 131–142, Berlin, Heidelberg, 2003. Springer-Verlag. ISBN 3-540-00976-0. (Cited on page 57.)

[4] M. Alam, M. Islam, X. Yao, and K. Murase. Recurring two-stage evolutionary programming: A novel approach for numeric optimization. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 41(5):1352 –1365, oct. 2011. ISSN 1083-4419. (Cited on page 49.)

[5] L. Altenberg. B2.7.2 NK fitness landscapes. In T. Baeck, D. Fogel, and Z. Michalewicz, editors, *Handbook of evolutionary computation*. New York: Oxford University Press, 1997. (Cited on pages 54 and 56.)

[6] P. Amar, G. Bernot, and V. Norris. Hsim: a simulation programme to study large assemblies of proteins. *Journal of Biological Physics and Chemistry*, 4:79–84, 2004. (Cited on pages 5, 133, and 135.)

[7] N. M. Amil, N. Bredeche, C. Gagné, S. Gelly, M. Schoenauer, and O. Teytaud. A statistical learning perspective of genetic programming. In *Proceedings of the 12th European Conference on Genetic Programming*, EuroGP '09, pages 327–338, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-01180-1. (Cited on page 54.)

[8] F. Archetti, S. Lanzeni, E. Messina, and L. Vanneschi. Genetic programming for human oral bioavailability of drugs. In M. Keijzer, M. Cattolico, D. Arnold, V. Babovic, C. Blum, P. Bosman, M. V. Butz, C. Coello Coello, D. Dasgupta, S. G. Ficici, J. Foster, A. Hernandez-Aguirre, G. Hornby, H. Lipson, P. McMinn, J. Moore, G. Raidl, F. Rothlauf, C. Ryan, and D. Thierens, editors, *GECCO 2006: Proc. of the 8th annual conference on Genetic and evolutionary computation*, volume 1, pages 255–262. ACM Press, 2006. ISBN 1-59593-186-4. (Cited on page 63.)

[9] F. Archetti, S. Lanzeni, E. Messina, and L. Vanneschi. Genetic programming for computational pharmacokinetics in drug discovery and development. *Genetic Programming and Evolvable Machines*, 8:413–432, 2007. ISSN 1389-2576. (Cited on pages 86, 94, 96, and 97.)

[10] D. Arnold, T. Jansen, J. Rowe, and M. Vose. 06061 executive summary – theory of evolutionary algoritms. In Arnold et al. [11]. (Cited on page 10.)

[11] D. Arnold, T. Jansen, M. D. Vose, and J. Rowe, editors. *Theory of Evolutionary Algorithms, 05.02. - 10.02.2006*, volume 06061 of *Dagstuhl Seminar Proceedings*, 2006. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany. (Cited on pages 10 and 176.)

[12] T. Bäck. The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In R. Männer and B. Manderick, editors, *PPSN*, pages 87–96. Elsevier, 1992. (Cited on page 108.)

[13] T. Bäck. Optimal mutation rates in genetic search. In *Proceedings of the fifth International Conference on Genetic Algorithms*, pages 2–8. Morgan Kaufmann, 1993. (Cited on page 108.)

[14] T. Bäck and R. Breukelaar. Using genetic algorithms to evolve behavior in cellular automata. In C. Calude, M. Dinneen, G. Paun, M. Pérez-Jiménez, and G. Rozenberg, editors, *UC*, volume 3699 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2005. ISBN 3-540-29100-8. (Cited on page 126.)

[15] P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality. *Physical Review A*, 38(1):364–374, Jul 1988. (Cited on page 57.)

[16] W. Banzhaf, F. D. Francone, and P. Nordin. The effect of extensive use of the mutation operator on generalization in genetic programming using sparse data sets. In W. E. *et al*, editor, *4th Int. Conf. on Parallel Problem Solving from Nature (PPSN96)*, pages 300–309. Springer, Berlin, 1996. (Cited on page 54.)

[17] W. Banzhaf, P. Nordin, R. Keller, and F. Francone. *Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann, San Francisco, CA, USA, Jan. 1998. (Cited on page 106.)

[18] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss. Sequential parameter optimization. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 773–780. IEEE, 2005. (Cited on pages 107 and 108.)

[19] L. Beadle and C. Johnson. Semantically driven crossover in genetic programming. In J. Wang, editor, *Proceedings of the IEEE World Congress on Computational Intelligence*, pages 111–116, Hong Kong, 1-6 June 2008. IEEE Computational Intelligence Society, IEEE Press. (Cited on pages 49, 51, and 52.)

[20] L. Beadle and C. G. Johnson. Semantic analysis of program initialisation in genetic programming. *Genetic Programming and Evolvable Machines*, 10(3):307–337, Sept. 2009. ISSN 1389-2576. (Cited on page 49.)

[21] H. Bersini and V. Detours. Asynchrony induces stability in cellular automata based models. In *Proceedings of Artificial Life IV*, pages 382–387. MIT Press, Cambridge, 1994. (Cited on page 133.)

[22] H.-G. Beyer, T. Jansen, C. Reeves, and M. Vose, editors. *Theory of Evolutionary Algorithms, 15.-20. February 2004*, volume 04081 of *Dagstuhl Seminar Proceedings*, 2006. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany. (Cited on page 10.)

[23] G. Birkhoff. *Lattice theory*. American Mathematical Society, 1967. (Cited on pages 12, 19, and 35.)

[24] A. Z. Broder and A. R. Karlin. Bounds on the cover time. *J. Theoretical Probab*, 2:101–120, 1988. (Cited on page 161.)

[25] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, 35:677–691, August 1986. ISSN 0018-9340. (Cited on pages 49 and 51.)

[26] E. Burke, S. Gustafson, and G. Kendall. Diversity in genetic programming: An analysis of measures and correlation with fitness. *IEEE Transactions on Evolutionary Computation*, 8(1):47–62, 2004. (Cited on page 33.)

[27] R. Buvel and T. Ingerson. Structure in asynchronous cellular automata. *Physica*, D 1:59–68, 1984. (Cited on page 133.)

[28] F. Cantelli. Sulla probabilità come limite della frequenza. *Rend. Accad. dei Lincei*, 24:39–45, 1917. (Cited on page 165.)

[29] R. Cappuccio, G. Cattaneo, G. Erbacci, and U. Jocher. A parallel implementation of a cellular automata based model for coffee percolation. *Parallel Computing*, 27(5):685–717, 2001. (Cited on page 1.)

[30] G. Cattaneo and D. Ciucci. Lattice with interior and closure operators and abstract approximation spaces. In J. P. et al., editor, *Foundations of Rough Sets*, LNCS – Transactions on Rough Sets X, pages 67–116. Springer, 2009. (Cited on page 12.)

[31] G. Cattaneo, A. Dennunzio, and L. Margara. Chaotic subshifts and related languages applications to one-dimensional cellular automata. *Fundamenta Informaticae*, 52:39–80, 2002. (Cited on page 133.)

[32] G. Cattaneo, A. Dennunzio, and L. Margara. Solution of some conjectures about topological properties of linear cellular automata. *Theor. Comput. Sci.*, 325(2):249–271, 2004. (Cited on page 133.)

[33] M. Caudill. Neural networks primer, part i. *AI Expert*, 2:46–52, December 1987. ISSN 0888-3785. (Cited on page 121.)

[34] P. Chaudhuri, D. Chowdhury, S. Nandi, and S. Chattopadhyay. *Additive Cellular Automata Theory and Applications*, volume 1. IEEE Press, New York, 1997. (Cited on page 133.)

[35] B. Chopard. Modelling physical systems by cellular automata. In G. R. et al., editor, *Handbook of Natural Computing: Theory, Experiments, and Applications*. Springer, 2011. To appear. (Cited on page 133.)

[36] P. M. Cohn. *Universal Algebra*. Harper and Row, 1965. (Cited on page 19.)

[37] J. Conway and S. Norton. Monstrous moonshine. *Bull. London Math. Soc*, 11(3):308–339, 1979. (Cited on page 2.)

[38] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines: and other kernel-based learning methods*. Cambridge University Press, 2000. (Cited on page 121.)

[39] R. Curry, P. Lichodzijewski, and M. I. Heywood. Scaling genetic programming to large datasets using hierarchical dynamic subset selection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(4): 1065–1073, 2007. (Cited on page 49.)

[40] A. Czarn, C. MacNish, K. Vijayan, and B. Turlach. Statistical exploratory analysis of genetic algorithms: the importance of interaction. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 2288 – 2295 Vol.2, june 2004. (Cited on page 106.)

[41] J. M. Daida, R. Bertram, S. Stanhope, J. Khoo, S. Chaudhary, and O. Chaudhary. What makes a problem GP-hard? analysis of a tunably difficult problem in genetic programming. *Genetic Programming and Evolvable Machines*, 2:165–191, 2001. (Cited on page 59.)

[42] C. Darwin. *The Origin of Species*. John Murray, 1859. (Cited on page 2.)

[43] T. Davis and J. Principe. A markov chain framework for the simple genetic algorithm. *Evolutionary computation*, 1(3):269–288, 1993. (Cited on page 9.)

[44] E. B. de Lima, G. Pappa, J. M. de Almeida, M. Goncalves, and W. Meira. Tuning genetic programming parameters with factorial designs. In *IEEE Congress on Evolutionary Computation (CEC 2010)*, Barcelona, Spain, 18-23 July 2010. IEEE Press. (Cited on pages 105 and 108.)

[45] K. Deb and D. E. Goldberg. Analyzing deception in trap functions. In D. Whitley, editor, *FOGA-2*, pages 93–108. Morgan Kaufmann, 1993. (Cited on page 59.)

[46] A. Dennunzio and E. Formenti. Decidable properties of 2d cellular automata. In *Developments in Language Theory*, volume 5257 of *LNCS*, pages 264–275. Springer, 2008. (Cited on page 133.)

[47] A. Dennunzio, P. Di Lena, E. Formenti, and L. Margara. On the directional dynamics of additive cellular automata. *Theoretical Computer Science*, 410:4823–4833, 2009. (Cited on page 133.)

[48] A. Dennunzio, B. Masson, and P. Guillon. Sand automata as cellular automata. *Theoretical Computer Science*, 410:3962–3974, 2009. (Cited on page 133.)

[49] A. Dennunzio, E. Formenti, and P. Kůrka. Cellular automata dynamical systems. In G. R. et al., editor, *Handbook of Natural Computing: Theory, Experiments, and Applications*. Springer, 2010. To appear. (Cited on page 133.)

[50] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. New York: Wiley, 1973. (Cited on page 66.)

[51] A. Ehrenfeucht and G. Rozenberg. Basic notions of reaction systems. In *Developments in Language Theory 8th International Conference, DLT 2004*, volume 3340 of *Lecture Notes in Computer Science*, pages 27–29, Auckland, New Zealand, 2004. Springer. (Cited on pages 1, 2, 101, and 104.)

[52] A. Ehrenfeucht and G. Rozenberg. Reaction systems. *Fundamenta Informaticae*, 75:263–280, 2007. (Cited on pages 1, 2, 103, and 104.)

[53] A. Ehrenfeucht and G. Rozenberg. Introducing time in reaction systems. *Theoretical Computer Science*, 410:310–322, 2009. (Cited on page 101.)

[54] A. Ehrenfeucht, M. Main, and G. Rozenberg. Combinatorics of life and death for reaction systems. *International Journal of Foundations of Computer Science*, 21:345–356, 2010. (Cited on page 111.)

[55] A. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 3(2):124 –141, jul 1999. (Cited on pages 105 and 108.)

[56] A. Eibena and G. Rudolphb. Theory of evolutionary algorithms: A birds eye view. *Theoretical Computer Science*, 229:3–9, 1999. (Cited on page 7.)

[57] M. Eigen. Selforganization of matter and the evolution of biological macromolecules. *Naturwissenschaften*, 58:465–523, 1971. ISSN 0028-1042. (Cited on page 57.)

[58] M. Eigen, J. Mccaskill, and P. Schuster. The molecular quasi-species. *Adv. Chem. Phys.*, 75:149–263, 1989. (Cited on page 57.)

[59] A. Ekárt and S. Z. Németh. Maintaining the diversity of genetic programs. In J. A. Foster, E. Lutton, J. Miller, C. Ryan, and A. G. B. Tettamanzi, editors, *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, volume 2278 of *LNCS*, pages 162–171, Kinsale, Ireland, 3-5 Apr. 2002. Springer, Berlin, Heidelberg, New York. ISBN 3-540-43378-3. (Cited on page 33.)

[60] F. Fagnani and L. Margara. Expansivity, permutivity, and chaos for cellular automata. *Theory Comput. Syst.*, 31(6):663–677, 1998. (Cited on page 167.)

[61] F. Farina and A. Dennunzio. A predator-prey cellular automaton with parasitic interactions and environmental effects. *Fundamenta Informaticae*, 83:337–353, 2008. (Cited on page 133.)

[62] N. Fatès and M. Morvan. An experimental study of robustness to asynchronism for elementary cellular automata. *Complex Systems*, 16(1):1–27, 2005. (Cited on page 133.)

[63] N. Fatès, M. Morvan, N. Schabanel, and E. Thierry. Fully asynchronous behaviour of double-quiescent elementary cellular automata. *Theoretical Computer Science*, 362:1–16, 2006. (Cited on page 133.)

[64] N. Fatès, D. Regnault, N. Schabanel, and E. Thierry. Asynchronous behaviour of double-quiescent elementary cellular automata. In *Proceedings of LATIN'2006*, volume 3887 of *LNCS*, pages 455–466. Springer, 2006. (Cited on page 133.)

[65] R. Feldt and P. Nordin. Using factorial experiments to evaluate the effect of genetic programming parameters. In *Genetic Programming, Proceedings of EuroGP 2000, volume 1802 of LNCS*, pages 271–282. Springer-Verlag, 2000. (Cited on page 108.)

[66] D. Fogel. Evolving computer programs. In D. Fogel, editor, *Evolutionary Computation: The Fossil Record*, chapter 5, pages 143–144. MIT Press, 1998. (Cited on page 104.)

[67] D. Fogel and A. Ghozeil. The schema theorem and the misallocation of trials in the presence of stochastic effects. In *Evolutionary Programming VII*, pages 313–321. Springer, 1998. (Cited on page 9.)

[68] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence Through Simulated Evolution*. John Wiley and Sons, New York, 1966. (Cited on page 104.)

[69] W. Fontana and P. Schuster. A computer model of evolutionary optimization. *Biophysical Chemistry*, 26(2-3):123 – 147, 1987. (Cited on page 57.)

[70] W. Fontana, W. Schnabl, and P. Schuster. Physical aspects of evolutionary optimization and adaptation. *Phys. Rev. A*, 40:3301–3321, Sep 1989. (Cited on page 57.)

[71] W. Fontana, P. F. Stadler, E. G. Bornberg-Bauer, T. Griesmacher, I. L. Hofacker, M. Tacker, P. Tarazona, E. D. Weinberger, and P. Schuster. Rna folding and combinatory landscapes. *Physical Review E*, 47(3): 2083–2099, Mar 1993. (Cited on page 57.)

[72] F. D. Francone, P. Nordin, and W. Banzhaf. Benchmarking the generalization capabilities of a compiling genetic programming system using sparse data sets. In J. R. K. *et al.*, editor, *Genetic Programming: Proceedings of the first annual conference*, pages 72–80. MIT Press, Cambridge, 1996. (Cited on page 53.)

[73] A. Frank and A. Asuncion. UCI machine learning repository, 2010. http://www.ics.uci.edu/~mlearn/. (Cited on pages 58 and 121.)

[74] M. Fréchet. Sur la notion de voisinage dans les ensembles abstraits. *Comptes rendus de l'Acadmie des Sciences*, 165:359–360, 1917. (Cited on pages 15 and 16.)

[75] R. Friedberg. A learning machine: Part 1. *IBM J. Research and Development*, Vol. 2:1:2–13, 1958. (Cited on page 104.)

[76] H. Fukŝ. Non-deterministic density classification with diffusive probabilistic cellular automata. *Physical Review*, E 66(2), 2002. (Cited on page 133.)

[77] C. Gagné, M. Schoenauer, M. Parizeau, and M. Tomassini. Genetic programming, validation sets, and parsimony pressure. In P. C. *et al.*, editor, *Genetic Programming, 9th European Conference, EuroGP2006*, Lecture Notes in Computer Science, pages 109–120. Springer, Berlin, Heidelberg, New York, 2006. ISBN 3-540-33143-3. (Cited on page 54.)

[78] G.Cattaneo, A. Dennunzio, E. Formenti, and J. Provillard. Non-uniform cellular automata. In *LATA*, volume 5457 of *LNCS*, pages 302–313. Springer, 2009. ISBN 978-3-642-00981-5. (Cited on page 133.)

[79] N. Geard, J. Wiles, J. Hallinan, B. Tonkes, and B. Skellett. A comparison of neutral landscapes - NK, NKp and NKq. *Proceedings of the World Congress on Computational Intelligence*, 1:205–210, 2002. (Cited on page 57.)

[80] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989. (Cited on pages 8, 10, 33, and 55.)

[81] F. Gomez, J. Togelius, and J. Schmidhuber. Measuring and optimizing behavioral complexity for evolutionary reinforcement learning. In *ICANN '09*, pages 765–774. Springer, 2009. (Cited on page 54.)

[82] R. L. Graham, B. L. Rothschild, and J. H. Spencer. *Ramsey Theory*. Wiley-Interscience Series in Discrete Mathematics and Optimization Advisory. Wiley-Interscience, 1990. (Cited on page 110.)

[83] G. Grimmett and D. Stirzaker. *Probability and random processes*. Oxford university press, 2001. (Cited on page 9.)

[84] H. Guo, L. B. Jack, and A. K. Nandi. Feature generation using genetic programming with application to fault classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35(1):89–99, 2005. (Cited on page 97.)

[85] S. Gustafson, A. Ekárt, E. Burke, and G. Kendall. Problem difficulty and code growth in genetic programming. *Genetic Programming and Evolvable Hardware*, 5(3):271–290, 2004. (Cited on page 33.)

[86] S. Gustafson, E. Burke, and N. Krasnogor. The tree-string problem: An artificial domain for structure and content search. In M. Keijzer, A. Tettamanzi, P. Collet, J. van Hemert, and M. Tomassini, editors, *EuroGP 2005*, pages 215–226. Springer, 2005. (Cited on page 59.)

[87] B. H, J. M, and Z. M. Developing new fitness functions in genetic programming for classification with unbalanced data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, pp(99):1–16, 2011. (Cited on page 49.)

[88] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11:10–18, November 2009. ISSN 1931-0145. `http://www.cs.waikato.ac.nz/ml/weka/`. (Cited on pages 96, 120, and 121.)

[89] P. Halmos. *Measure theory*, volume 38 of *Graduate texts in Mathematics*. Springer-Verlag, 1974. (Cited on page 164.)

[90] D. Heckerman. A tutorial on learning with bayesian networks. In *Innovations in Bayesian Networks*, volume 156 of *Studies in Computational Intelligence*, pages 33–82. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-85065-6. (Cited on page 121.)

[91] G. Hedlund. Endomorphisms and automorphisms of the shift dynamical system. *Mathematical Systems Theory*, 3:320–375, 1969. (Cited on page 166.)

[92] J. Hesser and R. Männer. Towards an optimal mutation probability for genetic algorithms. In *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, PPSN I, pages 23–32, London, UK, 1991. Springer-Verlag. ISBN 3-540-54148-9. (Cited on page 108.)

[93] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, Michigan, 1975. (Cited on pages 1, 10, 33, and 55.)

[94] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979. ISBN 0-201-02988-X. (Cited on page 137.)

[95] D. Jackson. Phenotypic diversity in initial genetic programming populations. In A. I. Esparcia-Alcazar, A. Ekart, S. Silva, S. Dignum, and A. S. Uyar, editors, *Proceedings of the 13th European Conference on Genetic Programming, EuroGP 2010*, volume 6021 of *LNCS*, pages 98–109, Istanbul, 7-9 Apr. 2010. Springer. (Cited on page 49.)

[96] D. Jackson. Promoting phenotypic diversity in genetic programming. In R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, editors, *PPSN 2010 11th International Conference on Parallel Problem Solving From Nature*, volume 6239 of *Lecture Notes in Computer Science*, pages 472–481, Krakow, Poland, 11-15 Sept. 2010. Springer. (Cited on page 49.)

[97] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley/Interscience, New York, NY, USA, 1 edition, Apr. 1991. ISBN 978-0-471-50336-1. (Cited on page 108.)

[98] T. Jansen and I. Wegener. On the utility of populations in evolutionary algorithms. In *GECCO 2001: Proceedings of the 3rd annual conference on Genetic and evolutionary computation*, pages 1034–1041. ACM, 2001. (Cited on page 10.)

[99] T. Jech. *Set Theory*. Springer Verlang, Berlin, 2006. (Cited on page 32.)

[100] T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, University of New Mexico, 1995. (Cited on page 57.)

[101] T. Jones. One operator, one landscape. Working Papers 95-02-025, Santa Fe Institute, Feb. 1995. (Cited on page 57.)

[102] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 184–192, San Francisco, CA, 1995. Morgan Kaufmann. (Cited on pages 33 and 57.)

[103] S. Jukna. *Extremal combinatorics: with applications in computer science*. Springer, 2001. (Cited on page 110.)

[104] W. Kantschik and W. Banzhaf. Linear-tree GP and its comparison with other GP structures. In *Genetic Programming, Proceedings of EuroGP'2001*, volume 2038 of *LNCS*, pages 302–312, Lake Como, Italy, 18-20 Apr. 2001. Springer-Verlag. (Cited on page 104.)

[105] W. Kantschik and W. Banzhaf. Linear-graph GP—A new GP structure. In *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, volume 2278 of *LNCS*, pages 83–92, Kinsale, Ireland, 3-5 Apr. 2002. Springer-Verlag. (Cited on page 104.)

[106] S. Kauffman and S. Levin. Towards a general theory of adaptive walks on rugged landscapes. *J. Theoret. Biol.*, 128(1):11–45, 1987. ISSN 0022-5193. (Cited on pages 55 and 56.)

[107] S. A. Kauffman. *The Origins of Order*. Oxford University Press, New York, 1993. (Cited on pages 55, 56, and 57.)

[108] J. L. Kelley. *General topology*. Springer-Verlag, 1955. (Cited on pages 12 and 136.)

[109] T. Kennedy. Managing the drug discovery/development interface. *Drug Discovery Today*, 2(10):436–444, 1997. (Cited on page 94.)

[110] J. Kingman. *Mathematics of genetic diversity*. Number 34 in CBMS-NSF regional conference series in applied mathematics. Society for Industrial and Applied Mathematics, Philadelphia, Pa., 2. druck edition, 1980. ISBN 0898711665. (Cited on page 56.)

[111] J. F. C. Kingman. *Poisson Processes*. Oxford University Press, 1993. (Cited on page 160.)

[112] I. Kola and J. Landis. Can the pharmaceutical industry reduce attrition rates? *Nat Rev Drug Discov*, 3(8):711–716, 2004. (Cited on page 94.)

[113] J. Koza. A hierarchical approach to learning the boolean multiplexer function. In G. J. E. Rawlins, editor, *Foundations of genetic algorithms*, pages 171–192. Morgan Kaufmann, Indiana University, 15-18 July 1990 1991. (Cited on page 126.)

[114] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992. ISBN 0-262-11170-5. (Cited on pages 48, 55, 58, 63, 64, 78, 104, 105, 120, and 126.)

[115] J. R. Koza, D. Andre, F. H. Bennett, and M. A. Keane. *Genetic Programming III: Darwinian Invention & Problem Solving*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 1999. ISBN 1558605436. (Cited on page 49.)

[116] K. Krawiec. Medial crossovers for genetic programming. In A. Moraglio, S. Silva, K. Krawiec, P. Machado, and C. Cotta, editors, *Proceedings of the 15th European Conference on Genetic Programming, EuroGP 2012*, volume 7244 of *LNCS*, pages 61–72, Malaga, Spain, 11-13 Apr. 2012. Springer Verlag. (Cited on page 52.)

[117] K. Krawiec and P. Lichocki. Approximating geometric crossover in semantic space. In G. Raidl, F. Rothlauf, G. Squillero, R. Drechsler, T. Stuetzle, M. Birattari, C. B. Congdon, M. Middendorf, C. Blum, C. Cotta, P. Bosman, J. Grahl, J. Knowles, D. Corne, H.-G. Beyer, K. Stanley, J. F. Miller, J. van Hemert, T. Lenaerts, M. Ebner, J. Bacardit, M. O'Neill, M. Di Penta, B. Doerr, T. Jansen, R. Poli, and E. Alba, editors, *GECCO '09: Proceedings of the 11th Annual conference on Genetic*

and evolutionary computation, pages 987–994, Montreal, 8-12 July 2009. ACM. (Cited on pages 49 and 52.)

[118] K. Kuratowski. *Introduction to Set Theory and Topology*. Pergamon Press, 1961. (Cited on page 136.)

[119] P. Kůrka. Languages, equicontinuity and attractors in cellular automata. *Ergodic Theory & Dynamical Systems*, 17:417–433, 1997. (Cited on pages 148 and 167.)

[120] P. Kůrka. Topological dynamics of one-dimensional cellular automata. In B. Meyers, editor, *Mathematical basis of cellular automata*, Encyclopedia of Complexity and System Science, pages 2232–2242. Springer Verlag, 2009. (Cited on pages 2, 5, and 133.)

[121] I. Kushchu. An evaluation of evolutionary generalization in genetic programming. *Artificial Intelligence Review*, 18(1):3–14, 2002. (Cited on page 53.)

[122] W. B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer, Berlin, 2002. (Cited on pages 50 and 53.)

[123] J. Lee, S. Adachi, F. Peper, and S. Mashiko. Delay-insensitive computation in asynchronous cellular automata. *J. Comput. Syst. Sci.*, 70: 201–220, 2005. (Cited on pages 134 and 149.)

[124] V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966. (Cited on page 140.)

[125] Y. Lin and B. Bhanu. Evolutionary feature synthesis for object recognition. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 35(2):156–171, May 2005. ISSN 1094-6977. (Cited on page 97.)

[126] L. Manzoni. Asynchronous cellular automata and dynamical properties. *Natural Computing*, 11(2):269–276, 2012. (Cited on pages 135 and 165.)

[127] A. Maruoka and M. Kimura. Injectivity and surjectivity for parallel maps for CA. *J. Comp. and Sys. Sci*, 18:47–64, 1979. (Cited on page 139.)

[128] J. McDermott, U. O'Reilly, L. Vanneschi, and K. Veeramachaneni. How far is it from here to there? A distance that is coherent with GP operators. In S. Silva, J. A. Foster, M. Nicolau, M. Giacobini, and P. Machado, editors, *Proceedings of the 14th European Conference on Genetic Programming, EuroGP 2011*, volume 6621 of *LNCS*, pages 191–202, Turin, Italy, 27-29 Apr. 2011. Springer Verlag. (Cited on pages 33 and 34.)

[129] J. McDermott, D. R. White, S. Luke, L. Manzoni, M. Castelli, L. Vanneschi, W. Jaśkowski, K. Krawiec, R. Harper, K. D. Jong, and U.-M. O'Reilly. Genetic programming needs better benchmarks. In *Genetic and Evolutionary Computation Conference, GECCO 2012*, pages 791–798, Philadelphia, USA, July 2012. ACM. (Cited on page 173.)

[130] N. F. McPhee, B. Ohs, and T. Hutchison. Semantic building blocks in genetic programming. Working Paper Series Volume 3 Number 2, University of Minnesota Morris, 600 East 4th Street, Morris, MN 56267, USA, 12 Dec. 2007. (Cited on page 51.)

[131] N. F. McPhee, B. Ohs, and T. Hutchison. Semantic building blocks in genetic programming. In *Proceedings of the 11th European conference on Genetic programming*, EuroGP'08, pages 134–145, Berlin, Heidelberg, 2008. Springer-Verlag. (Cited on pages 49 and 50.)

[132] J. Miller and P. Thomson. Cartesian genetic programming. In *Genetic Programming, Proceedings of EuroGP'2000*, volume 1802 of *LNCS*, pages 121–132, Edinburgh, 15-16 Apr. 2000. Springer-Verlag. (Cited on page 104.)

[133] M. Mitchell, S. Forrest, and J. Holland. The royal road for genetic algorithms: fitness landscapes and ga performance. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems, Proc. of the First European Conf. on Artif. Life*, pages 245–254. The MIT Press, 1992. (Cited on page 59.)

[134] T. Mitchell. *Machine learning*. New York: McGraw-Hill, 1997. (Cited on page 1.)

[135] D. Montana. Strongly typed genetic programming. *Evolutionary Computation*, 3(2):199–230, 1995. (Cited on page 104.)

[136] A. Monteiro and H. Ribeiro. Sur l'axiomatique des espaces (V). *Portugaliae Mathematica*, 1:275–288, 1937. (Cited on page 16.)

[137] A. Monteiro, H. Ribeiro, J. Paulo, and M. Z. Nunes. Les ensembles fermés et les fondaments de la topologie. *Portugaliae Mathematica*, 2: 56–66, 1941. (Cited on page 19.)

[138] E. H. Moore. *Introduction to a form of general analysis*, volume 2 of *AMS Colloq. Publ.* American Mathematical Society, Providence, Rhode Island, 1910. (Cited on page 19.)

[139] A. Moraglio. One-point geometric crossover. In R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, editors, *PPSN (1)*, volume 6238 of *Lecture Notes in Computer Science*, pages 83–93. Springer, 2010. ISBN 978-3-642-15843-8. (Cited on pages 3 and 11.)

[140] A. Moraglio. Geometry of evolutionary algorithms. In N. Krasnogor and P. Lanzi, editors, *GECCO (Companion)*, pages 1439–1468. ACM, 2011. ISBN 978-1-4503-0690-4. (Cited on pages 3 and 11.)

[141] A. Moraglio and R. Poli. Topological interpretation of crossover. In *In Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1377–1388. Springer, 2004. (Cited on pages 3, 11, 52, and 87.)

[142] A. Moraglio, K. Krawiec, and C. Johnson. Geometric semantic genetic programming. In *Parallel Problem Solving from Nature (PPSN)*, pages 21–31. Springer, 2012. (Cited on pages 4, 49, 50, 52, 53, 86, 87, 88, 90, and 91.)

[143] D. P. Muni, N. R. Pal, and J. Das. Genetic programming for simultaneous feature selection and classifier design. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 36(1):106–117, 2006. (Cited on page 97.)

[144] J. Munkres. Topology: a first course, 1975. (Cited on page 136.)

[145] K. Nakamura. Asynchronous cellular automata and their computational ability. *Systems, Computers, Control*, 5:58–66, 1974. (Cited on pages 133, 134, and 149.)

[146] V. Nannen and A. Eiben. Efficient relevance estimation and value calibration of evolutionary algorithm parameters. In *IEEE Congress on Evolutionary Computation*, pages 103–110. IEEE, 2007. (Cited on page 106.)

[147] V. Nannen, S. Smit, and A. Eiben. Costs and benefits of tuning parameters of evolutionary algorithms. In *Proceedings of the 10th international conference on Parallel Problem Solving from Nature: PPSN X*, pages 528–538, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-87699-1. (Cited on pages 105 and 107.)

[148] C. L. Nehaniv. Evolution in asynchronous cellular automata. *Artificial Life VIII*, pages 65–73, 2002. (Cited on pages 134 and 149.)

[149] J. Neumann and A. Burks. *Theory of self-reproducing automata*. University of Illinois Press, 1966. (Cited on page 2.)

[150] Q. U. Nguyen, X. H. Nguyen, and M. O'Neill. Semantic aware crossover for genetic programming: The case for real-valued function regression. In *Proceedings of the 12th European Conference on Genetic Programming*, EuroGP '09, pages 292–302, Berlin, Heidelberg, 2009. Springer-Verlag. (Cited on page 51.)

[151] A. Nix and M. Vose. Modeling genetic algorithms with markov chains. *Annals of mathematics and artificial intelligence*, 5(1):79–88, 1992. (Cited on page 9.)

[152] G. Ochoa, M. Tomassini, S. Vèrel, and C. Darabos. A study of NK landscapes' basins and local optima networks. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 555–562. ACM, 2008. (Cited on page 57.)

[153] P. Oliveto, J. He, and X. Yao. Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. *International Journal of Automation and Computing*, 4(3):281–293, 2007. (Cited on pages 7 and 10.)

[154] N. Ollinger. Universalities in cellular automata. In G. R. et al., editor, *Handbook of Natural Computing: Theory, Experiments, and Applications*. Springer, 2011. To appear. (Cited on pages 2, 134, and 149.)

[155] M. O'Neill and C. Ryan. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358, Aug. 2001. (Cited on page 104.)

[156] M. O'Neill, L. Vanneschi, S. Gustafson, and W. Banzhaf. Open issues in genetic programming. *Genetic Programming and Evolvable Machines*, 11:339–363, 2010. ISSN 1389-2576. (Cited on pages 53, 55, 58, 85, 104, and 105.)

[157] A. Orfila, J. M. Estevez-Tapiador, and A. Ribagorda. Evolving high-speed, easy-to-understand network intrusion detection rules with genetic programming. In M. Giacobini, I. De Falco, and M. Ebner, editors, *App. of Evolutionary Computing, EvoWorkshops2009*, LNCS. Springer Verlag, 2009. (Cited on page 58.)

[158] M. Orr. Introduction to radial basis function networks. Technical report, Centre For Cognitive Science, University of Edinburgh, Edinburgh, Scotland, 1996. (Cited on page 121.)

[159] C. Papadimitriou. *Computational complexity*. John Wiley and Sons Ltd., 2003. (Cited on page 1.)

[160] J. Platt. A fast algorithm for training support vector machines. Technical report, Microsoft Research, Redmond, USA, 1998. (Cited on page 121.)

[161] R. Poli. Exact schema theory for genetic programming and variable-length genetic algorithms with one-point crossover. *Genetic Programming and Evolvable Machines*, 2(2):123–163, 2001. (Cited on page 9.)

[162] R. Poli, W. B. Langdon, and N. F. McPhee. *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008. (With contributions by J. R. Koza). (Cited on pages 1, 3, 48, 53, 55, 95, 104, and 121.)

[163] B. Punch, D. Zongker, and E. Goodman. The royal tree problem, a benchmark for single and multiple population genetic programming. In P. Angeline and K. Kinnear, editors, *Advances in Genetic Programming 2*, pages 299–316, Cambridge, MA, 1996. The MIT Press. (Cited on page 59.)

[164] U. N. Quang, X. H. Nguyen, and M. O'Neill. Semantics based mutation in genetic programming: The case for real-valued symbolic regression. In R. Matousek and L. Nolle, editors, *15th International Conference on Soft Computing, Mendel'09*, pages 73–91, Brno, Czech Republic, June 24-26 2009. (Cited on pages 51 and 52.)

[165] C. Reeves and J. Rowe. *Genetic algorithms: principles and perspectives : a guide to GA theory*. Springer, 2002. (Cited on pages 7, 10, and 57.)

[166] D. Regnault. Abrupt behaviour changes in cellular automata under asynchronous dynamics. In *Electronic proc. of 2nd European Conference on Complex Systems*, ECCS. Oxford, UK, 2006. (Cited on page 133.)

[167] D. Regnault, N. Schabanel, and E. Thierry. Progresses in the analysis of stochastic 2d cellular automata: A study of asynchronous 2d minority. *Theoretical Computer Science*, 410:4844–4855, 2009. (Cited on page 133.)

[168] H. Ribeiro. Sur l'axiomatique des espaces topologiques de m. fréchet. *Portugaliae Mathematica*, 1:260–274, 1937. (Cited on page 15.)

[169] I. Rish. An empirical study of the naive bayes classifier. In *IJCAI-01 workshop on "Empirical Methods in AI"*, 2001. (Cited on page 121.)

[170] J. Rissanen. Modeling by shortest data description. *Automatica*, 14: 465–471, 1978. (Cited on page 53.)

[171] J. Rosca. Generality versus size in genetic programming. In e. a. J.R. Koza, editor, *GP 1996*, pages 381–387. MIT Press, 1996. (Cited on page 53.)

[172] J. Rowe, M. Vose, and A. Wright. Neighborhood graphs and symmetric genetic operators. In *FOGA*, pages 110–122, 2007. (Cited on page 10.)

[173] J. Rowe, M. Vose, and A. Wright. Representation invariant genetic operators. *Evolutionary Computation*, 18(4):635–660, 2010. (Cited on page 10.)

[174] G. Rudolph. Convergence analysis of canonical genetic algorithms. *Neural Networks, IEEE Transactions on*, 5(1):96–101, 1994. (Cited on page 9.)

[175] G. Rudolph. *Convergence properties of evolutionary algorithms*. Kovac, 1997. (Cited on page 2.)

[176] B. Schönfisch and A. de Roos. Synchronous and asynchronous updating in cellular automata. *BioSystems*, 51:123–143, 1999. (Cited on page 133.)

[177] W. Sierpiǹski. *General Topology*. University of Toronto Press, 1952. (Cited on page 16.)

[178] S. Silva and E. Costa. Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. *Genetic Programming and Evolvable Machines*, 10(2):141–179, 2009. ISSN 1389-2576. (Cited on page 53.)

[179] S. Silva and L. Vanneschi. Operator equalisation, bloat and overfitting: a study on human oral bioavailability prediction. In F. Rothlauf, editor, *GECCO*, pages 1115–1122. ACM, 2009. ISBN 978-1-60558-325-9. (Cited on page 53.)

[180] B. Skellett, B. Cairns, N. Geard, B. Tonkes, and J. Wiles. Maximally rugged NK landscapes contain the highest peaks. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 579–584, New York, NY, USA, 2005. ACM. ISBN 1-59593-010-8. (Cited on page 57.)

[181] S. Smit and A. Eiben. Comparing parameter tuning methods for evolutionary algorithms. In *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, pages 399 –406, may 2009. (Cited on page 108.)

[182] A. Smith. Cellular automata and formal languages. In *Switching and Automata Theory, 1970., IEEE Conference Record of 11th Annual Symposium on*, pages 216–224. IEEE, 1970. (Cited on page 2.)

[183] D. Song, M. I. Heywood, and A. N. Zincir-Heywood. A linear genetic programming approach to intrusion detection. In E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, and J. Miller, editors, *Genetic and Evolutionary Computation – GECCO-2003*, volume 2724 of *LNCS*, pages 2325–2336, Chicago, 12-16 July 2003. Springer-Verlag. ISBN 3-540-40603-4. (Cited on page 58.)

[184] T. Soule and J. Foster. Code size and depth flows in genetic programming. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 313–320, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann. (Cited on page 108.)

[185] B. Stadler, P. Stadler, M. Shpak, and G. Wagner. Recombination spaces, metrics, and pretopologies. *Z. Phys. Chem*, 216:2002, 2002. (Cited on page 10.)

[186] B. M. Stadler, P. F. Stadler, M. Shpak, and G. P. Wagner. Recombination spaces, metrics and pretopologies. *Zeitschrift für Physikalische Chemie*, 216:217–234, 2002. (Cited on pages 12, 24, and 26.)

[187] P. Stadler. Towards a theory of landscapes. In R. López-Peña, H. Waelbroeck, R. Capovilla, R. García-Pelayo, and F. Zertuche, editors, *Complex Systems and Binary Networks*, volume 461-461 of *Lecture Notes in Physics*, pages 78–163. Springer Berlin / Heidelberg, 1995. ISBN 978-3-540-60339-9. (Cited on page 57.)

[188] P. Stadler and G. Wagner. Algebraic theory of recombination spaces. *Evololutionary Computation*, 5(3):241–275, 1997. ISSN 1063-6560. (Cited on page 10.)

[189] W. A. Tackett. *Recombination, Selection, and the Genetic Construction of Computer Programs*. PhD thesis, University of Southern California, Department of Electrical Engineering Systems, USA, 1994. (Cited on page 59.)

[190] K. C. Tan, Q. Yu, and T. H. Lee. A distributed evolutionary classifier for knowledge discovery in data mining. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, pages 131–142, 2005. (Cited on page 49.)

[191] A. Teller and M. Veloso. PADO: A new learning architecture for object recognition. In K. Ikeuchi and M. Veloso, editors, *Symbolic Visual Learning*, pages 81–116. Oxford University Press, 1996. (Cited on page 104.)

[192] M. Tomassini, L. Vanneschi, F. Fernández, and G. Galeano. A study of diversity in multipopulation genetic programming. In *6th International Conference on Evolutionary Computation EA'03*, pages 69–81, 2003. (Cited on page 33.)

[193] M. Tomassini, L. Vanneschi, P. Collard, and M. Clergue. A study of fitness distance correlation as a difficulty measure in genetic programming. *Evolutionary Computation*, 13(2):213–239, Summer 2005. ISSN 1063-6560. (Cited on pages 33 and 59.)

[194] A. Tuffs. Bayer faces shake up after lipobay withdrawn. *BMJ British Medical Journal*, 323(7317):828, 10 2001. (Cited on page 94.)

[195] N. Q. Uy, N. X. Hoai, M. O'Neill, B. McKay, and E. Galvan-Lopez. An analysis of semantic aware crossover. In Z. Cai, Z. Li, Z. Kang, and Y. Liu, editors, *Proceedings of the International Symposium on Intelligent Computation and Applications*, volume 51 of *Communications in Computer and Information Science*, pages 56–65. Springer, 2009. (Cited on pages 49 and 50.)

[196] N. Q. Uy, N. X. Hoai, M. O'Neill, and B. McKay. The role of syntactic and semantic locality of crossover in genetic programming. In R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, editors, *PPSN 2010 11th International Conference on Parallel Problem Solving From Nature*, volume 6239 of *Lecture Notes in Computer Science*, pages 533–542, Krakow, Poland, 11-15 Sept. 2010. Springer. (Cited on page 51.)

[197] N. Q. Uy, N. X. Hoai, M. O'Neill, R. I. McKay, and E. Galvan-Lopez. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119, June 2011. ISSN 1389-2576. (Cited on pages 49 and 51.)

[198] N. Q. Uy, X. H. Nguyen, and M. O'Neill. Examining the landscape of semantic similarity based mutation. In N. Krasnogor, P. L. Lanzi, A. Engelbrecht, D. Pelta, C. Gershenson, G. Squillero, A. Freitas, M. Ritchie, M. Preuss, C. Gagne, Y. S. Ong, G. Raidl, M. Gallager, J. Lozano, C. Coello-Coello, D. L. Silva, N. Hansen, S. Meyer-Nieberg, J. Smith, G. Eiben, E. Bernado-Mansilla, W. Browne, L. Spector, T. Yu, J. Clune, G. Hornby, M.-L. Wong, P. Collet, S. Gustafson, J.-P. Watson, M. Sipper, S. Poulding, G. Ochoa, M. Schoenauer, C. Witt, and A. Auger, editors, *GECCO '11: Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1363–1370, Dublin, Ireland, 12-16 July 2011. ACM. (Cited on page 57.)

[199] L. Vanneschi. *Theory and Practice for Efficient Genetic Programming*. PhD thesis, Faculty of Sciences, University of Lausanne, Switzerland, 2004. (Cited on pages 2, 33, 57, 58, and 59.)

[200] L. Vanneschi and S. Silva. Using operator equalisation for prediction of drug toxicity with genetic programming. In L. S. Lopes, N. Lau, P. Mariano, and L. M. Rocha, editors, *EPIA*, volume 5816 of *Lecture Notes in Computer Science*, pages 65–76. Springer, 2009. ISBN 978-3-642-04685-8. (Cited on page 53.)

[201] L. Vanneschi and M. Tomassini. Pros and cons of fitness distance correlation in genetic programming. In A. M. Barry, editor, *GECCO 2003: Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference*, pages 284–287, Chigaco, 11 July 2003. AAAI. (Cited on page 57.)

[202] L. Vanneschi, S. Verel, M. Tomassini, and P. Collard. NK landscapes difficulty and negative slope coefficient: How sampling influences the

results. In *Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing*, EvoWorkshops '09, pages 645–654, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-01128-3. (Cited on page 57.)

[203] L. Vanneschi, M. Castelli, and S. Silva. Measuring bloat, overfitting and functional complexity in genetic programming. In *GECCO '10*, pages 877–884. ACM, 2010. (Cited on pages 53, 54, 60, 61, 64, 65, and 70.)

[204] V. K. Vassilev, T. C. Fogarty, and J. F. Miller. Information characteristics and the structure of landscapes. *Evol. Comput.*, 8(1):31–60, Mar. 2000. ISSN 1063-6560. (Cited on page 57.)

[205] V. Vazirani. *Approximation algorithms*. Springer Verlag, 2001. (Cited on page 1.)

[206] E. Čech. *Topological Spaces*. Wiley, London, 1966. (Cited on page 24.)

[207] S. Vérel, P. Collard, and M. Clergue. Where are bottleneck in NK fitness landscapes ? In *CEC 2003: IEEE International Congress on Evolutionary Computation. Canberra, Australia*, pages 273–280. IEEE Press, Piscataway, NJ, 2003. (Cited on page 57.)

[208] S. Verel, G. Ochoa, and M. Tomassini. Local Optima Networks of NK Landscapes with Neutrality. *IEEE Transactions on Evolutionary Computation*, volume 14(6):to appear, 2010. (Cited on page 57.)

[209] E. J. Vladislavleva, G. F. Smits, and D. den Hertog. 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, Apr. 2009. ISSN 1089-778X. (Cited on page 54.)

[210] J. Von Neumann. The general and logical theory of automata. *Cerebral mechanisms in behavior*, pages 1–41, 1951. (Cited on page 2.)

[211] M. Vose. *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, Cambridge, MA, USA, 1998. ISBN 026222058X. (Cited on pages 2, 9, and 10.)

[212] M. Vose. Course notes: genetic algorithm theory. In M. Pelikan and J. Branke, editors, *GECCO (Companion)*, pages 2647–2660. ACM, 2010. ISBN 978-1-4503-0073-5. (Cited on page 10.)

[213] G. Wagner and P. Stadler. Complex adaptations and the structure of recombination spaces. In *School of Mathematics, UEA, Norwich NR4 7TJ*, 1997. (Cited on page 10.)

[214] E. D. Weinberg. Local properties of kauffman's n-k model, a tuneably rugged energy landscape. *Physical Review A*, 44(10):6399–6413, 1991. (Cited on page 57.)

[215] E. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63:325–336, 1990. ISSN 0340-1200. (Cited on page 57.)

[216] P. A. Whigham. *Grammatical Bias for Evolutionary Learning*. PhD thesis, School of Computer Science, University College, University of New South Wales, Australian Defence Force Academy, Canberra, Australia, 14 October 1996. (Cited on page 104.)

[217] D. S. Wishart, C. Knox, A. C. Guo, S. Shrivastava, M. Hassanali, P. Stothard, Z. Chang, and J. Woolsey. Drugbank: a comprehensive resource for in silico drug discovery and exploration. *Nucleic Acids Res*, 34:668–672, 2006. (Cited on page 94.)

[218] T. Worsch. A note on (intrinsically?) universal asynchronous cellular automata. Preprint, 2010. (Cited on page 134.)

[219] S. Wright. The roles of mutation, inbreeding, crossbreeding, and selection in evolution. *Proceedings of the Sixth International Congress on Genetics*, 1932. (Cited on page 57.)

[220] S. Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. In D. F. Jones, editor, *Proceedings og the Sixth International Congress on Genetics*, volume 1, pages 356–366, 1932. (Cited on page 54.)

[221] F. Yoshida and J. G. Topliss. Qsar model for drug human oral bioavailability1. *Journal of Medicinal Chemistry*, 43(13):2575–2585, 2000. (Cited on page 94.)

Milano, Italia (Milan, Italy)
Lisboa, Portugal (Lisbon, Portugal)
Birmingham, UK
寝屋川市、日本 (Neyagawa, Japan)

October MMXII