EVOLUTIONARY SYNTHESIS OF ANALOG NETWORKS

THÈSE N^{O} 3199 (2005)

PRÉSENTÉE À LA FACULTÉ SCIENCES ET TECHNIQUES DE L'INGÉNIEUR

Institut d'ingénierie des systèmes

SECTION DE MICROTECHNIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Claudio MATTIUSSI

laurea in ingegneria elettronica, Università degli Studi di Trieste, Italie
et de nationalité italienne

acceptée sur proposition du jury:

Prof. D. Floreano, directeur de thèse
Prof. I. Harvey, rapporteur
Prof. A. Ijspeert, rapporteur
Dr B. McMullin, rapporteur

Lausanne, EPFL
2005

Doctoral Thesis

Evolutionary synthesis of analog networks

Claudio Mattiussi

Laboratory of Intelligent Systems
Institute of System Engineering
Ecole Polytechnique Fédérale de Lausanne (EPFL)

Laboratory of Intelligent Systems
Institute of System Engineering
Ecole Polytechnique Fédérale de Lausanne (EPFL)

Acknowledgments

There are so many people that I would like to thank.

First, my thanks go to my supervisor, Prof. Dario Floreano, for making this thesis possible by accepting me in his team and providing support, guidance and freedom in the friendly and motivating atmosphere of the laboratory.

Many thanks to Prof. Laurent Keller and Prof. Roland Siegwart for having accepted to be members of the thesis committee, and to Dr. Inman Harvey, Prof. Auke Jan Ijspeert and Dr. Barry McMullin for having accepted to act as members of the doctoral examination board and for their many valuable comments. I also thank Prof. Edoardo Charbon for his help and interest in my work.

I warmly thank Dominique Etienne and Marie-Jo Pellaud for making things happen in the laboratory and for providing help, encouragement, and friendship when it was most needed.

A great "Thanks!" all my present and past colleagues in the LIS and ASL teams, and especially to Antoine Beyeler, Jesper Blynel, Michael Bonani, Michele Bongiovanni, Gilles Caprari, Diego Federici, Yannick Fournier, Bjoern Jensen, Michel Lauria, Stéphane Magnenat, Davide Marocco, Francesco Mondada, Andres Perez-Uribe, Daniel Roggen, Mathieu Scherz, Mototaka Suzuki, Danesh Tarapore, Nicola Tomatis and Markus Waibel for their help and the many chats about all things, bio-inspired and not. With Jean-Christophe Zufferey I have also shared an office for three years, in addition to many good laughs, several scientific and philosophic ramblings, and a good deal of friendship.

Many thanks to my family and to all my friends for their support and encouragement, and especially to Luca, Giovanni, Maurizio, Valentina, Silvano, Sandra and Tristano for forcing me to explain what it was all about, to Clemente, Carlo Emilio, Eros, Gino, Larf, Lello, Leo, Jack, Ken, Milka, Molly, Molly, Mino, Pete, Poldo, Valentino and Vladdy for keeping asking me when the thesis would be finished, and to good ol' Angelo for cheering me up so many times with his contagious smile.

And, most importantly, loving thanks to Betty for her presence, support, enthusiasm and understanding.

Lausanne, March 2005

The work reported in this thesis was supported by the Swiss National Science Foundation, grant no. 620-58049. The thesis was prepared using LyX and LaTeX.

Abstract

The significant increase in the available computational power that took place in recent decades has been accompanied by a growing interest in the application of the evolutionary approach to the synthesis of many kinds of systems and, in particular, to the synthesis of systems like analog electronic circuits, neural networks, and, more generally, autonomous systems, for which no satisfying systematic and general design methodology has been found to date. Despite some interesting results in the evolutionary synthesis of these kinds of systems, the endowment of an artificial evolutionary process with the potential for an appreciable increase of complexity of the systems thus generated appears still an open issue.

In this thesis the problem of the evolutionary growth of complexity is addressed taking as starting point the insights contained in the published material reporting the unfinished work done in the late 1940s and early 1950s by John von Neumann on the theory of self-reproducing automata. The evolutionary complexity-growth conditions suggested in that work are complemented here with a series of auxiliary conditions inspired by what has been discovered since then relatively to the structure of biological systems, with a particular emphasis on the workings of genetic regulatory networks seen as the most elementary, full-fledged level of organization of existing living organisms.

In this perspective, the first chapter is devoted to the formulation of the problem of the evolutionary growth of complexity, going from the description of von Neumann's complexity-growth conditions to the specification of a set of auxiliary complexity-growth conditions derived from the analysis of the operation of genetic regulatory networks. This leads to the definition of a particular structure for the kind of systems that will be evolved and to the specification of the genetic representation for them. A system with the required structure – for which the name *analog network* is suggested – corresponds to a collection of devices whose terminals are connected by links characterized by a scalar value of interaction strength. One of the specificities of the evolutionary system defined in this thesis is the way these values of interaction strength are determined. This is done by associating with each device terminal of the evolving analog network a sequence of characters

extracted from the sequences that constitute the genome representing the network, and by defining a map from pairs of sequences of characters to values of interaction strength.

Whereas the first chapter gives general prescriptions for the definition of an evolutionary system endowed with the desired complexity-growth potential, the second chapter is devoted to the specification of all the details of an actual implementation of those prescriptions. In this chapter the structure of the genome and of the corresponding genetic operators are defined. A technique for the genetic encoding of the devices constituting the analog network is described, along with a way to implement the map that specifies the interaction between the devices of the evolved system, and between them and the devices constituting the external environment of the evolved system. The proposed implementation of the interaction map is based on the local alignment of sequences of characters. It is shown how the parameters defining the local alignment can be chosen, and what strategies can be adopted to prevent the proliferation of unwanted interactions.

The third chapter is devoted to the application of the evolutionary system defined in the second chapter to problems aimed at assessing the suitability in an evolutionary context of the local alignment technique and to problems aimed at assessing the evolutionary potential of the complete evolutionary system when applied to the synthesis of analog networks. Finally, the fourth chapter briefly considers some further questions that are relevant to the proposed approach but could not be addressed in the context of this thesis.

A series of appendixes is devoted to some complementary issues: the definition of a measure of diversity for an evolutionary population employing the genetic description introduced in this thesis; the choice of the quantizer for the values of interaction strength between the devices constituting the evolved analog network; the modifications required to use the analog electronic circuit simulator SPICE as a simulation engine for an evolutionary or an optimization process.

Riassunto

Il notevole aumento della potenza di calcolo disponibile verificatosi negli ultimi decenni è coinciso con un crescente interesse per l'impiego di metodologie evolutive per la sintesi di svariati tipi di sistemi, in particolare per la sintesi di strutture quali i circuiti elettronici analogici, le reti neurali e, più in generale, i sistemi autonomi, per i quali non sono state sviluppate finora delle tecniche di progettazione sufficientemente generali e sistematiche. Sebbene l'approccio evolutivo applicato alla sintesi di sistemi di questo tipo abbia prodotto risultati indubbiamente interessanti, il problema costituito dalla definizione di un processo di evoluzione artificiale in grado di dar luogo a un significativo aumento della complessità dei sistemi così generati appare tuttora aperto.

Questa tesi affronta il problema della crescita evolutiva della complessità prendendo come punto di partenza le intuizioni contenute nei resoconti delle ricerche svolte (ma rimaste incompiute) verso la fine degli anni '40 e l'inizio degli anni '50 del secolo scorso da John von Neumann nel campo della logica degli automi e della loro auto-riproduzione. Le condizioni per la realizzazione della crescita evolutiva della complessità stabilite da von Neumann vengono qui integrate da una serie di condizioni ausiliarie dettate da quanto si è scoperto nel frattempo in merito alla struttura dei sistemi biologici in generale e al funzionamento delle reti di regolazione genetica in particolare. L'attenzione speciale dedicata a queste ultime è motivata dal loro status di più semplice struttura che possa essere considerata a pieno titolo un livello di organizzazione degli organismi viventi esistenti.

In quest'ottica, il primo capitolo è dedicato alla formulazione del problema della crescita evolutiva della complessità, partendo dalla descrizione delle condizioni suggerite da von Neumann per giungere alla definizione delle condizioni ausiliare ispirate dall'analisi del funzionamento delle reti di regolazione genetica. Questa analisi conduce alla definizione del tipo di struttura che dovranno possedere i sistemi oggetto del processo evolutivo e alla specificazione della relativa rappresentazione genetica. Un sistema dotato del tipo di struttura richiesto – per il quale si suggerisce il nome di *rete analogica* – corrisponde a una collezione di dispositivi i cui terminali sono collegati da connessioni caratterizzate da un valore scalare che definisce l'intensità

dell'interazione da esse istituito. Una delle peculiarità del sistema evolutivo definito in questa tesi è la modalità di assegnazione dell'intensità di queste interazioni. Essa si basa sull'uso di sequenze di caratteri estratte dalla collezione di sequenze che costituisce il genoma della rete analogica soggetta a evoluzione e associate ai terminali dei dispositivi della rete stessa. Una funzione opportunamente definita si occupa di associare ad ogni coppia di sequenze il valore dell'intensità dell'interazione corrispondente.

Se il primo capitolo si limita a dare una serie di indicazioni generali per la definizione di un sistema evolutivo dotato di potenzialità di crescita evolutiva della complessità, il secondo specifica tutti i dettagli di una implementazione di un sistema evolutivo corrispondente a quelle indicazioni. Questo capitolo definisce dunque innanzitutto la struttura del genoma e degli operatori genetici corrispondenti. Vengono poi definite la codifica genetica per i dispositivi che costituiscono la rete analogica e una possibile realizzazione della funzione che determina l'intensità dell'interazione tra dispositivi del sistema evoluto e tra questi e i dispositivi che costituiscono l'ambiente esterno del sistema evoluto. La realizzazione della funzione in questione si basa sull'allineamento locale di sequenze di caratteri. Vengono descritte le modalità di scelta dei parametri dell'allineamento locale e alcune strategie che permettono di evitare la proliferazione delle interazioni indesiderate.

Il terzo capitolo è dedicato alla descrizione dei risultati dell'applicazione del sistema evolutivo definito nel secondo capitolo ad una serie di problemi aventi lo scopo di verificare l'adeguatezza della tecnica di allineamento locale delle sequenze a un ambito evolutivo e le potenzialità del sistema evolutivo completo applicato a problemi effettivi di sintesi di reti analogiche. Infine, il quarto capitolo si occupa brevemente di alcuni ulteriori temi attinenti alle tematiche trattate nella tesi.

Una serie di appendici è dedicata ad argomenti complementari: la definizione di una misura di diversità per una popolazione soggetta a evoluzione che impieghi la rappresentazione genetica introdotta in questa tesi; la scelta del tipo di quantizzazione dei valori di intensità dell'interazione tra dispositivi che costituiscono la rete analogica soggetta a evoluzione; le modifiche che è necessario apportare al simulatore di circuiti elettronici analogici SPICE al fine di poterlo utilizzare come motore di simulazione nell'ambito di un processo evolutivo o di ottimizzazione.

Contents

Chapter 1

Introduction

Overview

This goal of this chapter is to state in informal terms the goal of the thesis, and to describe graphically its overall structure. This description is meant to help understand the role of the manifold elements that participate in the development of the argument leading to the definition of the artificial evolutionary system introduced in this thesis.

1.1 Autonomous systems and evolution

The Greek roots of the term "autonomous" refer to the capability of a system of giving to itself (*autós*) a rule or law (*nómos*). Usually this law is intended in terms of a behavior suited to the prevailing circumstances. For example, from an engineering point of view a system is considered autonomous if it is capable to perform a function in a range of environmental conditions, without the need of external control or guidance.

Biological systems are the living proof of the feasibility of autonomous systems capable of adapting themselves to an enormous range of environments, and to rapidly varying external conditions. Nonetheless, the design of artificial systems having comparable characteristics, appears still an open problem. Given the almost universally accepted derivation of biological systems from an evolutionary process (Bateson, 1979; Bonner, 1988; Mayr, 2001), and the related designer's lore according to which (Gall, 1986) "a complex system that works is invariably found to have evolved from a simple system that worked", evolutionary design methodologies have been proposed as a tool for the development of autonomous systems.

In short, an evolutionary design methodology maintains a population of individuals in the form of a genetic description called *genome* or *genotype*[1] for each individual. Each genotype is mapped by a suitable function or process into an instance, called *phenotype*, of the system to be designed, which can be tested with regard to the function that it is expected to perform. A value of *fitness* for each individual ensues, which is used to subject the population to a process of differential reproduction accompanied by the action of *genetic operators* that are intended to mimic the process of mutation and recombination that characterizes the replication of the genomes of biological populations.

The evolutionary approach has been applied to the design of many kinds of systems, and, in particular, within the realm of evolutionary robotics, to the development of control systems for autonomous robots (Floreano and Mondada, 1996; Harvey et al., 1997; Nolfi and Floreano, 2000). The evolution of autonomous robots able to perform simple tasks in simple environments has indeed encountered consid-

[1] *Genome* and *genotype* will be used almost interchangeably in this thesis, although, strictly speaking, "genome" refers to the genetic material considered as a raw sequence of symbols, whereas "genotype" refers to the genetic description considered as the structured repository of information which is used to produce the phenotype.

erable success. However, the current approach met with considerable difficulties when applied to the development of controllers for significantly more sophisticated tasks (Brooks, 2001a,b).

The main difficulty, in this respect, appears to be the lack of a convincing methodology for achieving an *incremental increase of complexity*[2] during the evolution and for producing a truly *open-ended evolutionary process* (Pattee, 1973). In fact, in their simplest form, evolutionary approaches employ genotypes having fixed structures, along with simple, direct mappings of the genotype space to the phenotype space (Boers and Sprinkhuizen-Kuyper, 2001), which establish a one-to-one correspondence between parts of the genotype and phenotypic traits (Dellaert and Beer, 1994; Nolfi and Floreano, 2000; Boers and Sprinkhuizen-Kuyper, 2001). It has been argued (Harvey, 1997) that this amounts to the reduction of evolution to nothing more than a classical optimization technique, that is, to the search of a solution within a fixed, predefined space. Clearly this has little chance of leading to radical innovation (Schuster, 1996). The goal of this thesis is the definition of an evolutionary methodology for the synthesis of autonomous systems, addressing explicitly the problem of the potentiality for an evolutionary increase of complexity.

1.2 Darwinian evolution

In order to understand the origin difficulties encountered by evolutionary approaches to the synthesis of systems it is useful to review the ideas upon which is based the explanation of the evolution of natural organisms. The Darwinian theory of evolution by natural selection derives the adaptation of organisms to their environment as a consequence of the following four principles (Darwin, 1859; Lewontin, 1974; Brandon, 1990).

The principle of variation. There is variation in the phenotypic traits (physiology, morphology, and behavior) among the individuals of a population.

The principle of heredity. The variation is to some degree inheritable.

[2]The term *complexity* can be temporarily interpreted with its ordinary meaning, which refers loosely to the number and diversity of elementary units interacting to produce a given function. The next chapter explores further the problem constituted by the definition of this term.

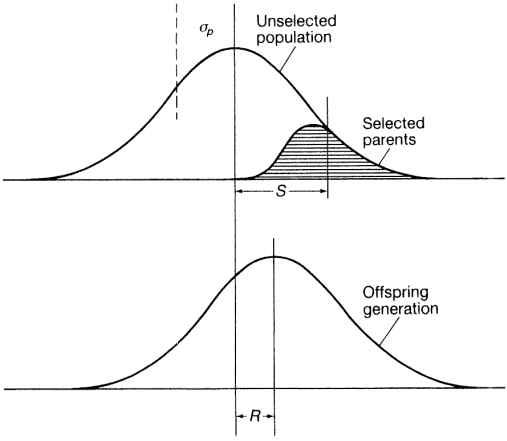
The principle of natural selection. Different variants leave different numbers of offspring in subsequent generations.

The principle of the struggle for existence. Variations that are useful to the organisms tend to increase their reproductive success.

The evolutionary design methodology described briefly in the previous section is obviously meant to implement those principles: the value of fitness associated with each individual system reflects its degree of functionality and influences its differential reproduction, with the objective of implementing the principle of natural selection and that of the struggle for existence; the action of the genetic operators acting on the genomes of the reproducing systems is aimed at introducing new variation in the offspring population while preserving to some degree the characters of the genome of the parents, with the objective of implementing the principle of variation and that of heredity.

The actual realization of the principle of heredity, however, is far from trivial. The difficulty lies in the fact that it is not a simple resemblance of the phenotypic traits of an individual with those of its parents that is sought but, rather, a resemblance of the *variation* of the phenotypic traits or, better, of the variation of the functionality corresponding to those traits. This important and often overlooked point can be better understood from an analysis of Figure 1.1, which is drawn from an evolutionary genetics textbook (Maynard-Smith, 1998). In the context of the evolution of artificial systems, the horizontal axis can be thought of as corresponding to a measure of the functionality of the system, the curves and the vertical lines represent the distribution and the mean, respectively, of the functionalities of the unselected population, of the collection of individuals selected as parents, and of the offspring population.

Let us first imagine that the distance denoted by R in the figure is actually zero, so that the distribution of the offspring population is identical to that of the unselected population, despite the fact that – due to the effect of selection – the distribution of the selected parents population is shifted to the right relatively to that of the unselected population. This would correspond to a case where the variation of the functionality is not heritable (note that the parameter $h^2 = R/S$ defined as *heritability* in the original caption of the figure is indeed zero in this case). It is easily seen that in this case the execution of a cycle of selection and reproduction does not result in any change in the distribution of the values of functionality of the individuals in the



Definition of some terms used in describing selection. The intensity of selection, $I = S/\sigma_p$; the realized heritability, $h^2 = R/S$.

Figure 1.1: *Figure 6.9 with its original caption from (Maynard-Smith, 1998) illustrates the fact that the concept of inheritance in the theory of evolution by natural selection refers to the* variation *exhibited by the evolutionary relevant characteristics of the selected parents, rather than to the characteristics of the parents. The effect of selection is to shift the distribution of the selected parents relatively to that of the unselected population, but evolution can take place only if the distribution of the offspring population is also shifted to some degree relatively to that of the unselected population, that is, if the characteristics of the variation of the selected parents are to some degree inheritable.*

population and, therefore, no evolution of the functionality can occur, no matter how many generations are observed.

If, on the other hand, the value of R is positive, so that the distribution of the functionality of the offspring population is shifted towards larger values relatively to that of the unselected population, a selection operated on the offspring population can result in a new collection of selected parents (not shown in the figure) whose distribution is shifted to the right relatively to the parents of the previous generation. As long as the value of R remains positive – that is, as long as the variation of functionality remains to some degree inheritable – the functionality can thus evolve, whereas as soon as R becomes null the evolutionary

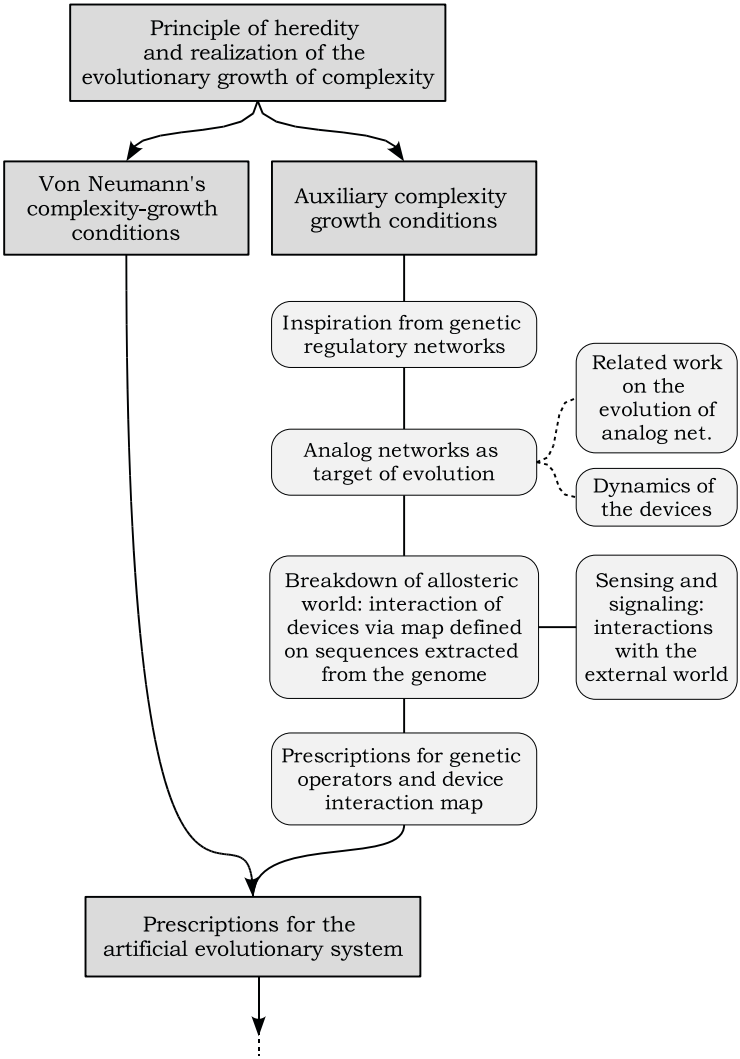
improvement of functionality comes necessarily to a halt.

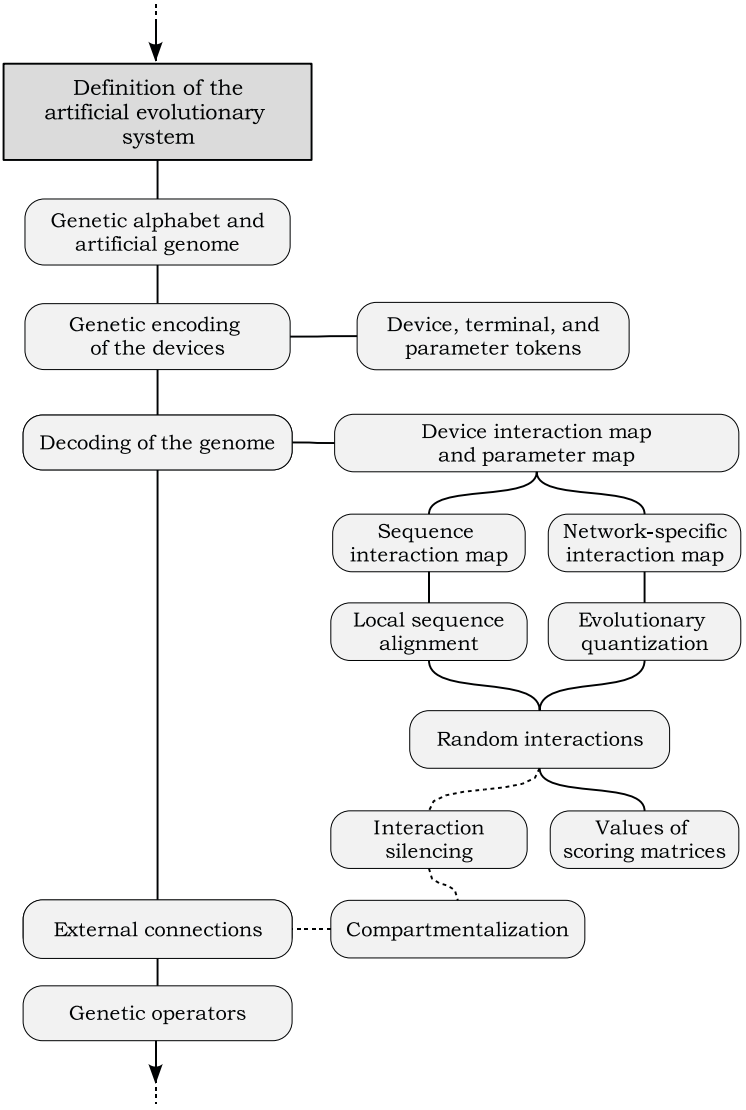
Although this is a very simplified model of what we can expect to observe in an actual artificial evolutionary scenario, it conveys the spirit of the principle of heredity in its referring to the inheritance of the variation of the functionality rather then in that of the phenotypic traits *per se*. In other words, in order for evolution to proceed, the operation of reproduction with mutation is required to produce the right kind of variation in the offspring population and, on the other hand, once the production of the right kind of variation is guaranteed, no additional obstacles exist for the realization of an effective artificial evolutionary process.

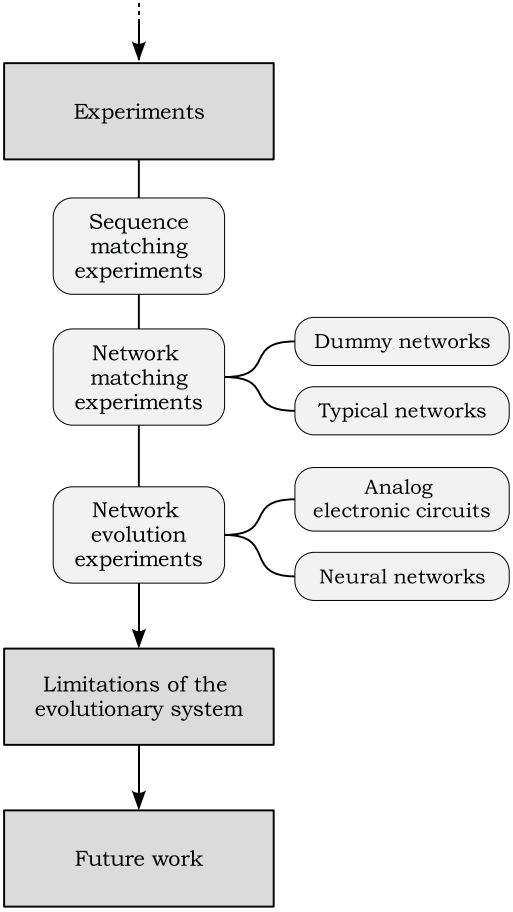
In the next chapter the problem of the realization of this condition is considered, first from an abstract point of view, and then from the perspective of our current knowledge of the working of biological systems. This will lead to the formulation of a series of prescriptions for the implementation of an artificial evolutionary system. In Chapter 3 these prescriptions will be turned into the details of an actual evolutionary system that can be implemented on a digital computer. In Chapter 4 the newly defined evolutionary system will be applied to a series of evolutionary problems, with the aim of assessing the evolutionary potential of its separate components and of the whole system when applied to the synthesis of systems of engineering relevance. Finally, Chapter 5 is devoted to an analysis of the limitations of the proposed evolutionary systems and to a description of the further developments that can be expected from the application of the principles that presided over its definition.

1.3 Structure of the thesis

The diagrams in the remaining pages of this chapter give a schematic overview of the thesis. The diagrams are meant to be used as a reference during the reading, assisting in the comprehension of the structure of the thesis, in the situating of each subject in the global context, and in the understanding of the relations between the different parts of the thesis. The rectangles with a dark background refer to the high-level nodal points of the argumentation, whereas the rounded rectangles detail the structure of the nodal point from which they originate, with the horizontal links providing either a further level of detail about particular subtopics (continuous lines) or a reference to related accessory topics (dotted lines).


Principle of heredity and realization of the evolutionary growth of complexity
Von Neumann's complexity-growth conditions
Auxiliary complexity growth conditions
Inspiration from genetic regulatory networks
Related work on the evolution of analog net.
Analog networks as target of evolution
Dynamics of the devices
Breakdown of allosteric world: interaction of devices via map defined on sequences extracted from the genome
Sensing and signaling: interactions with the external world
Prescriptions for genetic operators and device interaction map
Prescriptions for the artificial evolutionary system


Definition of the artificial evolutionary system
Genetic alphabet and artificial genome
Genetic encoding of the devices
Device, terminal, and parameter tokens
Decoding of the genome
Device interaction map and parameter map
Sequence interaction map
Network-specific interaction map
Local sequence alignment
Evolutionary quantization
Random interactions
Interaction silencing
Values of scoring matrices
External connections
Compartmentalization
Genetic operators


Experiments
Sequence matching experiments
Network matching experiments
Dummy networks
Typical networks
Network evolution experiments
Analog electronic circuits
Neural networks
Limitations of the evolutionary system
Future work

Chapter 2

The growth of complexity[1]

Overview

*The goal of this chapter is to introduce the problem of the **evolutionary growth of complexity** and to frame the objectives of the thesis in the context of that problem. The chapter starts with an analysis of the application of artificial evolution to the synthesis of autonomous systems in relation with the problem constituted by the necessity to obtain incrementality and open-endedness of evolution. First, **John von Neumann's insights** on the problem of the growth of complexity – which suggest the adoption of a system with a genotype distinct from the phenotype and a process of decoding of the genotype that can tolerate major reorganizations of the genotype – are discussed. The status of the complexity-growth conditions devised by von Neumann is then analyzed and the need to integrate those conditions with some **additional evolvability conditions** is argued. These additional conditions are investigated by analyzing and abstracting the working of genetic regulatory systems, considered as the first functional level of expression of the genotype-to-phenotype mapping. Finally, the principles that will guide the assembly of the **evolutionary system** presented in this thesis are formulated. The main points are the focusing on the evolution of network-like systems called **analog networks** that are decoded from genomes whose bio-inspired structure and decoding tolerates the action of a variety of genetic operators, which can produce both smooth and abrupt transformations in the structure of the decoded network. The chapter is concluded by a **review of related work**, focused on existing systems for the evolution of genetic regulatory networks, analog electronic circuits, and neural networks.*

[1] Parts of this chapter were published in (Mattiussi and Floreano, 2004b).

2.1 Complexity

John von Neumann was probably the first to address in formal terms the problem of the evolutionary increase of complexity. Although his work on self-reproducing automata was left unfinished, his notes on the subject were edited and published along with the transcript of a series of lectures on the "Theory and Organization of Complicated Automata" (von Neumann, 1966). In these lectures von Neumann does not attempt a formal definition of "complexity"; however, he clearly recognizes the importance of this concept – to which he often refers with the alternative term "complication" – to the point of entitling two of the lectures that compose that work: "The Role of High and of Extremely High Complication" and "Re-evaluation of the Problems of Complicated Automata - Problems of Hierarchy and Evolution". In the latter he writes

> There is a concept which will be quite useful here, of which we have a certain intuitive idea, but which is vague, unscientific, and imperfect. *This concept clearly belongs to the subject of information*, and quasi-thermodynamical considerations are relevant to it. I know no adequate name for it, but it is best described by calling it "complication." *It is effectivity in complication, or the potential to do things. I am not thinking about how involved the object is, but how involved its purposive operations are.* In this sense, an object is of the highest degree of complexity if it can do very difficult and involved things. (von Neumann, 1966, p. 78) [emphasis added]

Today, we still do not have a satisfying formal definition for the intuitive concept described by von Neumann (Edmonds, 1999). In this thesis the term "complexity" will be used in the heuristic sense indicated by von Neumann in the above quotation, that is, as an indicator of the potential of a system to engage in purposive operation.

Remark: Concerning the problem of a formalization of the concept of complexity, it interesting to quote von Neumann further

> There is thus this completely decisive property of complexity ... Now, none of this can get out of the realm of vague statement until one has defined the concept of complication correctly ... There is nothing new about this. It was exactly the same with ... the concepts of energy and entropy ... The

> simplest mechanical and thermodynamical systems had to be discussed for a long time before the correct conceptions of energy and entropy could be abstracted from them. (von Neumann, 1966, p. 80)

The history of Probability Theory as described by Jaynes (1986, 2003) can suggest how an acceptable formalization will eventually emerge. The crucial point is the realization that complexity – like entropy – cannot be considered a property of the system alone. As the value of entropy that we attribute to a system depends on our state of information about the system[2] (without becoming arbitrary, since two observers with the same state of information about the system will attribute to it the same value of entropy), the value of complexity can probably be defined only in relation to the kind of behavior of the system which interests us (possibly in view of prediction or control of this behavior). It is indicative, in this respect, that in relating the concept of complexity to the subject of information von Neumann used an expression that brings to mind the view of "Probability Theory as Extended Logic" advocated by Jaynes (1990, 2003)

> The view that probability is an extension of logics is not trivial, is not generally accepted, and is not the major interpretation of probability. It is however the classical interpretation. ... There's a great deal in other modern theories, for instance, in quantum mechanics, which inclines one very strongly to take this philosophical position, although the last word about this subject has certainly not been said and is not going to be said for a long time. (von Neumann, 1966, pp. 58-59)

2.2 Universal constructors

We owe to McMullin (1992, 2000) a persuasive analysis of the history of the reception of von Neumann's ideas in the scientific community. According to this analysis, there has been since the publication of (von Neumann, 1966) a tendency to assume that the problem that von Neumann was trying to solve with the definition of his automaton was that of self-reproduction, while in reality the focus of that work was the

[2] Although many physicist will object to this statement, this was the position of von Neumann and also that of Wolfgang Pauli, as testified by the quotations included in (Popper, 1982, Chapter I, Section 5). Note that in that essay Popper opposes strongly the interpretation supporting the subjective status of entropy.

problem of the growth of complexity. Supporting this interpretation is the fact that subsequent work (for example, Langton 1984) has proved that self-reproduction alone can be obtained with much simpler means than those employed by von Neumann.

A peculiarity of von Neumann's work is the effort to include in his automaton a *universal constructor* capable to interpret the description of an arbitrary automaton and use that description to build an actual instance of it[3]. The radically simpler self-reproducing automata introduced by subsequent authors, on the other hand, achieve their simplicity by giving up this key property. The following extract from (Langton, 1984) is eloquent in this respect

> [In this paper] self-reproduction in cellular automata is discussed with reference to the models of von Neumann and Codd. The conclusion is drawn that although the capacity for universal construction is a *sufficient* condition for self-reproduction, it is not a *necessary* condition. Slightly more "liberal" criteria for what constitutes genuine self-reproduction are introduced, and a simple self-reproducing structure is exhibited which satisfies these new criteria.

Langton's self-reproducing automaton, like von Neumann's, is composed of a description and a constructor. The rules of the "universe" where Langton's automaton is defined ensure that the interaction of the description with the constructor leads to the production of replicas of the automaton. Moreover, like von Neumann's, Langton's replication process ensures that a copy of the description is transmitted to the new automaton without being interpreted. The fundamental difference between Langton's and von Neumann's automata is that the structure which acts as constructor in the former is not intended to be universal but is instead specifically targeted to the particular description of that automaton. The resulting automaton is indeed self-replicating and much simpler than von Neumann's, but this simplicity comes at a price: the constructor is not explicitly designed to tolerate a large ensemble of mutations of the description. We could say that Langton's automaton is specified to be self-replicating, but not necessarily self-reproducing (although, as shown in (Salzberg et al., 2004) and in (Salzberg and Sayama, 2004), it still has some evolutionary po-

[3]The constructor is "universal" in the sense that it can build any machine that can be described in the format required by the constructor, not in the sense that it can build any machine *tout court* (Taylor, 2001).

tential), whereas von Neumann's is – at least in principle – defined to be truly self-reproducing[4].

We can summarize all this saying that von Neumann had a double insight: first, that to build a system with the potential of evolutionary growth of complexity it is useful to maintain a description of the system (a "genome") distinct from the machinery that builds the system (the constructor); second, that the constructor must be able to decode the description when the description is subjected to a wide range of mutations spanning the space of descriptions of a large collection of systems. Note that we are talking here of a *potentiality*, which must not be confused with the *certainty* of the growth of complexity. Intuitively, most of the mutation paths will lead to non-viable systems or to systems with smaller complexity (Edmonds, 1999; McMullin, 2000) and, in fact, natural evolution does not lead necessarily to an increase of complexity of the evolved organisms (McShea, 1991; Maynard-Smith, 1994, p. 469). However, in the spirit of von Neumann's observation quoted above, according to which complexity of an organism is related to "how involved its purposive operations are", if the evolutionary scenario can accommodate a niche whose exploitation requires the execution of involved purposive operations, the exploitation of that niche has the potential of complexity growth as a necessary prerequisite.

Note that von Neumann's analysis is based on the hypothesis of a fixed constructor operating on a mutable description. One could contemplate the possibility of subjecting also the constructor to mutation and therefore to evolution. Von Neumann considered briefly this pos-

[4]As observed by Sipper et al. (1997, p. 89)

> It is important to distinguish between two distinct terms, replication and reproduction, which are often considered synonymous. Replication is an ontogenetic, developmental process, involving no genetic operators, resulting in an exact duplicate of the parent organism. ... Reproduction, on the other hand, is a phylogenetic process, involving genetic operators such as crossover and mutation, thereby giving rise to variety and ultimately to evolution...

and in the same vein, von Neumann (1966, p. 86) writes

> One of the difficulties in defining what one means by self-reproduction is that certain organizations, such as growing crystals, are self-reproductive by any naive definition of self-reproduction, yet nobody is willing to award them the distinction of being self-reproductive. A way around this difficulty is to say that self-reproduction includes the ability to undergo inheritable mutations as well as the ability to make another organism like the original.

sibility but discarded it (von Neumann, 1966, p. 86) on the grounds of the low probability of obtaining a working constructor. This was probably a wise choice, given the goals of von Neumann's work and the simplifications that derive from keeping the constructor fixed. Nonetheless, the evolutionary potential of a system that can mutate both the description and the constructor is arguably greater than that of a system that limits the mutations to the description (McMullin, 1992, 2000; McMullin et al., 2001). In fact, it is difficult to conceive the evolution of existing organisms without the parallel evolution of both aspects, given the great complexity of their constructing machinery. On the other hand, once this complex "universal" constructor has evolved or has been defined by an experimenter, it is probably true that – as conjectured by von Neumann – most mutations will interfere with its operation, and the evolutionary process will concern almost exclusively the description.

Von Neumann also considered and discarded for fear of logical paradoxes the possibility of a machine that does not use a separate description and reproduces by self-inspection (von Neumann, 1966, p. 122; McMullin, 1992, p. 175; Pattee, 1995b, pp. 10-11). Laing (1977) proved that von Neumann's worries were unfounded. However, the reproduction by self-inspection can present some practical difficulty (Pattee, 1995b; McMullin et al., 2001) and apparently no existing organism reproduces using that approach. Looking for other possible reproduction strategies not based on the use of a separate description, we can consider that not all information is transmitted genetically in the reproduction of existing organisms. For example, unicellular organisms display a reproduction mechanisms that is in part based on the genetic information and in part on the structural information constituted by the reproducing cell (Harold, 2001, pp. 99-115), and even in sexually reproducing multicellular organisms there is a lot of information that is conveyed by the spatio-temporal cellular organization of the zygote (Jablonka and Lamb, 1995; Harold, 2001). However, the structural information transmitted by non-genetic means appears limited to spatio-temporal configurational information, whereas a corresponding transmission of, say, regulation or signalling network structure, or of the global structural information of multicellular organisms appears operationally much more problematic.

We can therefore conclude that the distinction of genotype (description) and phenotype (decoded description), the presence of a "universal" constructor for the description along with a high reorganizability

of the description, and the possibility of generating in this way a potentially unlimited variety of decoded systems are instrumental to the implementation of an evolutionary system endowed with the potential of growth of complexity. Let us call these conditions "von Neumann's complexity-growth conditions".

2.3 Auxiliary conditions

The previous section was deliberately vague about the status of von Neumann's complexity-growth conditions. Are these conditions necessary for the open-endedness and complexity-growthness[5] of an evolutionary system? Are they sufficient?

In a mathematical sense these conditions are almost certainly not necessary. As mentioned above, there appears to be no fundamental logical obstacle in the reproduction of a system by self-inspection, or using the existing system as template for the offspring, and both these processes can include inheritable mutations. Hence, at least the distinction between genotype and phenotype does not appear a necessary prerequisite for the evolutionary growth of complexity. On the other hand, the production of a constructive proof of the sufficiency of von Neumann's complexity-growth conditions was presumably the goal of the work reported in (von Neumann, 1966), and one of the reasons of the choice of the formal universe of cellular automata for the implementation of the system.

From a practical point of view, however, the von Neumann conditions are not sufficient for there being a reasonable probability of actually observing complexity-growth evolution within an artificial evolution experiment. The difference is that from a mathematical point of view a vanishingly small non-null probability of production of a certain outcome is often sufficient to prove that in a suitably defined limit sense a process will almost certainly produce that outcome. From a practical viewpoint, instead, we demand that the probability of the process actually producing the desired outcome be sufficiently high given the limited resources and time available for the experiment. More specifically, within an artificial evolutionary experiment we work with a finite population that we can think of as moving from generation to generation within an abstract genotype space according to the effect of

[5] "Complexity-growthness" is admittedly a neologism, shaped on the technical term "old-growthness" used in forestry (Holt et al., 1999).

the genetic operators acting in the genotype space and of the selection operator acting in the phenotype space (Lewontin, 2001). Given an initial population and preassigned resources for the evolutionary experiment, we could in principle calculate the probability of the population accessing certain regions of the genotype and phenotype space (Fontana and Schuster, 1998a,b; Stadler et al., 2001). The fact of satisfying von Neumann's complexity-growth conditions does not in itself ensure a significant probability of access to all the regions of the phenotype space. For example, changes that require the concurrence of two or more independent and improbable mutations of the genomes of existing individuals, would have a negligible probability of being observed (Conrad, 1972). If all the "interesting" phenotypes are confined to regions of the space that have a negligible access probability, the evolutionary process will almost certainly not produce any such phenotype. On the other hand, the remarks made in the previous section about the difficulty of achieving a substantial growth of complexity using reproduction by self-inspection or by template copy, suggest that the distinction of genotype and phenotype, although not necessary in a mathematical sense for the evolutionary growth of complexity, is still practically required for it (Pattee, 1981).

The moral of this story is that we must add some auxiliary conditions to von Neumann's conditions to obtain a set of evolutionary complexity-growth conditions that will work in practice, that is, a system that actually possesses *evolvability* (Kirschner and Gerhart, 1998; Wagner and Altenberg, 1996). These conditions should ensure the practical accessibility of all the phenotype and genotype space (or, at least, of all its interesting regions – although this concept could be difficult to define) on the part of the evolving population. It is clearly easy to ensure the accessibility of all the regions of the genotype space, since the "motion" of the population is due the genetic operators acting in the genotype space, and we are free to define their properties and to assign their probability at will. However, the probability of survival and reproduction of the individuals is determined in the phenotype space. Therefore, these auxiliary conditions for the growth of complexity must concern primarily the mapping from genotype to phenotype (Wagner and Altenberg, 1996).

As discussed in Chapter 1, this can be rephrased in more Darwinian terms. The march of an evolutionary process based on selection requires the presence of genetic variation, or diversity, that is felt at the phenotypic level and can be exploited by selection. A consequence of

the action of the selection process is that the selectively useful variation existing at a certain instant in the population tends to be "used up", and must be regenerated in order for evolution to continue its operation. The action of the genetic operators must ensure this regeneration of variation. However, it is not sufficient to ensure the regeneration of some arbitrary genetic diversity, and not even the regeneration of phenotypic diversity if this diversity cannot be exploited by selection to ensure the march of evolution. What we must instead ensure with some auxiliary conditions is the generation of diversity that is *useful* in selective, evolutionary terms. This means that, given a genotype with its corresponding phenotype, the genotypes that can be accessed with non negligible probability must permit the gradual evolutionary exploration of the phenotype space, without being systematically frustrated by the generation of non-viable or evolutionary grossly inferior phenotypes (Stadler et al., 2001).

The inquiry into these auxiliary "practical evolvability and complexity-growth conditions" will proceed in the direction opposite to that followed by von Neumann. When von Neumann formulated his conditions, the molecular details of the workings of the genome where still unknown, and his insights proceeded from the logic of the argument he was about to prove.[6] Since the goal of this thesis is not the formalization of evolvability and complexity-growth conditions but the definition and implementation of an artificial system actually displaying those properties, we will instead capitalize on the current knowledge of those molecular details trying to capture the mechanism that in biological organisms ensures the required evolutionary complexity-growth conditions.

2.4 Levels of organization

In the previous section it was argued that it is worth examining in detail the characteristics of the genotype-to-phenotype mapping of biological organisms since it plays a crucial role in determining their evolutionary complexity-growth potential. The phenotype of a complex organism, however, is organized into a functional hierarchy composed of several levels of organization (Grene, 1987; Miller, 1978) and it would be unwise (as well as unpractical) to consider at this point the unfolding of

[6] It is therefore telling that many of the concepts he formulated were later found to have a biological counterpart.

the genetic description into the whole hierarchy of levels. Fortunately, this unfolding can be analyzed level by level, with each phenotypic level of organization representing an ensemble of dynamic potentialities that are constrained by the next higher level to determine both its functionalities and a new ensemble of dynamic potentialities (Pattee, 1973; Polanyi, 1968). These, in turn, are harnessed by yet another level, and so on, up to the highest hierarchical level that does not transcend the organism's individuality (where, possibly, starts a sequence of *social* levels of organization).

To assess the properties of genotype-to-phenotype mapping it is therefore convenient to start by focusing on the mapping of the genotype into what is currently understood as one of the basic phenotypic functional level of cell organization: that of the *genetic regulatory network* (GRN)[7]. It is important to realize that this can be considered a complete genotype-to-phenotype mapping, since genetic regulatory networks are the information processing infrastructure of unicellular organisms (Bray, 1995; Welch, 1996; Simpson et al., 2004a; Weiss et al., 2003) and their operation has a fully realized functional significance irrespective of any higher level of functional and morphological organization. Moreover, GRNs have the advantage of being a context where a good part of the information exchange takes place internally, between elements of the system, and the system is therefore in some measure free to choose its own coding, unhampered by external constraints. As will be shown in Section 2.7, this translates into a particular simplicity of the interaction between the elements that constitute the nodes of the network, and, therefore, into a particular simplicity of their abstract models. Note that once a convincing evolutionary process has been established at this basic level of organization, nothing prevents us for exploring the possibility of enhancing the potential of evolutionary complexification of the system by considering also the unfolding of further levels of organization, especially in relation to the process of development of compartmentalized (for example, multicellular) and hierarchically organized systems (Pattee, 1973; Conrad, 1990).

[7] Strictly speaking, one can identify functional substructures within the level of organization of the genetic regulatory network of a cell (Alm and Arkin, 2003; Barabási and Oltvai, 2004). However, the properties of the genotype-to-phenotype mapping for these substructures is the same as that of the whole network.

2.5 Abstracting the biological constructor

Even moderately detailed reviews of what is currently known on the mapping of genotypes into the phenotypes of biological organisms and on the structure and function of GRNs, turn easily into bulky volumes (for example, Alberts et al., 2002; Lewin, 2004; Watson et al., 2003b). The exposition will therefore be limited to a brief, simplified, and schematic description of some aspects in a perspective that is functional to our goal.

From an abstract point of view, the genome of a biological organisms is a symbolic description constituted by one or more sequences of characters belonging to an alphabet of four letters (the genetic or DNA alphabet). With some abuse of terminology,[8] a single sequence will be called a *chromosome*. Mixing the abstract and molecular levels, the characters that enter each chromosome can be called *nucleotides*. Within each chromosome there are substrings (i.e., sequences of adjacent nucleotides)[9] called *coding regions* or *transcriptional units* that are subject to a process called *transcription* by a molecular machine called RNA polymerase. The result of the transcription of a coding region is still a sequence of characters, which is the transliterated version of the gene using another alphabet of four letters (the RNA alphabet), possibly with some transcription error. This new sequence of characters is typically subjected to some reorganization such as the excision of one or more substrings. The reorganized string can then have different fates. In the case that interests us now,[10] it is subjected to a *translation* which corresponds to an encoding that substitutes – possibly with some translation error – non-overlapping triplets of nucleotides with characters belonging to an alphabet of 20 letters (the amino acid alphabet). Since there are 64 different nucleotide triplets, the mapping from the set of nucleotide triplets to the 20 letter alphabet has a many-to-one nature.

Up to this point, the symbolic description constituted by the original

[8] A real chromosome is composed of two sequences of nucleotides, but since these two sequences are complementary each single sequence carries the information content of the complete chromosome.

[9] A *subsequence* of a sequence of characters (or, which is the same, of a string) is a sequence obtained deleting zero or more elements of the original sequence. Hence, a subsequence is composed of characters that are not necessarily adjacent in the original sequence. On the contrary, a *substring* of a string is composed of adjacent characters of the original string.

[10] We will consider later the direct use of fragments of the transliterated string as control elements in GRNs.

coding region has been simply converted into another symbolic description by a many-to-one mapping. We have remained within a symbolic realm where the correspondence with an abstract space of character sequences almost imposes itself. From this point starts a process that leads to the formation of proteins and which is based heavily on the real-world physical and chemical dynamics. To understand the implications of this further process from the point of view of the genotype-to-phenotype mapping, it is preferable to describe it in concrete rather than abstract terms. To this end, it is sufficient to specify that the resulting sequences from the 20 letter alphabet are materially composed of amino acids and correspond to *polypeptide chains*. A polypeptide chain is a chemical structure existing in the three-dimensional space. Due to the kind of chemical bonds that exist within amino acids and between the amino acids, the polypeptide chain is not a rigid structure but can assume many different three dimensional conformations. However, in an environment that allows the formation of weak chemical bonds (e.g., an aqueous solution with suitable physico-chemical characteristics and, of course, the interior of a living cell) a polypeptide chain folds into a unique three-dimensional structure.[11] The resulting molecule is a *protein*, and the process of polypeptide chain folding is called for short *protein folding*. The whole process that transforms a gene into a protein is called *gene expression*.

2.6 Devices interacting via sequences

By changing how much the proteins encoded in the genome are produced – from a minimal, basal level of production to the maximum rate of expression of the corresponding protein given the prevailing internal conditions – a cell can adapt its activity to the changing requirements of its environment and of its life cycle. This process is called regulation of gene expression or, for short, *gene regulation*. The term *gene* is used here to refer collectively to all the parts of the genome that are involved in the production of a given protein (or group of proteins, if the coding region corresponds to more than one protein) and in the regulation of this production.[12] This section will describe schematically how gene

[11] In some cases the presence of additional proteins called *chaperones* is required to guide the folding process (Lewin, 2004).

[12] More generally, the term *gene* will be applied to parts of genome that are involved in the transcription of a given transcriptional unit and in the regulation of that transcription. This includes the case leading to the production of proteins which is

regulation is performed in cells. The goal of this analysis is to show how gene regulation can be interpreted abstractly as the operation of a network of devices whose interaction is determined by sequences of nucleotides belonging to the genome.

As described above, gene expression starts with the transcription of the coding region by a molecular machine called RNA polymerase, and leads eventually to the production of one or more proteins. To initiate transcription, an instance of RNA polymerase must be recruited to the gene and put in the correct position relative to the start of the coding region. This phase is the one where gene regulation is most commonly performed (Ptashne and Gann, 2002, p. 5). A region of the genome called *promoter*, positioned near the start of the coding region, is responsible for attracting the RNA polymerase to the gene. In most cases, however, the mere existence of the promoter is not sufficient to guarantee the correct initiation of the transcription, and the expression of the gene proceeds at a very low level unless specific *gene regulatory proteins* called *activators* are bound to the genome. An activator recognizes a region of the genome called *activator binding site* situated typically (but not necessarily) in the vicinity of the coding region and interacts with the RNA polymerase in a way that facilitates the initiation of the transcription. The final result is *positive regulation* of the gene, that is, an increase of its level of expression. An opposite effect is caused by other gene regulatory proteins called *repressors*, which recognize and bind to regions of the genome called (for historical reasons) *operators* and interfere with the execution of the transcription. In these cases the result is *negative regulation* of the gene, that is, a decrease or a zeroing of its level of expression. The regions of the genome that act as binding sites for activators or repressors are called *regulator binding sites* or, generically, *regulatory regions* or *regulatory sequences*.

Cells use these mechanisms of regulation in the biological circuits that constitute the GRNs. In these circuits each type of molecule acts as a signal whose amplitude is represented by the local concentration of the molecule. The elementary circuit element is constituted by a *regulatory gene* which is activated or repressed by a protein A and produces a protein B. In turn, the protein A is produced by another gene, and the protein B can act as activator or repressor for still another gene. We have thus a scenario where the sequence of nucleotides that constitutes a regulatory gene leads to the production of a signal repre-

described here, and the case leading to the production of fragments of RNA endowed with regulatory activity, which is described in Subsection 2.7.1, below.

sented by the molecules of the protein encoded by the gene. This signal interacts with another sequence of nucleotides to regulate the level of expression of another gene, and so on.

We can reformulate all this saying that a gene (along with the transcription machinery) acts as a device which accepts input signals and produces output signals,[13] and where the strength of the interaction between the output of a device and the input of another device – i.e., the strength of the interaction between the genes – is determined by the relationship between a pair of sequences of nucleotides. To obtain a graphic description of this scenario, we can represent each gene as a device having a sequence of nucleotides associated with each input and with each output, as if each sequence constituted a label attached to the corresponding input or output.[14] In this representation the regulatory regions contain the sequences of nucleotides attached as sequences of characters (nucleotides) to the input terminals of the device, whereas the coding regions contain the sequences of nucleotides attached as labels to the output terminals of the device (Figure 2.1).

Using this representation for the genes, a GRN can be represented as a network of devices, where each gene is represented by a device, and the connections between the devices is determined by the interaction between the gene products (which can be proteins or RNA fragments) and the gene regulatory regions. It can be shown that the interaction between a pair of genes can be summarized by a scalar value that gives the strength of the interaction, and appears as a parameter in the equations that model the GRN and are used for its numerical simulation (Bower and Bolouri, 2001; Kitagawa and Iba, 2002; Whitehead et al., 2004). Since the strength of the interaction between two genes depends on the sequences of nucleotides that constitute the coding and regulatory regions of the two genes, we can think of the value of interaction strength between two genes as obtained through a map[15] from pairs of sequences to scalar values of interaction strength. Since – as explained above – the genes can be considered abstractly as devices, we will call this map *device interaction map*. The final result is a net-

[13] In general, many regulatory proteins, that is, input signals, can control the expression of a gene, and many different proteins, that is, output signals, can be produced by a single gene.

[14] An even more fitting representation sees the RNA polymerase playing the role of the device.

[15] *Map* (or *mapping*) is a synonymous of *function*, that is, it corresponds to a way of associating unique elements of a set – in this case the set of real values of interaction strength – to every element in another set – in this case the set of pairs of sequences of nucleotides.

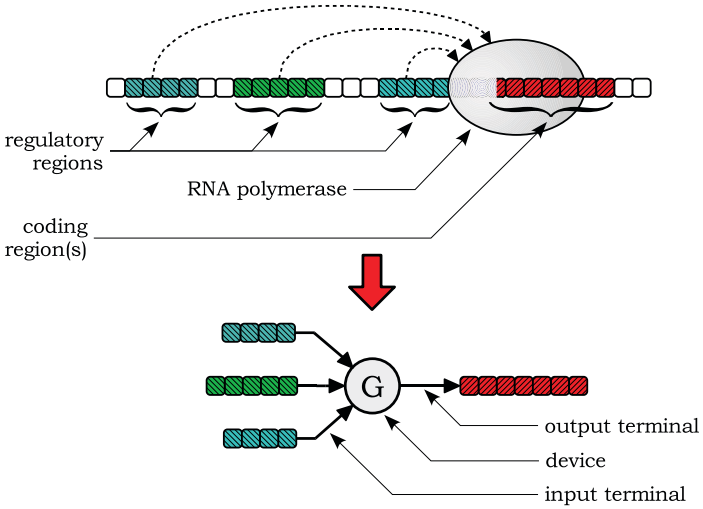


Figure 2.1: *A gene contained in a fragment of chromosome (represented schematically in the top part of the figure) can be interpreted as a device with input and output terminals, and with sequences of nucleotides associated with the terminals (bottom). The regulatory regions of the gene correspond to the sequences associated with the input terminals, the coding regions correspond to the sequence associated with the output terminals, and the RNA polymerase acts as a device that mediates between the inputs and the outputs.*

work of devices connected by links characterized by a scalar value of interaction strength which is obtained using a device interaction map that takes as arguments pairs of sequences of nucleotides belonging to the genome (Figure 2.2).

We can identify in this representation of GRNs two issues that are crucial to the success of their evolutionary synthesis. The first issue concerns the possibility of changing the number and type of devices that are present in the network. The second issue is the possibility to change the strength of the interaction that characterizes the connections between pairs of devices, including the possibility of eliminating altogether the connection and ensure the absence of any direct interaction between them. The evolutionary picture of biological GRN provided by molecular biology proves that biological systems have both

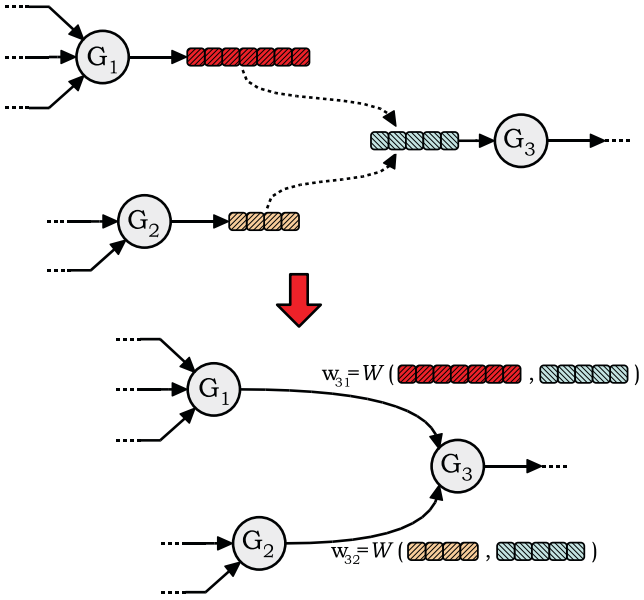


Figure 2.2: *A representation for a genetic regulatory network (GRN) can be deduced from the collection of its genes using the representation for the genes shown in Figure 2.1. Each gene corresponds to a device having some input and output terminal, to which sequences of nucleotides are associated (top). The strength of the interaction between two genes depends on the sequences of nucleotides through a device interaction map $W(s_j, s_i)$ from pairs of sequences (s_j, s_i) to values w_{ij} of interaction strength (bottom). The result is a network of devices connected by links that are characterized by a scalar value w_{ij}*

these possibilities. In Section 2.8 we will tackle the problem of how to implement these features in an artificial evolutionary system, having in mind the problem of the evolvability of the resulting system. Before doing that, however, we must deal with a serious problem concerning the nature of the map that associates values of interaction strength with pairs of sequences of nucleotides, i.e., a problem concerning the nature of the device interaction map. The problem is that if it turns out that the evolvability properties of biological system depend crucially on a way of implementing that mapping that is intractable with the current

computational resources, our analysis is bound to remain a theoretical speculation without possibility of actual implementation. The next section is devoted to the analysis of this question.

2.7 The breakdown of the allosteric world

Proteins can carry out many functions in cells and organisms, from being used as building material, to operating as molecular machines, to acting as transducers of chemical signals, to being signals in themselves, and still many others. In all cases the execution of their function depends critically on their three-dimensional shape. In many cases, it depends also on their changing shape in response to some specific signal, a phenomenon called *allostery*.

At the beginning of the molecular biology era, when most molecular details of gene regulation were still unknown, allostery was commonly assumed to play a major role in all phases of gene regulation. It is now recognized that the working of gene regulatory proteins depends on allostery only in a minority of cases (Ptashne and Gann, 1997, 2002; Watson et al., 2003b). In most cases the action of gene regulatory proteins is instead limited to the identification of a subsequence[16] of the sequence of nucleotides constituting the regulator binding site. Moreover, the interaction of an activator with the transcription machinery is typically limited to its recruitment to the corresponding gene via a generic adhesive interaction. In other words, not only allostery but also the complex details of the global three-dimensional shape of the gene regulatory proteins have been found to play a minor role, the latter typically bringing specificity to the interaction only in terms of recognition of a linear sequence of nucleotides.

The surprise of molecular biologists at these findings which overturn the "classical" assumptions is well testified by the following extract (Ptashne and Gann, 2002, pp. 174-176)

> Many of us, we suspect, came to assume, perhaps subconsciously, that allostery must lie at the heart of all biological regulation. Allostery appeals to our innate love of complexity: each allosteric response requires evolution of a protein that responds in an appropriate way to the allosteric signal, and one would expect no general rules. That is, each

[16] See footnote 9 on page 21.

> allosteric enzyme would have its own complicated, integrated structural features that would allow the proper conformational change to the signaling chemical.
>
> ... [W]e suspect that many of us tacitly assumed we lived in an "allosteric world", a world based on beautifully evolved sets of machines that were turned on and off much as a car engine is switched on and off by the turning of the key.
>
> The language we have used for many years reflects this bias: we speak of gene "activation" even for those cases (the majority we have argued) where either the polymerase nor the gene is activated in any traditional sense. Rather, the enzyme [i.e., the polymerase] is merely apposed with the substrate (in this case a specific gene) by an "activator".
>
> ... That so much of the specificity of regulation and hence so much of development and evolutionary change depends on simple binding interaction is (or we think should be) hard to swallow. It certainly is for us. We, and we suspect many others, had expected that the meanings of biological signals would have been, somehow, more solidly based.
>
> ... But ... we realize that these systems evolved, stepwise. And so it should hardly be surprising that underlying all the complexities are certain rather simple mechanisms that, by being reiterated and constantly added to, can produce living systems.

For us, the importance of this finding is that it shows a much simpler scenario for gene-to-gene interaction than was once thought. The point is that the simulation of the process of protein folding for the determination of its three dimensional structure is computationally very complex (Haspel et al., 2003), and of course even more so the dynamic simulation of the change of shape related to allostery. Should the gene-to-gene interaction and its evolvability depend crucially on allostery or on the details of protein folding that determine their complicated three-dimensional shapes, we would be at a loss in abstracting and implementing it within an artificial evolutionary system. The fact that in most cases this interaction is governed by a "simple binding interaction" aimed at the recognition of a linear sequence of nucleotides opens the way to the possibility of implementing it in terms of a mapping of low computational complexity.

Note that this must not be interpreted as saying that allostery and

the three dimensional shape of proteins play a minor role in the operation of biological systems. The working of the RNA polymerase in performing transcription, for one thing, and that of the other molecular machines operating in gene expression depend critically on their three-dimensional shape and on the possibility to change it. Moreover, the recognition of the nucleotide sequence operated by a gene regulatory protein is certainly highly dependent on the fit between the surfaces of the molecules that constitutes the genome and the protein. Finally, when proteins perform a structural function, metabolize nutrients, or exchange signals with the external world, they must adapt to the constraints of that world, a feat which depends crucially on shape mouldability and allostery[17]. However, typically the operation of those molecular machines proceeds automatically once the expression is initiated. Hence, global three-dimensional shape and allostery concern only the behavior of the molecular machines as devices, independently from the interactions with other devices which determines the regulation. Moreover, where proteins are used by an evolved system for "internal communication purposes" and are therefore "free to choose their own language", they appear to operate in the simpler terms of interactions between linear sequences of nucleotides.

Summing up, in the interactions between genes that take place in GRNs, protein folding appears to play essentially the role of a process that mediates between sequences of nucleotide so as to give the system the possibility of going beyond the mere comparison of identical or complementary sequences in the determination of their reciprocal interaction.[18] This possibility endows the system with several properties that appear to play a central role in its evolvability. One of these properties is the possibility of varying the strength of the interactions with graduality, thanks to the existence of aminoacid substitutions that alter only slightly the shape of the folded protein. This graduality has many favorable evolutionary consequences (Conrad, 1988). For example, it facilitates the detailed exploration of the phenotype space, and it permits the insertion of new devices that have initially a weak inter-

[17] As will be explained below, the same problem is met in defining the connections to what constitutes the "external world", its signals and energy sources for an artificially evolved system. However, for an artificial system we are free to attach an arbitrary identity to those external entities, whereas biological systems must cope with the arbitrary preassigned physical and chemical identity of those entities.

[18] Note, however, that the comparison of identical or complementary sequences is way to determine interactions that is also used by GRN in the form of the mechanism of RNA-mediated regulation described in the next subsection.

action with the existing network and thus have a small initial impact on its performance and are not systematically wiped out by the selection process. On the other hand this graduality does not preclude the complementary possibility of altering radically the strength of the interaction, since simple mutations in the sequence of nucleotides can in some case alter substantially the characteristics of the folded protein (Lewin, 2004).

This interpretation of the role of protein folding means that we can hope to abstract the mechanism and evolvability of gene interaction with mathematical models – which, as explained in the previous section, take the form of maps from pairs of character sequences to values of interaction strength – that do not require the simulation of protein folding. Of course we must devise models with the required characteristics, and this can be far from trivial. But the fundamental point is that we must realize "just" the properties of this interaction that are instrumental to the evolvability of the system, for example those mentioned in the previous paragraph, and we are free to realize those properties in the way that better suits the computational means at our disposal. The carrying out (or the simulation) of protein folding is just a way to implement this interaction; a way which may be the most efficient in a real, physical, biological substrate – that is, with the kind of *computational technology* implemented in cells (Bray, 1995; Conrad, 1999) – but is not necessarily the only way to obtain the required evolvability properties, nor the most efficient if we must count on a radically different computational technology.

2.7.1 RNA-mediated regulation

So far the analysis has been focused on the role of *proteins* in gene regulation. As described in Section 2.5, proteins can be thought as obtained through a process that involves the successive translation and reorganization of a sequence of characters from the genetic alphabet into a series of intermediate sequences from different alphabets (RNA alphabet, amino acid alphabet), followed by a process of protein folding. Since the product of this last step can no longer be considered merely a sequence of characters, it was necessary to present and comment at length on the evidence authorizing the interpretation of gene regulation via regulatory proteins as an interaction between sequences of characters.

Gene expression, however, does not appear to be controlled only by

regulatory proteins. A growing body of evidence suggests that RNA sequences play a substantial role in the functioning of GRNs (Mattick, 2001; Lewin, 2004; Meister and Tuschl, 2004). Contrary to the case of protein-mediated regulation, RNA-mediated regulation appears to be often based on a direct interaction between pairs of sequences of characters, without the additional complication of protein folding. Therefore, the interpretation advocated above of GRN device interaction in terms of an interaction between sequences of characters extends naturally to the case of RNA-mediated regulation.

2.8 Mapping and mutating for evolvability

Having clarified the reasons that make it conceivable an actual implementation of the auxiliary complexity-growth conditions within an artificial evolutionary system, we can now go back to the issue of their practical realization. In this section we will not try to describe the actual implementation details of such an evolutionary system (the next chapter is devoted to this task) but we will try instead to formulate a series of prescriptions that must be considered in the implementation. As anticipated at the end of Section 2.3, the analysis will proceed from biological facts to the characteristics with which to endow the artificial evolutionary system. Since it cannot be hoped that conditions that ensure the evolvability of a given kind of system apply unchanged to the evolution of another, arbitrary system,[19] this implies that the artificial evolutionary system to which we will apply our observations must have a structure similar to that of a GRN.

From the analysis conducted in Section 2.6 follows that a biological GRN can be seen as network of devices interacting through connections whose strength is determined by the relationship – represented by a device interaction map – between pairs of sequences of nucleotides extracted from the genome. Correspondingly, the individuals of our artificial evolutionary system will have a genome constituted by sequences of characters, and a phenotype that consists of a network of devices – whose nature will be elucidated in Section 2.10 – interacting through connections whose strength will be determined by a device interaction map taking as arguments pairs of sequences extracted from the artificial genome. In the biological context, the genome can be reorganized

[19] This can be considered a corollary of the No Free Lunch theorems (Wolpert and Macready, 1997).

by a collection of genetic operators and, correspondingly, the artificial system will be endowed with a collection of genetic operators defined on the artificial genome.

There are thus two aspects of the artificial evolutionary system for which we would like to give a list of prescriptions: the device interaction map, and the genetic operators. The two aspects are closely related and it is difficult to disentangle them. In other words, the device interaction map must ensure the required properties in the context of the genetic operators, and vice versa. Nonetheless, for the sake of clarity, we will to discuss the two aspects separately.

2.8.1 Prescriptions for the genetic operators

A biological genome is subject to many kinds of mutation and reorganizations (Graur and Li, 2000). At the simplest level, single nucleotides can be inserted, deleted, and substituted. More substantial reorganizations are the deletion of a whole fragment (that is, of a whole substring) of the genome, its duplication, and its transposition. The duplication, in particular, can concern whole chromosomes or the complete genome. Finally, there are operations that combine fragments of genome of different individuals, such as the recombination of chromosomes in the case of sexual reproduction and, more generally, the transfer of genome fragments from one individual to another, not necessarily of the same species.

All these operations can be assumed to play a role in the evolution of biological GRN. This is obviously the case for the operations of mutation of single nucleotides, which realize the smallest alteration that can be done at the level of the sequence of nucleotides and permit, for example, the fine control of gene interaction. The other operations of genome reorganization have also a recognized evolutionary role. For example, an important characteristic of proteins is their being typically composed of several *domains*, each with a specific structure and interaction properties (Alberts et al., 2002). The reorganization of gene fragments by deletion, duplication and transposition can thus lead to the additional phenomenon of reorganization of protein domains. In the same vein, considering simultaneously the sequences constituting a gene and those constituting its associated regulatory regions, some modularity and reuse of structure can be achieved by letting evolution assemble in close proximity groups of genes that participate in the realization of a given phenotypic function (Lawrence and Roth, 1996;

Hurst et al., 2004). In this way, subsequent duplications of a genome fragment that has been reorganized in this way, can lead to the duplication and reuse of that part of the network which realizes that function. Another important consequence of the presence of operators of duplication is the possibility to generate new genes by copying existing genes and then subjecting them to mutations, instead of relying only on the (evolutionarily very improbable) generation from scratch of new genes. Moreover, it is widely recognized that events of fragment, chromosome, and whole genome duplication played an important role in the evolution and complexification of existing organism (Ohno, 1970; Sherman et al., 2004). Finally, the evolutionary importance of the recombination of the genetic material of distinct individuals, be it in the form of the transfer of genetic material (Lawrence and Roth, 1996; Rosewich and Kistler, 2000; Lewin, 2004), or in the form of sexual recombination of chromosomes, is also generally recognized, although – in the case of sexual recombination – without universal consensus on the evolutionary mechanism of action.

From this simplified review of the nature and of the effects on GRN of biological genetic operators, we can now derive a set of prescriptions for the genetic operators of our artificial evolutionary system. These amount to saying that the set genetic operators must include

- Insertion, deletion, and substitution of single nucleotides.
- Duplication, transposition, and deletion of sequences of nucleotides.
- Recombination and transfer of sequences of nucleotides belonging to different individuals of the population.

It is readily seen that this prescription can be easily applied to an artificial genome constituted by one or more sequences of characters, as will be the case for our artificial evolutionary system.

2.8.2 Prescriptions for the device interaction map

In our artificial evolutionary system the device interaction map determines the interaction between devices which corresponds to the interaction between genes in biological GRNs. The artificial implementation of this map must thus possess the evolutionary properties of its biological counterpart. We remind that this map takes as arguments

two sequences of nucleotides and gives the scalar value of interaction strength between them.

As explained in Section 2.7, one of the mechanisms of interaction between genes in biological GRN is mediated by the process of protein folding, a process that adds evolvability by allowing, for example, the gradual change of the interaction strength and the corresponding gradual evolutionary transformation of the network. It must be also noted that many factors – from the existence of several synonymous triplets of nucleotides coding for certain amino acids, to the similarity of the chemical properties of some amino acids – result in a substantially many-to-one nature of the mapping from genes to protein function, and this constributes to a further buffering of the effect of small genome mutations. At the same time, the insertion or deletion of just a few nucleotides of a gene can completely alter the result of its translation into a sequence of amino acids and result in a dramatic change of the corresponding protein. Another important property of the biological device interaction map is its tolerance for the genome reorganization described in the previous subsection. This reorganization is evolutionary acceptable only if the interaction between genes is at least partially independent from the position of the genes in the genome, so that a device with its regulatory regions can be freely moved within the genome.[20]

From the introductory comments and this series of observation follows that the device interaction map must

- Possess a many-to-one nature.
- Permit both gradual changes and major reorganizations of the structure of system and of the strength of the interactions between its parts.
- Be compatible with a genome which has been subjected to the action of an arbitrary sequence of genetic operators.
- Be applicable to pairs of sequences irrespective from their original position in the genome

A further characteristics of the interaction between genes in biological GRNs is the possibility of having several regulatory regions control the

[20] In biological genomes this independence is not absolute, since the position of a gene in the genome can influence its level of expression and its interaction with the rest of the genome (Lewin, 2004).

expression of a single gene, and to have a single gene produce several distinct proteins. In the artificial implementation of the device interaction map, this possibility of having multiple interactions is obtained by default, by simply calculating the strength of the interaction between all pairs of sequences associated with the input and output terminals of the devices. However, if the device interaction map is defined so as to operate globally on the whole length of the sequences associated with the terminals, there is the possibility of interference in the determination of the interaction strength of, for example, several outputs relatively to a given input, and this could have a negative impact on evolvability. This problem can be solved in our artificial system admitting the possibility of having more than one input and more than one output terminals for each device. However, we could also consider the possibility of determining several interactions with distinct fragments of a single sequence. This corresponds to a device interaction map that has the possibility of operating *locally* on the sequences in determining the value of the mapping (Figure 2.3). This property, although not strictly necessary, contributes to the possibility to recombine interacting sequences as if they were domains of proteins and fragments of regulatory regions, a possibility that increases the evolvability of the system (Ptashne and Gann, 1998). Therefore, we require from the device interaction map the additional possibility to

- Determine multiple independent interactions operating locally on distinct fragments of the sequences of nucleotides.

From a practical point of view, the efficient implementation of the evolutionary process on a computer requires that the mapping

- Be characterized by a reasonably low computational complexity.

Finally, following from a more high-level perspective on GRN, the mapping must

- Permit the generation of large enough sets of independent sequences, that is, sequences with low interaction strength.

This last requirement guarantees the possibility of incrementally building networks with many devices without having to worry about the interference of newly added genome sequences with those already present. In general, given a device interaction map, we can expect that interference between sequences to hamper the genetic representation of networks composed of a very large number of weakly connected devices. In

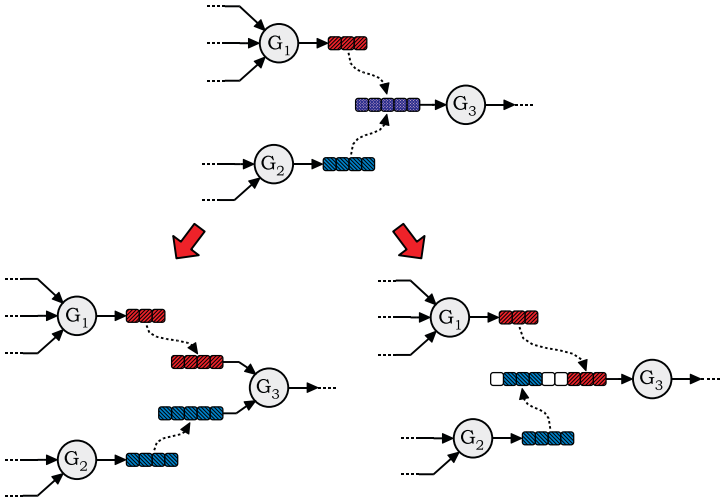


Figure 2.3: *Typically, several genes can influence the activity of a single gene (and a single gene can influence the activity of many distinct genes). In the network representation discussed in the text, this corresponds to the possibility that the outputs of several devices interact with the input of a device (or the possibility that the output of a device interacts with the input of several devices). If the device interaction map operates globally on the whole length of the sequences associated with the terminals, there is the possibility of interference in the determination of the interaction strength (top), and this could have a negative impact on evolvability. This problem can be solved using several input (and output) terminals (bottom, left), or defining a device interaction map that can operate locally on distinct fragments of a single sequence (bottom, right). Of course, the two strategies can be also used simultaneously.*

any case, the interference between genome sequences is a problem also for the GRN of biological organisms. It has indeed been hypothesized that many of the additional regulatory mechanisms existing in higher organisms have been evolved in order to exceed the limits that are imposed on the complexity of bacterial GRN by the relative simplicity of their mechanism of gene interaction (Mattick, 1994, 2001; Mattick and Gagen, 2001; Croft et al., 2003). Note, however, that the limit size of bacterial genomes is estimated at about 10,000 genes (Schuster, 2004), which is already a substantial number of devices for many kinds of

engineered networks, for example an artificial neural network or an analog electronic circuit. Moreover, the system developed in this thesis and described in detail in the next chapter lends itself naturally to the implementation of techniques of reduction of interference. An example of this kind of technique inspired from the biological mechanism of *post-transcriptional gene silencing* (Meister and Tuschl, 2004) will be presented in the next chapter. In any case the example of biological organisms suggests that the final remedy to the problem posed by unwanted interaction does not lie in the adoption of more complicated mappings from pairs of sequences to values of interaction strength, but in the modularization and hierarchical organization of the evolved system (Pattee, 1973).

2.9 Sensing and signaling

So far, the analysis of the properties of biological networks has been focused on the interaction between genes within the boundaries of a single cell. A living cell, however, interacts with the rest of the universe to absorb energy and matter, expel waste, exchange signals. This is true both for unicellular organisms and for multicellular organism. In this latter case, not only the survival but also the correct development of the multicellular structure of the organism depends crucially on this exchange of signals. Similar observations hold for artificial systems, which must be connected to the external world and exchange with it signals and energy in order to perform their function and, possibly, to allow their developing into a multicellular structure. It is thus necessary to specify how our artificial evolutionary system determines and evolves the connections between external devices and the networks defined by the genome, and between multiple copies of these networks. As before, the derivation will proceed from the analysis of the corresponding process at the cellular level of biological organisms to the abstraction of the process of *signalling* that takes place at that level, especially the signalling which influences gene expression.

Cells can detect and communicate to the genome the presence of an external physical or chemical signal (Krauss, 2003; Pires-daSilva and Sommer, 2003; Watson et al., 2003b; Weiss et al., 2003). Without entering in the detail of the process, we can just observe that the original signal almost never influences the level of expression of the genes directly. This is understandable when the signal is originated

in the external environment and cannot be expected to be tailored to the genome structure. Usually, however, this remains true even when the signal is generated by other cells within a multicellular organism. The reason is that there are typically physical constraints that are imposed on the signal in order to propagate in a suitable way from cell to cell. In general, there is therefore a process that mediates between the original signal and the genome. Typically, through a more or less complex chain of events, the original signal influences the concentration or the characteristics of one or more gene regulatory proteins and thus, indirectly, the level of expression of one or more genes.

When the detected signal originates as a gene product in another cell, the resulting interaction between genes can be abstracted just like the interaction between two genes within a cell described in the previous sections. The only difference is that a number of additional steps of signal transmission and transduction are typically added to the process of transcription, translation, protein folding, and gene regulation that characterizes the simpler interaction between genes. Although these additional steps typically depend crucially on allostery and protein shape, this is a consequence of the necessity to adapt the characteristics of the signal to the physico-chemical requirements of the path leading from the source to the target genes. Hence, we can hope to capture the evolvability properties of this communication scheme with a mapping from pairs of sequences of nucleotides to values of interaction strength, just as in the case of the gene-to-gene interaction within GRN abstracted by the device interaction map introduced in Section 2.8. The only novelty lies in the necessity to mark the sequences for "transport" and "exposure" outside the original "cell" constituting the first level of organization of the phenotype of our artificial evolutionary system. This shows that the sequence-based device interaction map described in the previous section in the context of the single level of organization of the network-like systems can be applied also to the study of compartmentalized and hierarchically structured evolutionary systems.

Going now back to the case of a signal originated by the environment external to the evolving system, it is clear that in this case the signal can hardly be conceived as originating from a sequence of characters. Nonetheless, the process that permits such a signal to influence the level of expression of the genes can be interpreted as an adaptation of the nature of these preassigned environmental signals to the sequence-like nature of the regulatory regions of the genes. In the case

of an artificial system, we can obtain the same result by associating a sequence-like identity to the existing input and output signals required by the external sensors, actuators, loads, and by any other preassigned device. In this way, the use of a sequence-based device interaction map can be extended to the evolution of the interactions between these external devices and the evolved network-like system described in Section 2.8. Note that, just like evolution permits the selection of the set of external signals to which a biological cell (or multicellular system) will respond, this technique permits to the artificial evolutionary process the selection of the set of external signals and "features" to which the evolved system is sensitive.[21]

2.10 Devices, networks, and simulators

The description given in Section 2.6 shows that GRNs can be interpreted abstractly in terms of networks of devices. In this interpretation the RNA polymerase plays the role of the device, and the links of the network are determined by the relationship between the sequences of nucleotides that constitute the regulatory regions and the sequence that constitutes the gene. As the final step toward the determination of the properties of our artificial evolutionary system, it is now necessary to analyze the properties of these devices that are relevant to our effort.

Many functions within living cells are performed by proteins in their role of *enzymes*, that is, of protein catalysts (Alberts et al., 2002; Creighton, 1993; Watson et al., 2003b). In the simplest scenario of enzyme operation, a chemical substance generically called the substrate must be converted into another substance called the product. The free energy of the substrate is higher than that of the product. Hence, the former would convert spontaneously into the latter. However, the conversion requires the passage through a less favorable transition state. In the absence of the enzyme, the barrier constituted by the transition state keeps the reaction rate low. The effect of the enzyme is to lower the barrier and thus accelerate the reaction rate. In other words, enzymes permit the variation of the rate of a physical process. They are therefore the key to the implementation of constraints to the spontaneous

[21] An example of this phenomenon can be observed in the example of network shown in Figure 4.44 on page 184, evolved in the context of Experiment 14, Chapter 4. In this case, evolution connected the evolved neuron shown in Figure 4.44 only with a subset subset of the available external signals, which are shown in Figure 4.42 on page 180.

dynamics of those physical processes. As argued by Pattee (1995a), natural selection leads indeed to the formation of structures whose presence influences the dynamics of the surrounding space-time in ways that favor the persistence and, eventually, the self-reproduction of these structures. The "devices" of GRNs – the RNA polymerases – are precisely an example of such evolved structures. We could thus mimic closely also this aspect of GRN by adopting for our artificial evolutionary system, devices capable to influence the rate of a physical process.

A good example of devices with this property is constituted by active electronic devices such as transistors. Their principle of operation is conceptually similar to that of enzymes. A bipolar junction transistor (BJT) is composed of three adjacent regions of semiconductor having different physical characteristics (Cooke, 1990). These regions are called the emitter, the base, and the collector. In the typical circuit configuration, the voltages applied to emitter and collector make the flowing of current carriers from emitter to collector energetically favorable. This current, however, must pass through the base. When the base is left unconnected, it acts as a barrier to the current flow, which is therefore small. A suitable voltage applied to the base lowers this barrier with the effect of increasing the current flowing from emitter to collector. This description reveals a striking analogy in the operation of enzymes and transistors (Figure 2.4). We can say that evolution designed the basic devices of life just as engineers designed the basic devices of electronics. The fact that in biological cells there are many instances of only a few types of RNA polymerases just like in an electronic circuit there are typically many instances of only a few types of active electronic device, corroborates this correspondence.

Of course, we can adopt for our artificial evolutionary system other kinds of devices besides active electronic devices. For example, we could imagine artificial neuron models, or dynamical models mimicking the action of gene expression. There seem to be reasons, however, to avoid devices that implement a static input-output relation (Harvey, 1995), such as, for example, classical artificial neuron models (Haykin, 1999), in favor of dynamical models like those used in continuous-time recurrent neural networks (CTRNN) (Beer, 1995; Vohradský, 2001). The reason is still related to the necessity of considering the working of evolution as devising ways to constrain a rich, realistic dynamics (Pattee, 1973; Polanyi, 1968). From this point of view, an additional benefit of adopting electronic devices as devices of our artificial evolu-

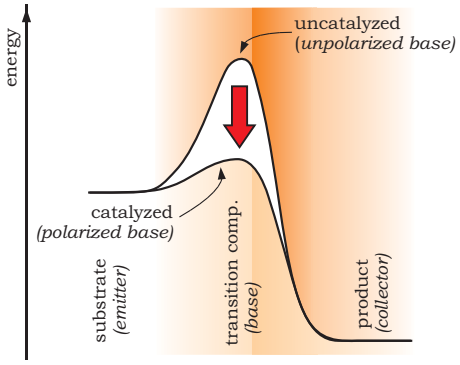


Figure 2.4: *The analogy between the stages of a chemical reaction, and the regions of a bipolar transistor. The vertical axis represents the free energy of the substances (substrate, transition compound, product) in the course of the chemical reaction or the energy of the current carriers in different regions (emitter, base, collector) of the transistors body. The presence of the enzyme (i.e., of the protein catalyst) decreases the height of the barrier that hinders the transformation of the substrate into the products, just as a suitable polarization of the transistors base decreases the barrier that hinders the flow of current carriers from the emitter to the collector.*

tionary system is that we can take advantage of the existence of analog electronic circuits simulators (Vladimirescu, 1994). In these simulators, the physics of the devices and of their interaction is modeled at high level, through a set of algebraic and differential equations, which embed the relevant laws of physics (and, in particular, conservation laws). The resulting implementation is efficient and physically sound. Moreover, it can reproduce the dynamics of chemical processes characteristic of biological GRN (Bhalla, 2003, p. 56; Hiratsuka et al., 1999; Simpson et al., 2003, 2004a,b). Finally, by using the models of energy storing components such as capacitors, this approach allows the implementation of different time scales in the dynamics, and the modeling of delays to the propagation of signals, a phenomenon that affects also chemical signals that must diffuse across spatially extended structures in cells.

The interpretation of the RNA polymerase as the device of the GRNs suggests also the way to encode the artificial devices in the artificial

genome. The existence in the genome of a gene is "signaled" to the RNA polymerase by the presence of the promoter at the start of the gene. The regions of the promoter that are recognized by the RNA polymerase belong typically to a few stereotyped structures. In other words, the promoter acts as a *token* for the device. Correspondingly, we can insert in the artificial genome a token for the artificial device. Note that the RNA polymerase is also encoded in the biological genome, and thus subjected to evolution. We can do the same in our artificial evolutionary system, although in many case the encoding of some device's parameter besides the sequences that determine the connectivity is sufficient to leave some room to the evolution of the devices.

2.11 Outline of an evolutionary system

Assembling all the arguments exposed in the previous sections it is now possible to formulate a set of prescriptions for the synthesis of an artificial evolutionary system bringing together von Neumann's complexity-growth conditions and the auxiliary bio-inspired evolvability conditions. In this evolutionary system:

- Individuals have a genotype (genome) distinct from the phenotype.
- The genome is a collection of sequences of characters belonging to a finite alphabet.
- The genome can be mutated and reorganized through the action of a set of genetic operators complying with the conditions listed in Section 2.8.
- The phenotype is a network of devices connected through links characterized by a scalar strength.
- The devices correspond to structures that act as constraints for a non-trivial and physically consistent actual or simulated dynamics.
- The evolvable properties of the devices, if any, are represented in the genome as sequences of characters.
- The genome contains sequences of characters that are extracted and associated with the terminals of the devices.
- Sequences of characters are associated also with the terminals of preassigned external devices.

- The presence in the genome of a fragment representing a device, the presence of a sequence of characters that must be associated to the terminals of a device, and the presence of a sequence of characters that represents a device parameter, is signaled through a series of specific tokens, that is, of sequences of characters specific to the device, to the terminals, or to the device properties.

- The strength of the interaction between two terminals is given by a device interaction map complying with the conditions listed in Section 2.8, which transforms pairs of labels attached to the terminals into scalar values.

The phenotypes prescribed by these guidelines – that is, networks of devices linked by connections characterized by a scalar value of interaction strength – will be called *analog networks.* Besides analog electronic circuits, neural networks (Haykin, 1999) and, of course, GRN, can be considered analog networks and can therefore be assumed as the object of an evolutionary process following these guidelines. The guidelines expressly state that the interaction between the devices represented in the genome is defined *implicitly* through the device interaction map, instead of being explicitly represented in the genome. As emphasized by Lones and Tyrrell (2004a) this feature is the rule in biological systems and is fundamental in preventing the loss of meaning of the genome following its reorganizations.

Note that usually in evolutionary computation (EC) the genotype-to-phenotype map is related to the concept of *genetic representation* (Bäck et al., 2000a; Rothlauf, 2002; Rothlauf and Goldberg, 2003). The word "representation" evokes a priority of the phenotypic structure over the genotype. The idea is that the problem defines a set of structures that constitute a possible solution to the problem, and that in order to apply an evolutionary approach we need to devise a genotype corresponding to these structures interpreted as phenotypes (Eiben and Smith, 2003, p. 18). In this sense, it is said that a genotype *encodes* a phenotype and that a genotype is *decoded* into a phenotype. Contrary to this perspective, the derivation of the evolutionary system guidelines listed above did not proceed from a predefined phenotype, that is, there were no predefined structures waiting to be represented genetically. Rather, the phenotypic structure and the corresponding "genetic representation" followed as a consequence of the formulation of the complexity-growth and evolvability conditions. This is the reason why the term "representation" was not used in the first place. Nevertheless, this term will also

be used from now on to refer to the genotype-to-phenotype map.

Of course, given the atypical path that led to the phenotype structures, it is necessary to ascertain that they comply with the original goal, that is, if they can act as a foundation for the synthesis of autonomous systems. Since biological GRNs act not only as control and information processing systems of single cells, but produce also the dynamics on which to build more structured systems, it is possible to answer in the affirmative to this crucial question.

Note that the guidelines for an artificial evolutionary system listed above are just heuristic conditions, whose adoption is suggested by the analysis that led to them. We could dispense with some of them, typically at the risk of jeopardizing the evolutionary properties that they bring to biological organisms. For example, we could renounce the existence of a realistic dynamics constrained by the devices, probably at the risk of not being able to open the system to the emergence of further levels of organization. Since electronic devices and easily available analog circuit simulators provide this realistic dynamic, this thesis will mostly show examples of evolution of analog electronic circuits.

Note also that many aspects of the biological processes were intentionally disregarded in the analysis that led to the formulation of those guidelines. An incomplete list of missing features includes the interaction of the dynamics of the expressed GRN with the genome, the existence of non-local control of gene expression through the activation and inactivation of fragments or of whole regions of the genome, and many others. Other aspects, such as the importance of developmental processes, modularity, and hierarchical organization for the achievement of evolutionary open-endedness have been just mentioned.

Finally, no mention has been made so far of the evolutionary algorithm that must govern the artificial evolutionary system built according to the guidelines listed above. The reason for this is that a standard genetic algorithm will be used as the evolutionary algorithm. The only novelty in this respect is the inclusion of the non-standard genetic operators described in Section 2.8, whose implementation details will be discussed in Section 3.6. Since this kind of algorithm is extensively described in textbooks (Bäck, 1996; Bäck et al., 2000a,b; Eiben and Smith, 2003; Fogel, 2000; Goldberg, 1989; Michalewicz, 1996), there is no need to comment further on this topic at this point. Nonetheless, the limitations of current artificial evolutionary algorithms is certainly not a secondary issue – it would deserve, in fact, a separate thesis – and will be discussed further at the end of Chapter 4.

2.12 Related work

The guidelines listed in the previous section identify as the object of this thesis the definition of an evolutionary system for *analog networks*, that is, an evolutionary system for networks of devices linked by connections characterized by a scalar value of interaction strength. As said above, at least the following three kinds of systems fall into this class of networks: GRN; analog electronic circuits; neural networks. There is a vast literature dealing with the modelling and evolution of this large collection of systems, and of systems including them as subsystems. Here, the review will be limited almost exclusively to works dealing with the artificial evolution of analog networks using systems that follow at least in part the guidelines listed in the previous section, giving priority to those focused on the evolution of autonomous systems. The examples will be grouped according to the kind of analog network considered.

2.12.1 Genetic regulatory networks

Circular proteins. A first example of application of GRN concepts to the evolutionary design of control architectures for autonomous robots, is due to Jakobi (2003)[22]. In this work the artificial GRN controls the development of a neural network, which is then used as a control system for an autonomous robot. In this sense, this work is typical of the majority of applications of GRN concepts to artificial evolution, where it is not directly the functionality of the GRN that is evaluated, but rather the functionality or the morphology of a further structure obtained via a developmental process based on the GRN dynamics.

In Jakobi's system, the genome is composed of a single string of characters belonging to an alphabet of four letters. The genome contains genes of fixed length whose starting point is identified by fixed, predefined tokens. The coding region of the gene is preceded by regions of fixed length defining the type of the protein, a threshold, and a link template having a fixed length of three characters which plays the role of regulator binding site. The protein corresponding to a gene is obtained translating the gene into a 64 letter alphabet and joining the ends of the resulting string to create a circle. The interaction between genes is obtained through a complicated process of multiple template matching involving the proteins, the link templates, and a further

[22] This paper reports work realized in 1994.

collection of predefined templates (Jakobi, 2003, p. 396). The evolutionary process proceeds as follows. The genome of each individual of the evolving population is attributed to a single cell located in a two-dimensional space where a number of predefined protein sources is present. The rules of interaction of genes and proteins within and between cells then lead to the development of a neural network, whose performance in controlling an autonomous robot determines the fitness of the individual.

In this system, the GRN interaction scheme was explicitly devised with the aim of obtaining open-endedness, robustness relative to the action of a wide variety of genetic operators capable of determining both gradual and abrupt changes in network properties, and the possibility to genetically encode reusable modular structures. The results eventually obtained were not judged satisfactory by the author, who observed that the effects of the action of the genetic operators on the developed network had the tendency to be either absent or catastrophic in terms of functionality (Jakobi, 2003, p. 402). Considering that the network is determined by the GRN through the mediation of a complicated developmental process, this unsatisfying behavior is not necessarily due to the GRN implementation. As observed by Dellaert and Beer (1996): "development completely transforms the structure of the space that is being searched. If we're lucky, the transformation will allow us to evolve interesting agents more easily. But if we're unlucky, we could actually make the search problem *harder*." In any case, the template matching based technique of definition of the device interaction map used by Jakobi appears a bit convoluted and interference-prone, due to the use of only a handful of characters for the matching.

Sequence matching. The work of Reil (1999, 2003) and that of Geard and Wiles (2003) (the latter taking inspiration from the former) are not primarily concerned with artificial evolution, but rather with the effort of defining models of GRN that are at the same time simple and capable to display a biologically plausible dynamics. It is worth examining them here because the interaction map is based in both cases on the comparison of sequences of characters extracted from the genome.

In both cases the genome is a sequence of characters from a finite alphabet. The presence of a gene in the genome is signaled by the presence of a "promoter", that is, a predefined sequence – a token – of fixed length. A gene is constituted by a fixed, predefined number of characters following a promoter. In (Reil, 1999), an activated gene pro-

duces another sequence of characters which is a transliterated version of the gene. The new sequence is interpreted as a "regulatory protein". If a regulatory protein matches exactly a sequence of characters in the genome, it regulates the expression of the gene that immediately follows the matched genome sequence. The regulation can be positive or negative, and is of the on-off type, with repression prevailing over activation. Geard and Wiles (2003) refined this model by complexifying the map that transforms a gene into a regulator protein, adding a further level of regulation which mimics the action of small RNA regulation in real genomes and defining regulation in terms of a weighted sum of regulator proteins effects.

Implementing and running this kind of artificial GRNs produced behaviors featuring many properties observed in real GRNs. Reil's original experiments were not aimed at the assessment of the evolutionary properties of the system. More recent experiments (Watson et al., 2003a; Hallinan and Wiles, 2004) provide only limited information on the actual evolutionary potential of the system for complex control tasks. We can however still consider the nature of the device interaction maps in the light of the requirements listed in Section 2.8. In the case of (Reil, 1999), the interaction map is defined in terms of a simple transliteration followed by exact sequence matching and on-off activation. Hence, the result can be hardly expected to permit gradual evolutionary changes of the GRN properties, and is not a good candidate for the implementation of a system complying with those requirements. Note, however, that Bongard (2002) reported interesting results from the application of a GRN model similar to Reil's to the development of the morphology and of the neural networks controlling evolved simulated autonomous agents. In any case, the more complicated mapping defined in (Geard and Wiles, 2003) seems to adhere more closely to the principles listed in Section 2.8.

Quick et al. (2003) is an example of the use of GRNs directly as evolvable control systems for robots. The authors emphasize the fact that in their system the structure of the genome continues to determine the nature of the interaction of the phenotype with the environment, whereas in in most experiments of artificial evolution based on GRN concepts the genome "becomes totally redundant, having no further role to play in the structure or behaviour of the resultant artificial organism." In this system the number of genes and proteins is fixed, and each gene has the same number of regulatory regions, which can result either in positive or negative regulation of gene expression. The

interaction between genes is obtained through template matching of proteins and regulatory regions – which are both binary sequences of the same fixed length – weighted by a global evolvable parameter specifying the proportion of available matching proteins that will bind to the regulatory region.

Banzhaf (2003) devised another approach to the of use of sequence matching for the definition of the interaction between the genes within an artificial genome. In his system the genes are binary strings of fixed length, and the mapping from "genes" to "proteins" is not a simple transliteration but a many-to-one transformation that associates a single protein bit with several gene bits using a majority rule. The interaction of the resulting protein with the regulatory regions of the genome is implemented applying first a XOR operation to compute the number of bits that are complementary in the genome and protein, and then transforming exponentially the resulting number into a real value of interaction strength between the gene that has produced the protein and the gene to which the regulatory region belongs. The resulting interaction mapping is characterized by a high redundancy and – according to the preliminary experiments reported in (Banzhaf, 2003) – the whole system appears to possess good evolvability properties.

Morphogenesis. Eggenberger (1996, 1997b, 2003, 2004) pioneered the application of artificial GRNs to the evolution of morphologies, and later extended the approach to the evolution of neural networks (Eggenberger, 1997a; Eggenberger et al., 2002). In his system, the genome is constituted by a sequence of digits, and is subdivided in a predefined number of genes of fixed length. The interaction between genes is based on the possibility to interpret suitably defined substrings of the string constituting a gene as integers, and to operate arithmetically on them. The resulting system is undoubtedly endowed with the possibility of both fine tuning and changing radically the strength of the interaction. With its fixed genome structure, however, this approach negates the fundamental requirements for evolutionary complexity-growth expounded above, even if the presence of a developmental phase coupled with a rich simulated physical dynamics permits in any case the production of interesting evolutionary results (Eggenberger, 2003).

Dellaert and Beer (1996) have defined an evolutionary system that uses a simple GRN model with gene interaction based on boolean functions to develop and control agent morphologies. The operation of the GRN leads first to the development of the morphology of the agent, and

then, on top of it, of a neural structure controlling its behavior. Clearly, a boolean function is a radical simplification of the process of gene interaction in biological GRN described in Section 2.6.

Other interesting results in the field of evolutionary morphogenesis based on artificial GRN have been reported by Kumar and Bentley (2003). Their system is based on the use of two genomes, one encoding real-valued parameters as floating point numbers, and the other encoding the GRN genes and their regulatory regions. The interaction between genes, however, is based on a simple matching of symbols associated with genes and symbols associated with the regulatory regions.

Fractal Proteins. An original approach to the definition of gene interaction in artificial GRNs is constituted by Bentley's work on fractal proteins (Bentley, 2003, 2004). In this work, the genome is constituted by a collection of genes having a fixed, predefined structure. Each gene contains a few parameters defining thresholds and gene type, and two triplets of rational numbers, one corresponding to the promoter for the gene, and one corresponding to the coding region of the gene. In the interpretation of gene interaction given in Section 2.6, the two triples correspond to the input and the output of the gene considered as a device. Each triple of numbers defines a square subset of the Mandelbrot fractal set.

The genome is thought as existing in the cytoplasm of a virtual cell. When a gene is activated, its "output triple", that is, the triple corresponding to the coding region of the gene, releases a "fractal protein" into the cytoplasm. The activation or repression of a gene is determined by the similarity between the fractal proteins existing in the cytoplasm and the fractal set defined by the "input triple" of the gene. The purpose of defining interaction using this technique is very similar to the ideas that inspire the requirements for the device interaction map listed in Section 2.8, namely, to obtain a mathematical model for gene interaction with reasonable computational complexity and still rich enough to display some of the evolutionary potential of the interaction mediated by protein folding. In particular, the fractal device interaction map is redundant and permits both gradual and abrupt changes in the strength of the interaction following the action of the genetic operators.

The experiments reported in Bentley (2004) witness the possibility of generating small sets of proteins with low interaction, and to evolve GRNs capable to generate predefined temporal patterns. A limitation

of this work, relatively to the guidelines listed in Section 2.11, appears the very simple dynamics of the implemented "devices".

Artificial chemistry An interesting model of gene expression and regulation was devised and applied to the evolution of artificial organism by Kennedy and Osborn (2000, 2001). In this model, the genome is a string of bits and the presence of genes and regulatory regions is signaled by predefined tokens. Active genes produce sequences of characters interpreted as proteins through the mediation of virtual molecules called "spiders", which play a role summarizing that of the RNA polymerase and of the translation machinery in real cells.

The interaction between genes is defined in terms of probability of a gene product binding as regulator protein to a regulator binding site. This probability is obtained comparing the sequence of characters that constitutes the protein with the sequence constituting the regulator binding site. This comparison is intended to mimic the binding interaction between the molecules represented by the two sequences. In this sense, the approach can be considered as defining a kind of artificial chemistry inspired by the real chemistry of cells. In practice, the comparison of two interacting sequences is implemented as a count of the number of possible ways the protein can "bind" to the regulator sequence. The value obtained is converted into a binding probability via the action of an exponential map. The resulting device interaction map appears to comply with most of the requirements listed in Section 2.8 (the possibility to generate large sets of independent sequences remains to be investigated). To reinforce the artificial chemistry flavor of this approach, and its correspondence with the precepts of Section 2.11, gene expression is implemented by integrating a set of differential equations which depend on the strength of the interaction between genes, thus making of the "devices" actors of reasonably complex dynamics.

The evolutionary experiments reported in (Kennedy and Osborn, 2001) use a genome with fixed length, but there seem to be no obstacle to the definition of operators changing the length of the genome. The objective of the experiments was the evolution of cells able to exist in a simple virtual environment, and the results suggest a promising evolutionary potential for the system.

While the system defined by Kennedy and Osborn is not intended as a definition of an artificial chemistry, Ziegler and Banzhaf (2001) present an interesting example of evolution of a full-fledged artificial chemistry used as control system. However, the genome of the system

is used only to define and evolve the properties of the artificial chemistry, without the intervention of any GRN concepts.

2.12.2 Analog electronic circuits

Genetic programming. The most impressive results to date in the evolutionary synthesis of analog electronic circuits have been obtained by Koza and co-workers (Koza et al., 1999, 2003). This work is based on the application of the evolutionary technique known as genetic programming (GP) (Koza, 1992; Langdon and Poli, 2002). GP uses a tree-based genetic description which allows the representation and evolution of all kinds of analog network, including analog electronic circuits and genetic, metabolic, and neural networks. In the case of analog electronic circuits, the tree that represents the genome of an individual contains the instructions to "develop" the circuit starting from a preassigned initial circuit. To determine the analog electronic circuit (phenotype) corresponding to a tree (genotype), the tree is parsed and the resulting instructions are applied sequentially to to the developing circuit. The definition of the genetic operators must ensure that mutated and reorganized trees represent legal sequences of developmental instructions.

Clearly, the GP approach does not take inspiration from the principles that led to the complexity-growth and evolvability requirements described in Section 2.8 and Section 2.11. The virtually complete absence of fitness graphs from the published reports makes it difficult to estimate the characteristics of the evolutionary process leading to functional circuits. Some of the author's comments, along with the fact that the best-of-run circuits are obtained typically after only a few dozen generations and using huge populations point to an evolutionary scenario where the results are obtained through recombination of building blocks that in good part are already present in the initial population. The outcome of the experiments described in (Koza et al., 1999, 2003) testify that this approach to artificial evolution (advocated also in (Goldberg, 2002)) is certainly capable of producing interesting results. However, it appears essentially an engineering search strategy loosely related to the process of biological evolution. In any case, given the state-of-the-art status of this approach to the evolutionary synthesis of analog electronic circuits, many problems and results described in (Koza et al., 1999, 2003) will be used as benchmarks and references for the results obtained with the approach developed in this thesis.

Genetic algorithms Genetic algorithms (GA) (Goldberg, 1989) are another class of evolutionary algorithms that has been used extensively for the synthesis of analog electronic circuits (Zebulum et al., 2002). The genome used by GAs is typically a sequence of characters, thus, the first step for the application of GAs to this field is devising a suitable sequence representation for analog circuits (Zinchenko et al., 2003). If the topology of the circuit is predefined and fixed, the representation concerns only the component values and the problem is trivial. This case, however, does not concern us here, since the GA operates merely as an optimization algorithm and no evolutionary complexity-growth or open-endedness can be envisaged.

A simple encoding scheme that permits the representation and evolution of both the circuit connectivity and the component values was proposed by Grimbleby (1995)[23]. In this representation, each component is represented by a gene, which specifies the type of component, its value, and the nodes to which its terminals are connected. This encoding allows the representation of arbitrary circuits, although randomly assigned genomes would typically produce dangling nodes, unconnected components, and multiply connected circuit graphs. Hence, it is advisable to implement some curing of the decoded circuit, or some additional encoding rules encouraging the production of circuits with a reasonable topology (Kruiskamp and Leenaerts, 1995). Although interesting result were obtained using this representation (Grimbleby, 1995; Zebulum et al., 2000, 2002), the possibility of implementing gradual changes and, more generally, the compliance with the principles listed in Section 2.11, is scarce. Similar considerations apply to the representations discussed in (Zinchenko et al., 2003).

Lohn and Colombano (1999) present an alternative approach to the use of character sequences for the representation of analog circuits. Their approach is explicitly aimed at obtaining the possibility of evolving both the component values and the topology, while keeping low the computational complexity of the decoding and ensuring a non catastrophic effect of the genetic operators. The genome is structured as a list of instructions that are applied to an predefined initial circuit and produce the actual phenotype. This representation is obviously very similar to the one used by GP, and the remarks made for GP apply to it.

[23] In Grimbleby's original paper the component values were determined separately, but the representation permits their evolution.

Enzyme Genetic Programming. Strictly speaking, Enzyme Genetic Programming (EGP) (Lones, 2004; Lones and Tyrrell, 2004a,b) does not belong to the class of methods specifically devoted to the synthesis of analog electronic circuits, since the examples of application of this method concern the synthesis of digital electronic circuits. Nonetheless, the approach that led to the development of EGP has many points in common with the approach advocated in the present work. First, there is at the roots of EGP an explicit concern for the issue of evolvability; second, the working of biological systems at the cellular level are expressly taken as inspiration for the development of EGP; third, the EGP genome contains the description of "devices" (implementing boolean functions) and of "shapes" that can be though of as associated with the terminals of the devices to determine the strength of the interaction between the devices. There are, however, major differences between EGP and the system described in this thesis: in EGP the circuit decoded form the genome is obtained through a process of "development" derived form genetic programming, and the interaction between terminals is given in terms of presence or absence of connection and not as a range of interaction strength values. Moreover, the descriptors of the device and of the shapes are not represented homogeneously in the genome as sequence of characters, and the interaction strength is not determined by a mapping from pairs of sequences extracted from the genome. As a consequence, EGP as originally formulated is not suited to the synthesis of analog networks. It remains true, however, that EGP approach appears methodologically very close to the one described in the previous pages, in particular in its emphasizing the importance of implicitly defining the interactions between devices encoded in the genome, in order to increase the reorganizability of the genome.

2.12.3 Neural networks

Direct representation and developmental schemes As pointed out by the review paper by Yao (1999), there are many examples of application of evolutionary algorithms to the synthesis of neural networks (NN). These examples can be roughly divided into two classes: the first using a genome that directly represents the NN (for example, Whitley et al., 1990; Pujol and Poli, 1998; Kobayashi and Ohbayashi, 1999; Stanley and Miikkulainen, 2002, 2004) and the second using the genome to define a developmental process leading to the NN (for example, Kitano (1990); Belew (1993); Cangelosi et al. (1994); Gruau

(1994, 1995a,b); Nolfi and Parisi (1995); Eggenberger (1997a); Astor and Adami (2000); Eggenberger et al. (2002)).

The examples falling into the first class that go beyond simple weight optimization, resort typically to a genetic representation where each artificial neuron correspond to a gene. This gene specifies the type of neuron, its parameters – if any –, and the other neurons of the network to which the neuron represented by the gene is connected. This representation is clearly analogous to that suggested by Grimbleby (1995) for analog electronic circuits and examined in Section 2.12.2, and the observations made for that representation therein apply to it.

The examples that fall into the second class, on the other hand, obtain a NN only through the mediations of a developmental process and tend to suffer from the problem signaled by Jakobi (2003), and discussed in Section 2.12.1, namely, the difficulty of relating the changes in the genome to the changes in the phenotype and to define genetic operators capable of mediating between the absence of phenotypic effects and the production of catastrophic changes.

In any case, there do not appear to exist examples of evolutionary systems that directly interpret a GRN-like system as a network of neurons (and neither as an analog electronic circuit, for that matter). Therefore, existing examples of evolutionary approaches to NN do not appear related to the spirit of the approach adopted in this thesis. Nonetheless, it is worth examining in more detail two examples that have some feature relevant to our endeavour.

Cellular encoding. Gruau (1994, 1995a,b) has defined a scheme called *cellular encoding* that uses a tree-like genome to define the "development" of a NN. In this sense the approach is similar to the tree-based representation of GP. What is peculiar about Gruau's approach is his effort to define the properties of the encoding scheme. Two of these properties, in particular, resemble two of von Neumann's complexity growth conditions. They are what Gruau has called the property of *closure* and the property of *completeness* of an architecture encoding scheme. According to Gruau, an encoding scheme is closed relatively to a set of architectures if every description can be decoded in an architecture belonging to the set, and is complete relatively to the set if any element of the set can be encoded by the scheme. Gruau shows that cellular encoding has these two properties and, therefore, that it fulfils von Neumann's complexity growth conditions.

NEAT. NEAT is an evolutionary system for NNs developed by Stanley and Miikkulainen (2002, 2004). It uses a representation belonging to the first of the classes identified above, that is, a genome whose genes correspond to the neurons and to theirs connectivity. The peculiarity of NEAT relatively to the other encoding schemes of this class, is its including a series of features to increase evolvability, for example, the presence of genetic markers that allow the implementation of fitness sharing (Eiben and Smith, 2003) and the definition of a genetic operators of neuron creation that tries to minimize the perturbation caused to the function of existing network.

2.12.4 Discussion

This review of existing evolutionary systems for analog networks shows that, as could be expected, the examples most relevant to the approach adopted in this thesis belong to the class of artificial GRN models. In particular, the systems of Kennedy and Osborn (2000), Banzhaf (2003), Bentley (2004), and Lones (2004), comply with some of the guidelines listed in Section 2.11. There seems to be, however, room for improvement with respect to all the systems considered, especially in the use of a more realistic and physically sound dynamics, and in the definition of device interaction maps that are explicitly defined to comply with those guidelines. On the other hand, the approaches used to define evolutionary systems for NNs and analog electronic circuits appear to be based on rationales quite remote from that adopted in this thesis. In addition, none of the systems that belong to this group and which use GRN-inspired concepts explore the possibility of directly interpreting the GRN structure as a NN or as an analog electronic circuit. Nonetheless, the examples of Gruau's definition of the cellular encoding properties, and that of the evolvability-increasing features of NEAT, attest an awareness of the importance of the issues that led to the formulation of the rationale on which this thesis is based, and which will lead to the definition of the details of the evolutionary system. A task that will be carried out in the next chapter.

Chapter 3

An evolutionary framework for analog networks[1]

Overview

In this chapter the guidelines for the definition of an artificial evolutionary system derived in the previous chapter are transformed into the specification of an actual evolutionary system for analog networks. The chapter starts with the exposition of the characteristics and of the structure of the ***artificial genome****, and with the illustration of the basic idea on which the* ***genetic representation*** *rests, namely, the association of sequences of characters extracted from the genome with the terminals of the devices that can appear in the network. This operation is based on the use of tokens to identify the devices, the sequences associated with their terminals, and the sequences associated with the evolvable parameters. The next step is the* ***connection of the devices*** *decoded from the genome. This operation is based on the application of a device interaction map that associates a value of interaction strength with each pair of sequences of characters. The device interaction map is described first in abstract terms and then an actual example of this map is presented, and its properties and implementation are discussed. Then, it is shown how the technique used to define the connection of the devices decoded from the genome can be extended to define the connections between the evolved circuit and the preassigned* ***external devices****, and to allow the evolution of compartmentalized networks. Finally, the ensemble of* ***genetic operators*** *that can be applied to the artificial genome is defined and described.*

[1] Parts of this chapter were published in (Mattiussi and Floreano, 2004b) and (Mattiussi and Floreano, 2004a).

3.1 The artificial genome

3.1.1 The genetic alphabet

Complying with the guidelines derived in the previous chapter, the genome of the artificial evolutionary system defined in this thesis is composed of one or more finite and nonempty sequences of characters.[2] The sequences of characters that compose the genome are also called *chromosome*s, and the characters that compose the chromosomes are also called *nucleotides*. The nucleotides belong to a finite *genetic alphabet* $\mathcal{G}$, whose size is denoted by $|\mathcal{G}|$. The value of $|\mathcal{G}|$ is not defined directly by the guidelines listed in the previous chapter, following only indirectly from them. As will be shown later, this value influences in particular the possibility of generating large sets of independent sequences, which can be analyzed only after the details of the device interaction map have been specified. As this will be done only in Section 3.4, the exposition of the arguments leading to the choice of $|\mathcal{G}|$ is postponed to that section. In the meantime, the uppercase alphabetic characters of the ASCII character set will be assumed without justification as composing the genetic alphabet, and used to illustrate the examples described in the text.

3.1.2 Devices, terminals and parameters

Figure 3.1 shows an example of a genome constituted by a single chromosome. The chromosome is seen here as an unstructured sequence of characters belonging to the genetic alphabet. According to the derivations of the previous chapter, this genome must represent a network of devices, with each device possibly possessing some evolvable parameter.

For each evolutionary experiment the experimenter assigns a *device set* which specifies the kind of devices that can appear in the network. For example, an evolutionary experiment aimed at the synthesis of an analog electronic circuit could have a few types of transistors as the elements of the device set, and an evolutionary experiment aimed at the synthesis of a neural network could have a few types of artificial

[2] The possibility of having an artificial genome composed of several distinct sequences of characters – several chromosomes –, although not essential, can be useful in many respects, for example in facilitating the evolution of compartmentalized systems (Subsection 3.5.3), or in allowing the parallelization of the operations of replication, mutation, and decoding of the whole genome.

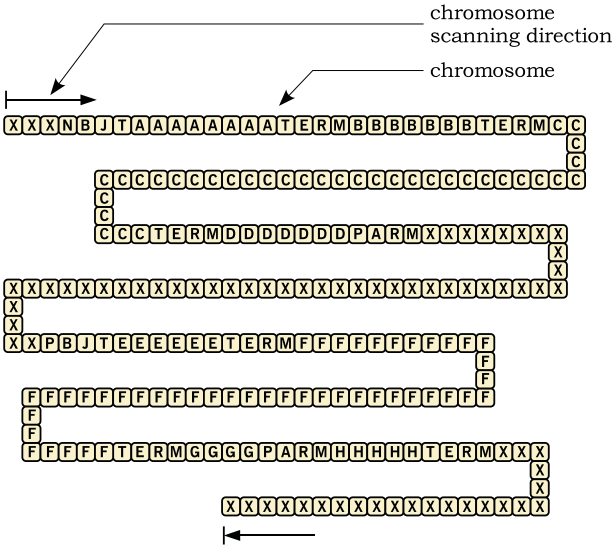


Figure 3.1: *A genome constituted by a single chromosome, that is, a single sequence of characters belonging to the genetic alphabet.*

neuron models as the elements of the device set.

The fundamental idea of the genetic representation proposed in this thesis is the association of sequences of characters extracted from the genome with the terminals and with the evolvable parameters of the devices that will compose the network that is encoded by the genome. To perform this association, a collection of specific sequences of characters that we call *tokens* is defined. The role of the tokens is to signal the presence of fragments of genome associated with the devices of the network, and to delimit the sequences of nucleotides that must be extracted from the genome and associated with the terminals and with the parameters of the devices. To permit the representation of all the devices belonging to the device set, one specific *device token* is associated with each device belonging to the device set. The identification of the sequences that must be extracted from the genome and associated with the terminals and with the parameters is based on the specifica-

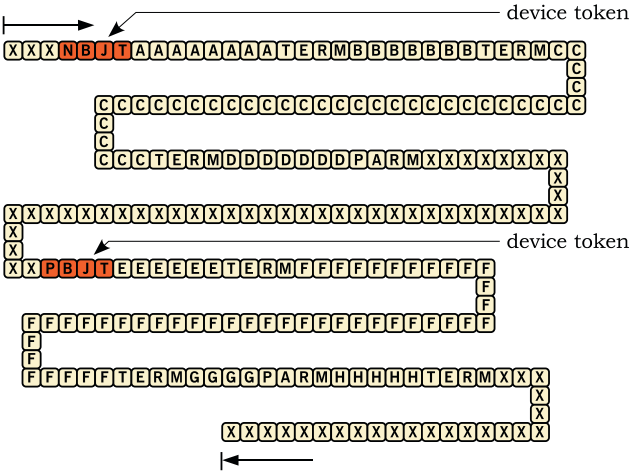


Figure 3.2: *A chromosome with two device tokens (highlighted), one signaling the start of a fragment of genome potentially encoding an NPN BJT, and one signaling the start of a fragment of genome potentially encoding a PNP BJT.*

tion of a *terminal token* and of a *parameter token*[3].

The role of the device tokens is to signal the start of a fragment of genome potentially encoding the corresponding device. Figure 3.2 shows an example for the case of analog electronic circuits where the start of a fragment potentially representing an instance of bipolar junction transistor of type N (NPN BJT) is identified by the device token `NBJT` and the start of a fragment potentially representing an instance of bipolar transistor of type P (PNP BJT) is identified by the device token `PBJT`.

As hinted above, the presence of a device token in the genome is only potentially indicative of the presence of a corresponding device in the decoded network. A device is actually encoded in the genome only if all the sequences of characters that must be associated with the terminals and with the parameters of that device are present in the fragment of genome that follows the device token. The role of the terminal and of

[3]Some additional tokens will later be defined for the connection of the evolved circuit to the external world.

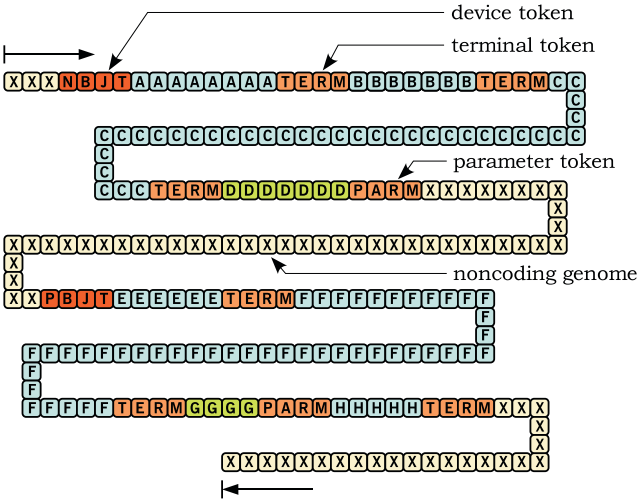


Figure 3.3: *A chromosome encoding two electronic devices: an NPN BJT device and a PNP BJT device, both with an evolvable parameter. The* `NBJT` *and* `PBJT` *device tokens signal the start of the fragments of genome coding for the BJTs. The terminal token* `TERM` *and the parameter token* `PARM` *signal the end of a sequence of characters that is associated with a terminal or with a parameter of the device and, possibly, the start of another sequence associated with a terminal or with a parameter. The fragments that do not correspond to a token or to a sequence associated with a terminal or with a parameter, correspond to non coding fragments of the genome. Apart from tokens, sequences of identical characters for each coding and non coding fragment is used here for illustrative purposes, to facilitate the visual identification of the sequences. In an actual evolved genome these sequences will be typically heterogeneous (see, for example, the evolved genome shown in Figure 4.46 on page 185).*

the parameter tokens is to signal the end of a fragment of genome that is associated with a terminal or with a parameter, respectively, and the start of the successive fragment associated with a terminal or with a parameter, if any. This means that all the terminal and parameter tokens required by that particular device must be present in the corresponding genome fragment. Figure 3.3 shows an example of a chromosome representing two BJTs (which are three-terminal devices). Here,

besides the device tokens NBJT and PBJT introduced above, the token TERM is used as terminal token, and the token PARM is used as device token, and each BJT is assumed to have, besides its standard three terminals, one evolvable parameter. The fragments of genome not corresponding to tokens and not associated with terminals or parameters, correspond simply to *non coding genome*. Note that the use of evocative tokens like NBJT, PBJT, TERM, and PARM in the examples, derives simply from the desire to facilitate the visual inspection of the genome. The use of similar tokens like NBJT and PBJT for similar devices is a choice that lets evolution transform one device into the other with a mutation as simple as a single nucleotide substitution. This facilitates the evolutionary production of circuits that have a given structure but use different devices.

The length of the tokens and the size of the genetic alphabet determine the probability of randomly generating the tokens. With an alphabet of size $|\mathcal{G}|$ the probability p_t of randomly generating a token t of length $|t|$ is $|\mathcal{G}|^{-|t|}$. Although the random generation of a token is perfectly acceptable and can even benefit the evolutionary process, if the value of p_t is too large a randomly generated or randomly mutated genome would be almost surely cluttered with spurious tokens, and this would interfere with the evolutionary process. Given the genetic alphabet and a maximum acceptable value for p_t (typically following from heuristic considerations), a minimal length for the tokens follows. The examples shown in the previous figures use the values $|\mathcal{G}| = 26$ and $|t| = 4$ and correspond to a probability $p_t \approx 2.2 \cdot 10^{-6}$ which ensures that on average very few spurious tokens are generated in a genome with a length of some thousand nucleotides, as is the case for the examples shown in the next chapter.

The number and kind of elements required for the correct decoding of a device varies from device to device. For example – considering for the moment only the terminals – a bipolar transistor always requires the existence of three sequences associated with its three terminals, a capacitor requires the existence of two sequences associated with its two terminals, whereas an artificial neuron could be specified as requiring the existence of two sequences associated with the input and output terminals, respectively, or could instead be specified as requiring the presence of at least one sequence associated with its output and an arbitrary number of sequences associated with a corresponding number of inputs.

The number of sequences associated with device parameters can

also vary from device to device, and depends on how many evolvable parameters a given device is attributed in a given experiment. For example, an experiment could use completely predefined bipolar transistors, in which case no evolvable parameters would be associated with these devices; another experiment could assume as evolvable, say, the transistor's current gain β, in which case there would be one evolvable parameters that would be associated with transistors; still another experiment could use simultaneously both kinds of transistors, one with evolvable parameters, and one without.

Devices for which no evolvable parameters are defined can still have a separate evolvable global description of the device encoded in the genome. The evolution of this global description would influence collectively the characteristics of all the devices of the given kind, contrary to what happens with parameters encoded in the fragment of genome corresponding to a device, which refer only to that particular instance of the device.

When the terminals of a device are not interchangeable, like in the case of the base, collector, and emitter of a transistor, or of the input and output terminal of an artificial neuron, a predefined association order (for example collector→base→emitter, or output→input) is specified. If more than one evolvable parameter is present, a predefined association order is specified also for the evolvable parameters. On the other hand, since their tokens are different, terminals and parameters cannot be confused; hence, they are left free to appear and mix in any order in the fragment of genome that codes for a device. For example, Figure 3.3 shows a fragment of genome coding for an NPN BJT where the evolvable parameter is the last element of the fragment, but the evolvable parameter could be the first element of the fragment, or the third element, like in the case of the PNP BJT encoded in the same chromosome. This freedom to mix terminal and parameter sequences makes the decoder more tolerant to genome reorganizations.

3.2 Device extraction

The representation technique described in the previous section leads to the following procedure for the extraction of the devices encoded in the genome. Each chromosome is scanned in search of any of the device tokens belonging to the device set used in the evolutionary experiment. If one of the device tokens is found, the fragment of genome starting

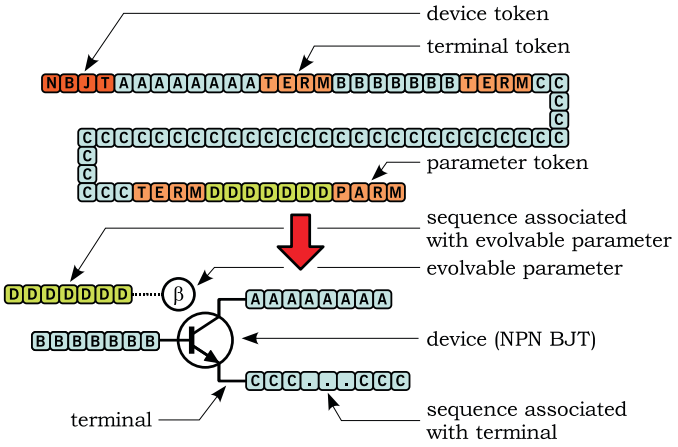


Figure 3.4: *A fragment of the chromosome represented in Figure 3.3, encoding an NPN BJT having one evolvable parameter. The decoding creates a device of the type specified by the device token, and associates a sequence of characters with each terminal and with each evolvable parameter of the device.*

after the token is scanned in search of all the terminal and parameter tokens required by the device. If all the required tokens are found before the next device token or before the end of the chromosome, a device – for the moment, unconnected – is created and the sequences of characters delimited by the tokens are associated with the terminals and parameters of the device. Then, the next device token is searched in the genome, and so on until all the genome has been examined.

Figure 3.4 shows an example of decoding of a transistor from a fragment of genome that contains all the tokens required by an NPN BJT with one evolvable parameter. In this example, the sequence of nucleotides comprised between the end of the device token `NBJT` and the start of the first terminal token `TERM`, is associated with the first terminal of the NPN BJT, the sequence of nucleotides comprised between the end of the first `TERM` and the start of the second `TERM` is associated with the second terminal of the NPN BJT, the sequence of nucleotides comprised between the end of the second `TERM` and the start of the third `TERM` is associated with the third terminal of the NPN BJT, and

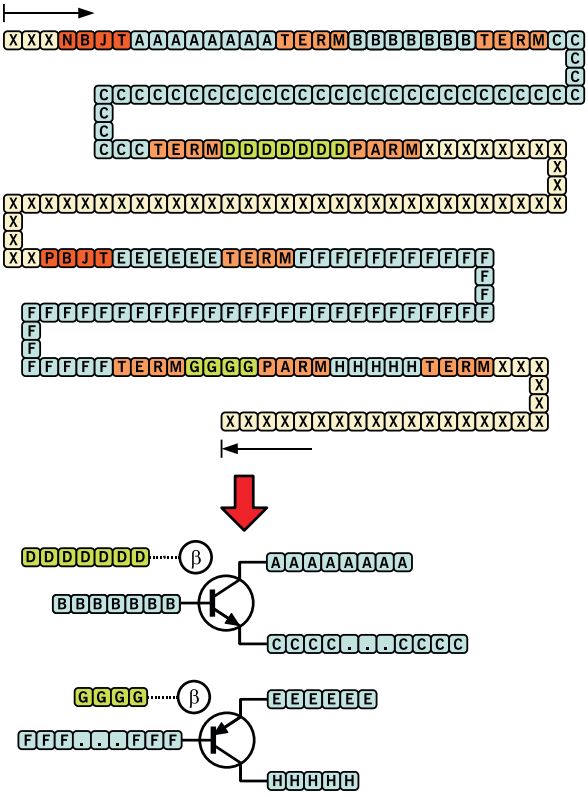


Figure 3.5: *A genome composed of a single chromosome encoding the devices of an analog electronic circuit (top), and the devices extracted from it following the procedure described in the text (bottom).*

the sequence of nucleotides comprised between the end of the third TERM and the start of the PARM token, is associated with the evolvable parameter that the BJT is assumed to possess in the evolutionary experiment, for example, the current gain β. Figure 3.5 shows the result of extracting all the devices of an analog electronic circuit encoded in a genome composed of a single chromosome.

The procedure of component extraction described above does not admit the overlapping of sequences of characters corresponding to different devices, but can be easily modified so as to permit this overlapping. To this end, it is sufficient to stipulate that, once a device token is found, the chromosome is scanned until all the tokens required by a device are found or the chromosome end is reached, irrespective of the presence of another device token. In this case, for example, if one of the TERM tokens following the token NBJT and preceding the token PBJT of the chromosome represented in Figure 3.5 is removed or invalidated, the search of the third TERM token for the NPN BJT produces the first TERM token following the PBJT token, and the whole sequence from the end of the first PARM token to the start of this TERM token is associated with the third terminal of the NPN BJT. Although gene overlapping is known to occur in natural genomes (Graur and Li, 2000), preliminary experiments and the sequence matching experiments described in Chapter 4 suggest that the presence of device overlapping tends to speed the initial phases of evolution but also to generate an interaction between devices that appears to hamper further evolutionary progress. For this reason, in the experiments reported below the possibility of device overlapping is excluded. This corresponds to consider potentially associated with a device only the fragment of genome that goes from the end of the a device token to the start of the next one.

Figure 3.6 shows another example of chromosome encoding of the devices of an analog network. In this case the analog network is a neural network instead of an analog electronic circuit. The devices are now artificial neurons, identified by the token NEUR, while the terminals and the parameters are still identified by the tokens TERM and PARM. Artificial neurons are typically characterized by one output and one or more input terminals. Since – as explained in the previous chapter – the device interaction map must allow the determination of multiple interactions with a single sequence, it could be sufficient to use and encode in the genome only one output terminal and one input terminal for each device. To facilitate the task of evolution, however, we might decide instead to permit the association with the input of each neuron of more than one sequence of characters extracted from the genome. This can be obtained by specifying that the first sequence of characters delimited by a TERM token is associated with the output, and the subsequent ones are associated with the inputs. Figure 3.6 presents an example of this kind of encoding. Here, one of the neuron encoded in the chromosome has a single input sequence whereas

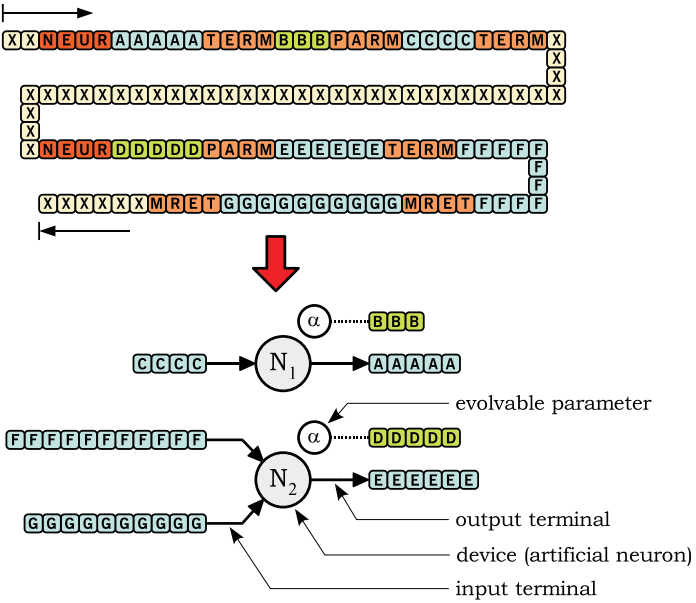


Figure 3.6: *A chromosome that encodes the devices belonging to an artificial neural network, and the two neurons extracted from it. Note that the first neuron has a single input, whereas the second has two inputs. This is an example of encoding that does not impose the presence of a fixed and predefined number of terminals of a certain type. The possibility of having several input terminals permits the definition of densely connected networks without burdening the sequences of characters associated with the terminals with the necessity of generating too many independent interactions. Associated with each neuron there is also an evolvable parameter α, for example, a parameter determining the slope of a sigmoidal activation function.*

the other neuron has two sequences associated with its inputs. Note that the case of no associated input sequences could be also accepted, for example to determine within the network bias neurons with fixed output and no inputs (Haykin, 1999). If a distinction between excitatory and inhibitory neurons is desired, it can be obtained by defining different device models, each with a specific device token, for example

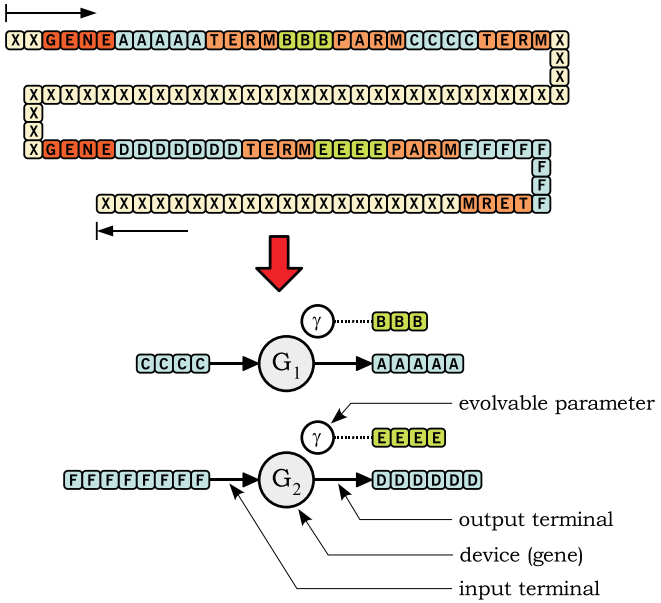


Figure 3.7: *A chromosome that encodes the devices belonging to an artificial genetic regulatory network, and the two genes extracted from it. An evolvable parameter γ is also associated with each gene.*

`NEUE` and `NEUI` for excitatory and inhibitory neurons, respectively. If it is instead required to distinguish excitatory and inhibitory inputs, it is sufficient to use distinct terminal tokens for the output terminal, the excitatory input terminals, and the inhibitory input terminals.

The other kind of analog network that was mentioned in the previous chapter is constituted by genetic regulatory networks (GRN). Figure 3.7 shows how a GRN can be encoded using the approach described above. The devices are now genes, and the encoding corresponds, *mutatis mutandis*, to that of the neurons illustrated in Figure 3.6. Like in the case of the neurons, if the possibility of having more than one input terminal is required, it can be obtained by stipulating that all terminals tokens following the first must be decoded and interpreted as associating a sequence to an input. If several input and output terminals are

required, it is sufficient to define two distinct terminal tokens for them.

3.3 Interaction strength and parameter values

The result of the device extraction process described in the previous section is a collection of unconnected devices that have sequences of characters associated with their terminals and with their evolvable parameters, as illustrated by Figure 3.5, Figure 3.6, and Figure 3.7. To turn this collection of devices into an actual analog network we need to connect the devices and assign actual values to their evolvable parameters.

3.3.1 Connecting the devices

As explained in Section 2.6, the connection of the devices is based on the definition of a *device interaction map*, which transforms pairs of character sequences associated with two distinct device terminals into a scalar value that represents the strength of the direct interaction between the terminals.[4] For example, if the analog network is an electronic circuit, the strength of the direct interaction between two device terminals can be represented as a value g of conductance[5] inserted between the terminals. If we denote with s_1 and s_2 the sequences of characters associated with two distinct terminals, we can denote with $G(s_1, s_2)$ the device interaction map that produces the value of conductance that must be inserted between the two terminals (Figure 3.8). Repeating this process for each pair of distinct sequences, we obtain a value of conductance between each pair of terminals of the devices extracted from the genome. In this way, the collection of originally unconnected devices is transformed into an actual analog network. Note that the device interaction map can associate a null value of conductance

[4] We call *direct interaction* or *direct connection* between two device terminals the interaction that is established externally relatively to the devices, through the action of the device interaction map defined below. For example, in the case of electronic circuits, when the terminals belong to a single device, say a transistor, there is obviously the physical interaction between the terminals mediated by the internal structure of the transistor, but this interaction is not considered a direct connection in the sense specified above. On the contrary, a resistor or a wire inserted between the terminals constitutes a direct connection, and the two terminals are not directly connected when there is no such wire or resistor connecting them.

[5] The electrical conductance g of a conductor (whose SI unit of measurement is the *siemens* - symbol S) is the reciprocal of its electrical resistance r (whose SI unit of measurement is the *ohm* - symbol Ω).

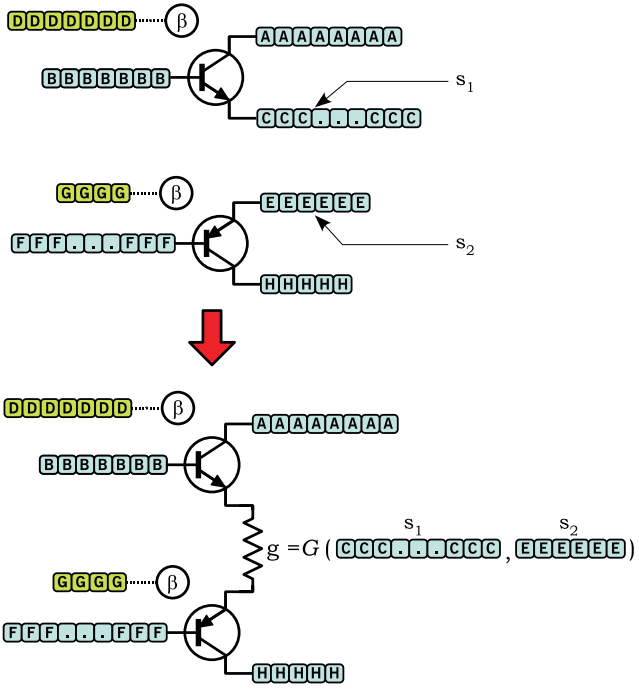


Figure 3.8: *The connection between two device terminals having sequences of characters associated with them is determined by a device interaction map that associates with each pair of sequences a value of interaction strength between the corresponding terminals. In the case of electronic circuits considered here, the device interaction map $G(s_1, s_2)$ produces a value g of conductance that must be inserted between the terminals. Note that the process illustrated in the figure is repeated for all pairs of distinct sequences of characters associated with the terminals, even if – for the sake of clarity – the bottom part of the figure does no longer show the pair of sequences that have been processed to produce g.*

with certain pairs of sequences. This corresponds to the absence of direct interaction between terminals carrying those pairs of sequences, and prevents the need for each device terminal to be necessarily connected to all other device terminals.

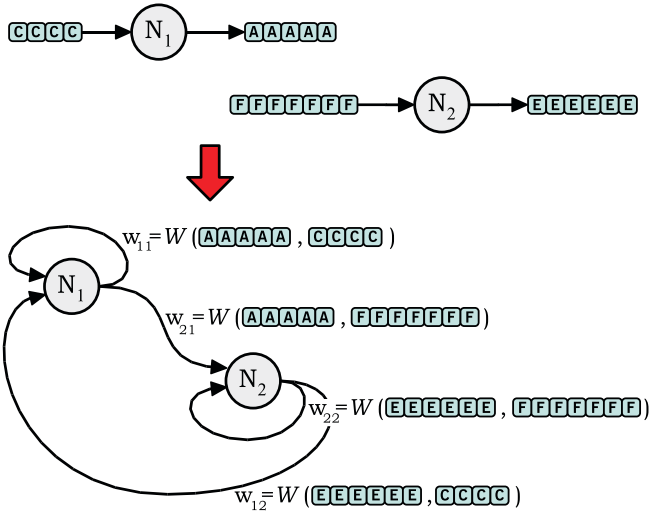


Figure 3.9: *Two artificial neurons extracted from a genome and having sequences of characters associated with their input and output terminals (top) are connected and thus transformed into a neural network (bottom) applying the device interaction map $W(s_1, s_2)$ that associates link weights w with pairs of character sequences. An analogous illustration can be drawn for the genes of the artificial GRN shown in Figure 3.7.*

Contrary to the case of analog electronic circuits, where the terminals of devices such as bipolar transistors do not have a clear characterization as inputs or outputs, in the case of neural networks this characterization exists and it makes sense to consider only the connection of output terminals to input terminals. The value of interaction strength corresponds to what in the terminology of neural networks is called the *weight* w_{ij} of the link connecting the output of the j-th neuron with the input of the i-th neuron (Haykin, 1999). The device interaction map $W(s_1, s_2)$ that determines the connections will therefore associate values of weights with pairs of sequences of characters associated with one output terminal and one input terminal of the neurons decoded from the genome. Figure 3.9 shows the establishment of the connections between two neurons, which transform the neurons

assumed as extracted from a genome, into a neural network (for the moment, without external inputs and outputs). In this case too, the device interaction map might associate null values of weights to certain pairs of sequences,[6] thus giving the possibility of leaving pairs of input and output terminals without a direct connection.

The case of genetic regulatory networks corresponds closely to that of neural networks, and the corresponding remarks and illustrations apply in general also to this kind of analog network. There is typically just a difference in the meaning given to the value of the interaction strength in GRN. Often this value is interpreted as a probability of occupation of a given regulatory site from the part of a regulatory protein. If this is the case, the values of interaction strength must be eventually normalized in order to actually represent probabilities. An alternative approach, which eliminates the problem of proliferation of interactions from the start, is to retain for each gene only the strongest of the interactions potentially determined by all the pairs of sequences associated with the input terminal of that particular gene and with all the output terminals of all the genes present in the network, considering all other potential interactions recessive (Lones and Tyrrell, 2004a).

3.3.2 Assigning parameter values

The assignment of a value to the evolvable parameters is based on the definition of a *parameter map* $P_1(s)$ that transforms the sequence s of characters extracted from the genome and associated with the parameter into the value of the parameter (Figure 3.10).

There are obviously many ways to map a sequence of characters into a real number. For example, the sequence of characters could be interpreted as an integer number written in base $|\mathcal{G}|$, and this integer could be further mapped into an interval corresponding to the range of variation selected for the parameter. Using this approach, however, the nature of the parameter map would possibly be very different from that of the device interaction map described above, which acts instead on pairs of sequences. In particular, this would be true for the consequences of mutations and reorganizations of the genome on the value of interaction strength and parameter values. For this reason, it is

[6] Anticipating some material from Section 3.4, we observe that to facilitate the evolution of sparsely connected networks it is sufficient to define a device interaction map that associates a null value of interaction strength with a large set of pairs of sequences, i.e., with an entire *range* of sequence alignment scores rather than with a single value of alignment score.

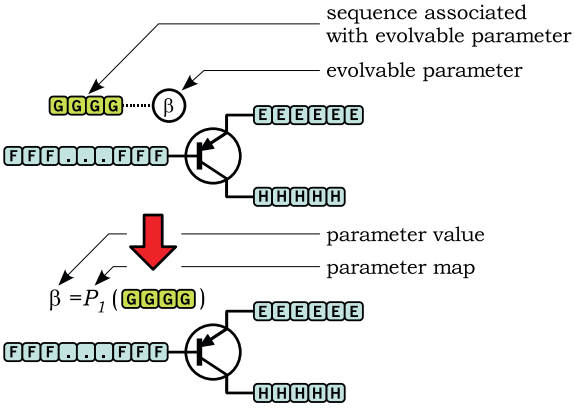


Figure 3.10: *The value of an evolvable parameter is assigned by applying a parameter map $P_1(s)$ to the sequence s of characters associated with the parameter*

worth considering the possibility of defining the parameter map $P_1(s)$ as $P_1(s) = P_2(s, s_p)$, where $P_2(s, s_p)$ is a map that associates a parameter value with the pair constituted by the sequence s and a fixed sequence of characters s_p (Figure 3.11). In this way, both the device interaction map and the parameter map correspond to a mapping from pairs of sequences to scalar values.

3.4 Device interaction map

As explained in Section 2.6 and in Section 3.3.1 the role of the device interaction map is that of transforming pairs of character sequences associated with two distinct device terminals into a scalar value that represents the strength of the direct interaction between the terminals. This map is obviously specific of the kind of analog network considered. For example, in the case of electronic circuits the device interaction map will produce values of conductance (Figure 3.8), whereas for neural networks it will produce weight values (Figure 3.9), which are typically dimensionless, and for genetic regulatory networks it will produce, for example, values of probability of activation of one gene on

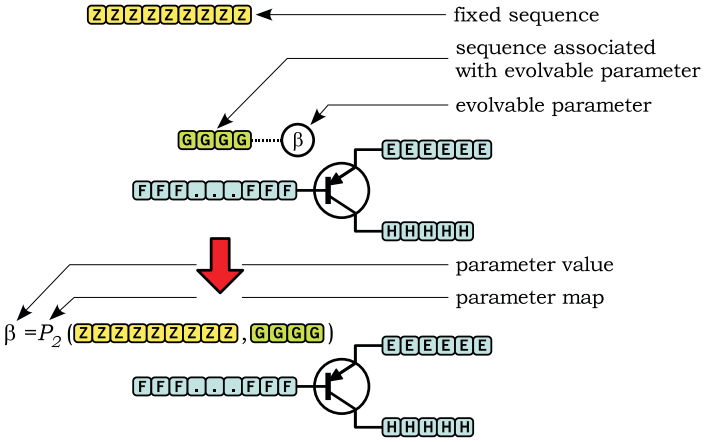


Figure 3.11: *The parameter map $P_1(s)$ can be defined in terms of a map $P_2(s, s_p)$ involving the sequence s associated with the evolvable parameter, and a fixed, predefined sequence s_p. In this way, both the device interaction map and the parameter map correspond eventually to a map from pairs of character sequences to scalar values.*

the part of another gene. Since we need to ensure that the device interaction map always satisfies the requirements derived in the previous chapter (Subsection 2.8.2), it is expedient to distinguish the network-specific component of this mapping, from a generic component which provides the required evolvability and complexity-growth potential. In this way, we could reuse this generic component for any kind of analog network evolution, and obtain in this way a device interaction map complying with the given prescriptions, without the need to reconsider each time the suitability of a device interaction map redefined from scratch for each particular kind of analog network.

We pursue this distinction of generic component from a specific component of the device interaction map by writing it as a composed map $N(L(s_1, s_2))$ formed by a generic *sequence interaction map* $L(s_1, s_2)$ that transforms pairs of sequences into abstract *sequence interaction values* i, and by a *network-specific interaction map* $N(i)$ that transforms sequence interaction values i into network-specific values of interaction

strength.[7] The current section is devoted to the definition of all the implementation details of the device interaction map used in this thesis, following the approach just mentioned.[8] Since many concepts and definitions are introduced in the following pages, a schematic overview of the principal elements that will be discussed is given in Figure 3.12, along with a brief description of the role of the various elements in the overall framework. The reader is encouraged to come back to this general scheme each time a new element is added to the global picture constituting the device interaction map.

3.4.1 Sequence interaction map

The sequence interaction map $L(s_1, s_2)$ is a generic map that transforms pairs of sequences into abstract sequence interaction values. Loosely speaking, this map must implement a generic interaction between sequences of characters which abstracts the interaction between fragments of biological genomes within genetic regulatory networks and complies with the prescriptions listed in Subsection 2.8.2.

There are certainly many ways to approach the definition of the sequence interaction map. A general observation concerning the device interaction map is that the fact of associating with pairs of sequences a scalar value representing the strength of their interaction defines implicitly a notion of similarity between pairs of sequences. Thus, we could be tempted to base the definition of the sequence interaction map on traditional ways to define the distance between two sequences. The simplest example that comes to mind is the Hamming distance, which counts the number of differences between two sequences of equal length. It is clear, however, that this simple choice does not comply with the prescriptions given in Subsection 2.8.2, if only for the constraint constituted by the required equal length of the sequences, which would restrict excessively the set of admissible genetic operators.

The use in the definition of the sequence interaction map of such a simple technique of evaluation of the distance between two sequences is in fact conceivable only if some additional transformation is defined, which mediates between the original pair of sequences and the pair of

[7] The sequence interaction map can then be also used to define the map $P_2(s, s_p)$ that is part of the parameter map.

[8] Note that other implementations of the device interaction map can be used in the context of the approach proposed in this thesis. It is sufficient that the alternative implementations of the sequence interaction map comply with the list of prescriptions given in Subsection 2.8.2.

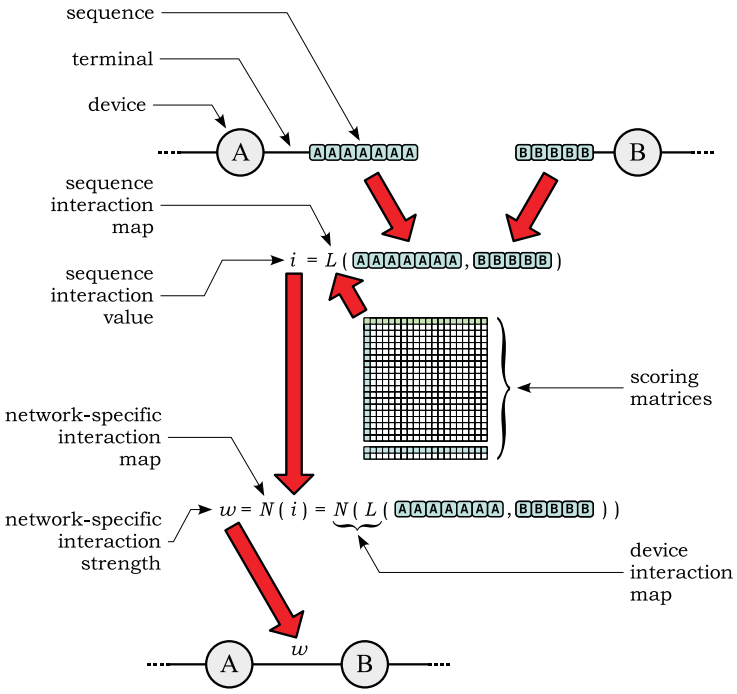


Figure 3.12: *A schematic representation of the elements that enter the definition of the device interaction map. First, the sequence interaction map $L(s_1, s_2)$ gives for each pair of sequences of characters (s_1, s_2) associated with a pair of device terminals, a scalar sequence interaction value i. Then, the network specific interaction map $N(i)$ transforms each sequence interaction value i into a scalar network specific interaction strength w between the terminals of the devices. The device interaction map corresponds to the composed map $N\,(L\,(s_1, s_2))$, which associates values of interaction strength with pairs of sequences associated with the terminals of the devices. When the sequence interaction map is defined in terms of sequence alignment, it is characterized by a pair of scoring matrices.*

sequences that are eventually compared, and which endows the composite map with the desired properties. For example, a definition directly inspired from the mediating role of the process of protein folding

in the biological context, could combine a map that realizes the embedding of one of the sequences in a high dimensional space (mimicking the mapping of the sequence of nucleotides coding for the protein into the high-dimensional space of protein shapes), another map that projects the result back in a space of character sequences (mimicking the generation of a region of the protein that must recognize the sequence of nucleotides that constitutes a regulatory binding site along the genome), and a final map that evaluates the distance of the resulting sequence relatively to the other, unprocessed sequence of the pair (mimicking the physical phenomena that determine the strength of the interaction between the regulatory binding site and the abovementioned protein region). Unfortunately, devising a pair of mappings that realize the embedding and the projection and endow the composed map with the required properties when combined with an elementary sequence similarity assessment, is a far from obvious task.

A simpler approach is to look for a more flexible definition of the similarity between sequences and use it directly as sequence interaction map. There is in fact a way to define a notion of similarity between sequences of characters, which – as will be shown below – when used directly as sequence interaction map endows it with the required properties. This notion of similarity corresponds to the concept of *local alignment* of pairs of sequences of characters, and will be described in the next section.

3.4.2 Global and local sequence alignment

A *global alignment* between two sequences of characters is a correspondence between the two sequences that puts each character of one sequence in correspondence with a character of the other sequence, or with a special *space* character (Figure 3.13). Note that, given two sequences, many different global alignments can be typically established between them.

The concept of global alignment can be used to define a measure of similarity between two sequences, called *global alignment score*. To define the concept of global alignment score it is sufficient to assign a *substitution score* to each possible correspondence of two characters of the two sequences, an *insertion score* to each correspondence of a character of the first sequence with a space, and a *deletion score* to each correspondence of a space with a character of the second sequence.[9]

[9]These scores are typically grouped in the *scoring matrices* which are represented

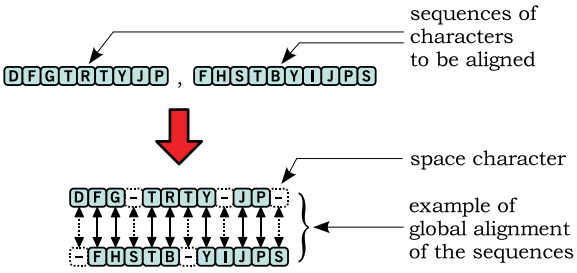


Figure 3.13: *A global alignment of two sequences of characters puts each character of one sequence in correspondence with a character of the other sequence, or with a special space character. The underlying idea is that one sequence is transformed into the other through a sequence of operations of substitution, insertion, and deletion of single characters. Here, substitutions (which include as a particular case the substitution of a character with itself) are represented by a solid arrow, whereas insertions and deletions are represented by a dotted arrow. Note that the figure represents only one of the many different global alignments that can be typically established between two given sequences.*

The underlying idea is that one sequence is transformed into the other using a series of character substitutions, insertions and deletions (abbreviated as *indels*), and that the more similar the characters put into correspondence are, the higher their substitution score. For each particular global alignment of two sequences it is therefore possible to assign an *alignment value* summing the scores of all correspondences established by the alignment. The global alignment score of two sequences is then defined as the highest alignment value attainable considering all the possible global alignments of the sequences (Gusfield, 1997). Note that the highest alignment value can be attained by two or more distinct global alignments of the two sequences.

If we compare the properties of the global alignment seen as a candidate for the implementation of the sequence interaction map, with the prescriptions listed in Section 2.8, we notice at once that one of the requirements is certainly not fulfilled, namely, the possibility for a single sequence of determining multiple independent interactions (Fig-

schematically in Figure 3.12 and whose structure will be described in Section 3.4.4.

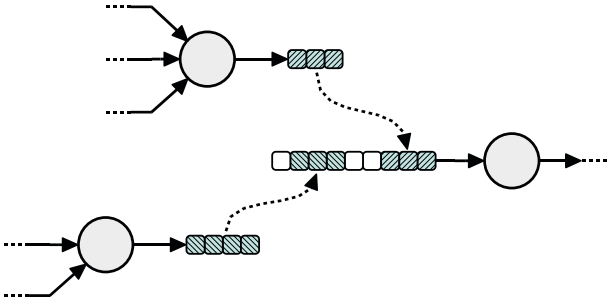


Figure 3.14: *As explained in Section 2.8, the evolvability of the analog networks can be expected to be favoured by the use of a device interaction map that has the possibility of determining multiple independent interactions on a single sequence. In terms of the sequence interaction map, this can be obtained by defining a map that has the possibility to operate locally on several distinct fragments of a single sequence, as shown in this figure. A sequence interaction map that forces instead a global matching on the whole sequence incurs the risk of producing interferences that can hamper the evolutionary process (see also Figure 2.3 on page 36).*

ure 3.14). This limitation follows from the *global* nature of the alignment, which forces the establishment of a correspondence between all the characters of the two sequences. This problem can be solved considering instead of the global alignment of two sequences, their *local alignment*. The difference is that in a local alignment not all characters of the two sequences must be put into correspondence, but only the characters of two portions of the sequences, called *matching regions*. The only condition is that the matching regions be composed of characters that are adjacent in the original sequences, so that only the initial and final portions of the sequences can be ignored in the alignment (Figure 3.15). In practice, this is obtained by assigning a negative or null value of alignment score to some character substitutions and indels, and ignoring in the alignments the leading and trailing characters of the sequences that do not contribute with a positive score to the alignment value.[10] The end result is an alignment where a single sequence can match independently many other sequences with several

[10] As a consequence, the local alignment score is a non-negative quantity.

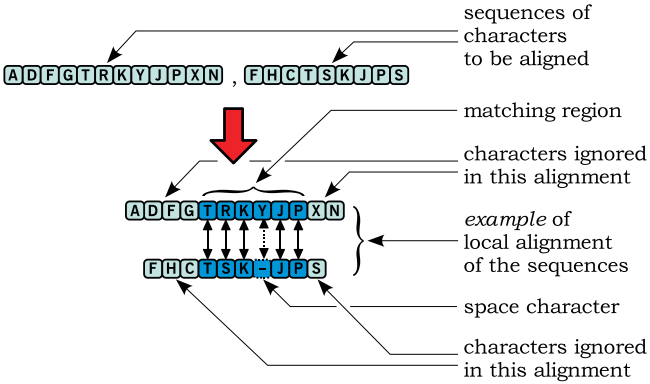


Figure 3.15: *A local alignment of two sequences of characters puts adjacent characters of one sequence in correspondence with adjacent characters of the other sequence or with a special space character, possibly ignoring characters before and after the matching region constituted by the characters that are put in correspondence. Like in the case of the example of global alignment given in Figure 3.13, this figure represents only one of the many different local alignments that can be typically established between two given sequences.*

distinct matching regions (Figure 3.16). Like for the global alignment, given the substitution, insertion, and deletion score for single characters it is possible to assign to each particular local alignment of two sequences an alignment value summing the scores of all correspondences established by the alignment. The *local alignment score* of two sequences is defined as the maximum alignment value attainable considering all the possible local alignments of the sequences (Gusfield, 1997).

The local alignment score is ideally suited to the implementation of the sequence interaction map, being endowed with the following properties:

- With a suitable choice of the character substitution and indel scores it can be made highly redundant.

- A single sequence can determine multiple independent interactions.

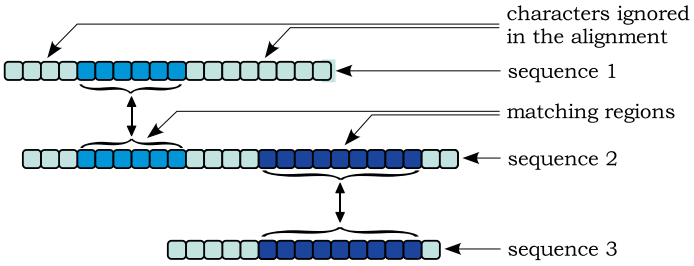


Figure 3.16: *Using the local alignment rather than the global alignment, one sequence can match independently many other sequences with multiple separate matching regions, although overlapping of the matching regions is still possible.*

- It operates on pairs of sequences of arbitrary length and independently from the original position of the sequences in the genome.
- With a suitable choice of the character substitution and indel scores, changes in the sequences can produce both small changes and large variations in the alignment score.

Note that the fact that the alignment score is defined as the maximum of all the alignment values attained by all the possible alignments implies that an alignment realizing the maximum value (an *optimal* alignment) masks all the alignments with a smaller alignment value. A mutation in the sequences can degrade the value of a formerly optimal alignment, promoting as new champion one of the previously masked alignments, which can thus be considered a "recessive" alignments relatively to the previously dominant alignment. This is potentially useful in an evolutionary perspective, and it is thus worth mentioning as an additional property of local alignment

- Besides the "dominant" interaction, two sequences can produce several "recessive" interactions.

From the point of view of the computational complexity, at first sight the calculation of the local alignment score of two sequences requires the generation of all possible correspondences of all portions (substrings) of one sequence with all portions (substrings) of the other

sequence, a feat which is computationally unfeasible for even moderate string lengths. There are, however, algorithms based on dynamic programming that compute the global and local alignment score of two strings of length m and n with computational complexity of order mn in time, order m in space (i.e., in memory), and order $|\mathcal{G}|$ in the alphabet size. A dynamic programming algorithm to compute global alignment was first described in (Needleman and Wunsch, 1970), while a dynamic programming algorithm for local alignment was introduced in (Smith and Waterman, 1981). Algorithms for global alignment are thus typically referred to as Needleman-Wunsch algorithms, whereas those devoted to local alignment are called Smith-Waterman algorithms. Since both the global and the local alignment of sequences are heavily used in bio-informatics, many references and textbooks exist on the subject (for example, Crochemore et al., 2001; Crochemore and Rytter, 2002; Gusfield, 1997; Mount, 2001; Sankoff and Kruskal, 1983). The reader is referred to these texts for the details of these algorithms. The only thing worth noting here concerning this aspect of the local alignment is that

- Its computational complexity is reasonably low.

The quadratic computational complexity suggests, however, the setting of a limit for the length of the sequences extracted from the genome and associated with the terminals and parameters of the devices.

Of the requirements listed in Section 2.8 for the device interaction map it remains therefore to assess only the possibility of generating large enough sets of independent sequences, that is, the possibility of keeping low the unwanted interference between distinct sequences. This aspect is related to the choice of the alphabet and of the substitution and indel scores. To proceed with these choices, however, besides considering the problem of interference between sequences, that is, besides considering what we want to avoid in our interaction map, we must ascertain what we positively want to obtain from it. This, in turn, is related to the number of distinct interaction strength values that we want to obtain from the map. Therefore, before considering the problem of alphabet size and scores value, we turn to consider briefly the nature of the network-specific interaction map whose role was briefly described at the beginning of this section and is illustrated in Figure 3.12.

3.4.3 Network-specific interaction map

The role of the network-specific interaction map is to transform the abstract sequence interaction values produced by the sequence interaction map into network-specific values of interaction strength between the terminals of the devices. The domain of the sequence interaction map is the set of pairs of character sequences built on the genetic alphabet. If we set an upper limit for the length of the sequences extracted from the genome, this domain is finite, otherwise it is countable. Hence, the range of the sequence interaction map is also finite or at most a countable set. The network-specific interaction map must thus associate with each element belonging to this set a value of interaction strength that is meaningful for the actual analog network considered in the experiment.

In practice, the finite resources available for the implementation of our evolutionary system imply that the set of possible interaction values is always finite. This is not a serious limitation, provided that we distribute wisely the finite number of interaction values that we can actually represent, within the (typically continuous) range of network-specific interaction values that we would like to represent. This is a problem that exists each time we have at our disposal finite resources to represent a continuous range of values. A first example is the choice of the numerical representation for real numbers within a computer. Another example is the choice of the finite set of standardized values of passive electronic component such as resistors, inductors, and capacitors, that can be actually manufactured and put at the engineer's disposal for the design of electronic circuits. Still another example is the finite set of coin and banknote values that is manufactured and publicly circulated.

Appendix B is devoted to the problem of choosing the distribution of a finite set of values within a continuous interval of real values. In that appendix it is shown that in many cases, including the case of the resistors in an analog electronic circuit, a logarithmic distribution of the selected values within the interval appears the optimal choice. Note that in the three examples considered above, if we assume as typical the floating point representation for real numbers within a computer, the distribution of the selected values is fundamentally logarithmic in all three cases.

It is worth examining in some detail the distribution and range of the standardized sets of resistors used in electronic circuit design, be-

cause it gives some indication of the total number of values that is considered sufficient in an actual, real-world scenario. The commonest standardized series of resistors is the so-called E12 series, which has 12 elements for each decade of electric resistance value and, consequently, for each decade of electric conductance[11] value (see Appendix B for the actual values of the E12 series). The range of values used by engineers goes typically from 1Ω to $10\text{M}\Omega$. Values outside this range are manufactured for special purposes, but are not considered part of the standardized series. Hence, the set of available resistors for the design of the typical electronic circuit comprises 85 elements, with values approximately logarithmically distributed across 7 decades of electric resistance. Note that limitations of printed board space and reasons of circuit reliability rule out the use of combinations of several resistors to produce a non-standard resistance value. By comparison, coins and banknotes – which are instead usually combined to produce "non-standard" monetary values – cover something like 5 decades with about 15 elements almost logarithmically distributed in the spanned range, using typically only 3 standardized values per decade.

To facilitate the task of evolution, we can stipulate that the number of interaction strength values available must be sufficient to avoid the need to combine several interactions to span the desired range of values. Note that we could be tempted to use many more elements than strictly needed, but this would have the undesirable effect of unnecessarily enlarging the search space, since the set of interaction strength values requires a corresponding number of sequence interaction values. In some cases it can be advisable to add to the required number of interaction strength values a few values of sequence interaction corresponding to the absence of any direct interaction. For example, in the case of electronic circuits, a range of low sequence interaction values could be associated with the zero-valued conductance (corresponding to the insertion of an infinite-valued resistor between the terminals, that is, to the insertion of no resistor at all). This choice is typically dictated by the desire to avoid the presence of too many connections in the analog network decoded from the genome, which could slow the simulations of the network or complicate their implementation. On the other hand, if the presence of many connections is not a problem (for example, if the simulation technique does not suffer from their presence), it is probably better to avoid the use of a "dead-zone" of absence

[11] See Note 5 on page 69.

of interaction, to give evolution the possibility of gradually probing the effect of the various interactions.

Another useful expedient is the association with a predefined maximum value of interaction strength, of all the sequence interaction values above a certain limit. For example, in the case of electronic circuits, the maximum interaction strength between two terminals corresponds – theoretically – to a null value of resistance. i.e., to an infinite value of conductance of the corresponding connection. Instead of considering arbitrarily large values of conductance (which, in practice, are operationally unfeasible) as associated with increasing values of sequence interaction strength, it is preferable to consider a finite and operationally reasonable maximum value of conductance as associated with a given sequence interaction strength, and to transform all sequence interaction strengths above that maximum value into a connection with infinite conductance.

Example: Let us consider how a network-specific interaction map $i \stackrel{N(i)}{\longrightarrow} g$ from a range of integer sequence alignment scores i to a set of conductance values g complying with these indications can be actually defined. Let us assume a logarithmic quantization of the conductance range (Appendix B) determining a set of discrete conductance values having n_d elements per decade of conductance value. We denote by g_0 the zero-valued conductance, by g_{min} the minimum non null value of conductance, by g_{max} the maximum finite value of conductance, and by g_∞ the infinite-valued conductance. The sequence of conductance values corresponds to

$$\{g_0,\, g_{min},\, \alpha g_{min},\, \alpha^2 g_{min},\, \ldots\, , \alpha^{n_d-1} g_{min},\, 10\, g_{min},\, 10\, \alpha g_{min},\, \ldots\, g_{max},\, g_\infty\}$$

where $\alpha = 10^{1/n_d}$. Denoting by n_s the number of elements in the subsequence $\{g_{min,}\, \ldots\, , g_{max}\}$ (that is, the number of discrete conductance values excluding the connection with infinite conductance represented by g_∞ and the absence of connection represented by g_0), g_{max} must satisfy the condition $g_{max} = \alpha^{n_s-1} g_{min}$. Denoting by i_{min} the positive integer sequence interaction value associated with g_{min}, we obtain the network-specific interaction map shown in Table 3.1.

Note that i_{min} defines the "dead-zone" of sequence interaction values associated with g_0, that is, with the absence of direct interaction. If $i_{min} = 0$, no dead zone exists and any positive sequence interaction value (that is, any positive local alignment score) from two sequences

$$
\begin{aligned}
i < i_{min} &\longrightarrow g_0 \\
i_{min} &\longrightarrow g_{min} \\
i_{min} + 1 &\longrightarrow \alpha\, g_{min} \\
i_{min} + 2 &\longrightarrow \alpha^2\, g_{min} \\
&\vdots \\
i_{min} + n_d - 1 &\longrightarrow \alpha^{n_d - 1}\, g_{min} \\
i_{min} + n_d &\longrightarrow 10\, g_{min} \\
i_{min} + n_d + 1 &\longrightarrow 10\, \alpha\, g_{min} \\
&\vdots \\
i_{max} = i_{min} + n_s - 1 &\longrightarrow g_{max} = \alpha^{n_s - 1}\, g_{min} \\
i > i_{max} &\longrightarrow g_\infty
\end{aligned}
$$

Table 3.1: *A theoretical example of network-specific interaction map using a logarithmic distribution of interaction strength values with n_d values of interaction strength per decade. The range $i < i_{min}$ of sequence interaction values is associated with the absence (or, more generally, minimum strength) of interaction g_0, and the range $i > i_{max}$ of sequence interaction values is associated with the direct (or, more generally, maximum strength) connection g_∞. The integer parameter n_s corresponds to the number of discrete interaction strength values besides the direct connection and the absence of connection.*

associated with device terminals results in the insertion of a conductance between the terminals. The range $i > i_{max}$ of sequence interaction values is associated with the connection having an infinite-valued conductance which is represented by g_∞. Setting $i_{max} = +\infty$ (or, equivalently, $n_g = +\infty$) produces an unbounded sequence of interaction values.

To illustrate in practice this technique of definition of the network-specific interaction map, let us assign some actual values to the parameters and coefficients just introduced. Choosing a minimum resistance value of $1\,\Omega$ and a maximum resistance value of $1\mathrm{M}\Omega$, we obtain $g_{min} = 10^{-6}\,\mathrm{S}$ and $g_{max} = 1\,\mathrm{S}$. The range of conductance values spans six decades. Setting the number of values per decade to $n_d = 8$ we obtain $n_g = 6\,n_d + 1 = 49$ and $\alpha = 10^{1/8} \approx 1.33$. If we choose a value of $i_{min} = 20$ (this choice will be discussed further in the next subsection) we obtain the actual network-specific interaction map $i \xrightarrow{N(i)} g$ illustrated in Table 3.2.

If the circuit to be synthesized must be realized with discrete com-

$$
\begin{aligned}
i < i_{min} = 20 \quad &\longrightarrow \quad g = g_0 = 0 \ \ (\textit{no connection}) \\
i = 20 \quad &\longrightarrow \quad g = g_{min} = 10^{-6}\,\mathrm{S} \\
i = 21 \quad &\longrightarrow \quad g = \alpha\, g_{min} \approx 1.33 \times 10^{-6}\,\mathrm{S} \\
i = 22 \quad &\longrightarrow \quad g = \alpha^2\, g_{min} \approx 1.78 \times 10^{-6}\,\mathrm{S} \\
&\vdots \\
i = i_{min} + n_g - 1 = & \\
= 20 + 49 - 1 = 68 \quad &\longrightarrow \quad g = g_{max} = 1\,\mathrm{S} \\
i > i_{max} = 68 \quad &\longrightarrow \quad g = g_\infty = \infty \ \ (\textit{direct connection})
\end{aligned}
$$

Table 3.2: *An actual example of a network-specific interaction map using a logarithmic distribution of interaction strength values with $n_d = 8$ values of interaction strength per decade of conductance values, a minimum non-zero value $g_{min} = 10^{-6}\,\mathrm{S}$, a maximum finite value $g_{max} = 1\,\mathrm{S}$ of conductance, and $i_{min} = 20$.*

ponents, it is advisable to use directly the elements of one of the standard resistor series – for example the E12 series mentioned above – rather than analytically defined logarithmically distributed values. In that case, g_{max} would correspond to the minimum resistance value of the series, g_{min} would correspond to its maximum resistance value, n_g would be the number of elements in the series, and the device interaction map would associate values of sequence interaction strength from i_{min} to i_{max} with the resistance (conductance) values of the standard series.

3.4.4 Scoring matrices

Now that the details and requirements of the network-specific interaction map have been analyzed, we can go back to the definition of the details of the sequence interaction map $L(s_1, s_2)$. To actually implement the sequence interaction map in terms of local alignment of sequences we must assign the genetic alphabet and the scores that are required for the computation of the local alignment over that alphabet. This means assigning the *substitution scores* $c_{x \to y}$, the *insertion scores* $c_{- \to y}$ and the *deletion scores* $c_{x \to -}$ for all characters x and y belonging to the genetic alphabet. The scores are grouped into a *substitution matrix*, an *insertion vector*, and a *deletion vector*, which, together, form the *scoring matrices*.

The choice of the scoring matrices is a central problem of parameter assignment for an evolutionary system that uses the alignment of sequences to implement the device interaction map. In a sense, the scoring matrices implicitly define the "physics" of the interaction between the portions of the genome. To solve this parameter assignment problem, we can start by listing some elementary conditions that the scores must satisfy. First, although the scores could be real numbers, there is no loss of generality in considering them integers, since at this level we are concerned only with the ordering of the interactions. It is the role of the network-specific interaction map to transform these ordered elements into actual values of interaction strength. Second, some non negative scores must be present in the scoring matrices, otherwise the local alignment algorithm will always return two empty matching regions and a null alignment score. Third, not all entries of the scoring matrices can be positive, otherwise the alignment algorithm will tend to put in correspondence all the characters of the two strings instead of ignoring some characters at the start or end of the sequences, and the alignment would therefore tend to be global rather than local. Typically, there are some pairs of characters that are considered exact or close matches and assigned a positive substitution score, and all other pairs are considered mismatches and attributed a negative substitution score. Insertion and deletion scores are typically also assigned a negative score. Fourth, the condition $c_{x\to y} \geq c_{x\to -} + c_{-\to y}$ must be observed, otherwise the substitution $x \to y$ will never appear in an optimal alignment, the pair formed bz the deletion of x and the insertion of y achieving higher score than the direct substitution of x with y. Fifth, since the pair of sequences that must be aligned is assumed as unordered, the symmetry conditions $c_{x\to y} = c_{y\to x}$ and $c_{-\to y} = c_{y\to -}$ must be respected by the substitution, insertion, and deletion scores. This means that the substitution matrix is symmetric and that the insertion vector and the deletion vector coalesce into a unique *indel vector* of *indel scores*. Figure 3.17 shows an example of scoring matrices satisfying all these requirements. Note that besides the conditions given above, the substitution matrix shown in Figure 3.17 is also *circulant*, that is, its rows are cyclically shifted versions of each other. Moreover, in Figure 3.17 the indel score is the same for all characters. Both choices, although not indispensable, simplify the implementation of the alignment algorithm. All the scoring matrices considered in this work will have both these characteristics and, therefore, only the first row of the substitution matrix and the first element of the indel vector will be ex-

substitution matrix

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	5	2	1	0	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1	0	1	2
B	2	5	2	1	0	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1	0	1
C	1	2	5	2	1	0	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1	0
D	0	1	2	5	2	1	0	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1
E	-1	0	1	2	5	2	1	0	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2
F	-2	-1	0	1	2	5	2	1	0	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5
G	-5	-2	-1	0	1	2	5	2	1	0	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5
H	-5	-5	-2	-1	0	1	2	5	2	1	0	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5
I	-5	-5	-5	-2	-1	0	1	2	5	2	1	0	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5
J	-5	-5	-5	-5	-2	-1	0	1	2	5	2	1	0	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5
K	-5	-5	-5	-5	-5	-2	-1	0	1	2	5	2	1	0	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5
L	-5	-5	-5	-5	-5	-5	-2	-1	0	1	2	5	2	1	0	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5
M	-5	-5	-5	-5	-5	-5	-5	-2	-1	0	1	2	5	2	1	0	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5
N	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1	0	1	2	5	2	1	0	-1	-2	-5	-5	-5	-5	-5	-5	-5
O	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1	0	1	2	5	2	1	0	-1	-2	-5	-5	-5	-5	-5	-5
P	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1	0	1	2	5	2	1	0	-1	-2	-5	-5	-5	-5	-5
Q	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1	0	1	2	5	2	1	0	-1	-2	-5	-5	-5	-5
R	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1	0	1	2	5	2	1	0	-1	-2	-5	-5	-5
S	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1	0	1	2	5	2	1	0	-1	-2	-5	-5
T	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1	0	1	2	5	2	1	0	-1	-2	-5
U	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1	0	1	2	5	2	1	0	-1	-2
V	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1	0	1	2	5	2	1	0	-1
W	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1	0	1	2	5	2	1	0
X	0	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1	0	1	2	5	2	1
Y	1	0	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1	0	1	2	5	2
Z	2	1	0	-1	-2	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-2	-1	0	1	2	5

indel vector

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
–	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3

Figure 3.17: *An example substitution matrix (top) and indel vector (bottom). These scoring matrices comply with the conditions given in the text for the substitution, insertion, and deletion scores. In addition, the substitution matrix is a circulant matrix, and the indel score is the same for all characters in the alphabet. These two additional properties simplify the implementation of the alignment algorithm.*

plicitly shown from now on.

The list of requirements given above puts a series of constraints on the structure of the scoring matrices, but does not determine the size of the genetic alphabet, nor the actual entries of the scoring matrices. To proceed further in the parameter assignment we thus need to

consider what we positively want to obtain from our sequence alignment. This, in turn, derives from the requirements specified by the network-specific interaction map analyzed in the previous subsection. A first requirement is the possibility to exceed with the local alignments score a certain maximum value i_{max} that we want to associate with the maximum network-specific interaction strength. At the same time we want to avoid the necessity of dealing with extremely long sequences in order to generate values of alignment score in the i_{max} range, since that would require long genomes and long computation times for the alignment algorithm. These two requirements can be met ensuring that some of the substitution scores have large enough positive values. These large positive values, however, would tend to favour the indiscriminate production of "global-like" alignments with high alignment scores. To avoid an excessive interference between sequences and to keep local the alignment we must thus balance the positive score values with negative values. A reasonable policy is to ensure a negative average score for the elements of the scoring matrices (Gusfield, 1997, p. 235). It is especially easy to comply with policy, even in presence of high positive scores, if the size of the genetic alphabet is large, as this ensures the existence of many pairs of characters that can be considered mismatches and to which a negative substitution score can be attributed. The increase of the size of the genetic alphabet has also the beneficial effect of increasing the redundancy of the device interaction map. On the other hand, a larger alphabet enlarges the search space and slows the sequence alignment algorithms, although the additional redundancy can compensate for this, hopefully transforming the search for a needle in a haystack into the search for a needle in a haystack full of needles (Pattee, 2001, p. 18). Finally, the entries of the scoring matrices must permit the production of all the possible values of alignment score up to i_{max}, in order to permit the production of all the values of interaction strength belonging to the range of the network-specific interaction map. This means that there must be scores of various magnitude in the scoring matrices which, when combined, must be able to produce all values of sequence interaction strength composing the domain of the network-specific interaction map. The scoring matrices shown in Figure 3.17 comply with these requirements, with large positive scores up to 5, an alphabet of 26 characters that permits the balancing of the positive scores with many negative scores, and a gradual transition from positive to negative substitution scores. Figure 3.18 and Figure 3.19 show two sets of scoring matrices still complying with

substitution matrix

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	4	3	2	1	0	-1	-2	-3	-4	-5	-5	-5	-5	-5	-5	-5	-5	-5	-4	-3	-2	-1	0	1	2	3

⋮

indel vector

	A	⋯
–	-3	

Figure 3.18: *An example of scoring matrices complying with the requirements listed in the text and with a smoother transition from positive to negative values of substitution score relatively to the case illustrated in Figure 3.17. Remember that the substitution matrix is a symmetric circulant matrix, and the indel score is the same for all characters in the alphabet.*

substitution matrix

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	2	1	0	-1	-2	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-2	-1	0	1

⋮

indel vector

	A	⋯
–	-2	

Figure 3.19: *An example of scoring matrices with a smaller maximum positive substitution scores relatively to the cases illustrated in Figure 3.17 and Figure 3.18.*

the above mentioned requirements but with a smoother transition from positive to negative values in the substitution matrix relatively to the case shown in Figure 3.17, and with progressively smaller magnitude of the maximum substitution score.

3.4.5 Randomly generated interactions

In the preceding subsections it was shown that the value of i_{max} follows as a consequence of the number n_s of interaction strength values that we want to represent, and of the value i_{min} defining the limit of the dead-zone of sequence interaction values associated with the absence of direct interaction. The need for this dead-zone stems from

the fact that pairs of randomly generated sequences will tend to have a non null local alignment score. It can be convenient to have most of these random alignment scores fall within the dead-zone, in order to minimize their interference with the alignments determined by the evolutionary process in its effort to functionally structure the evolving network. On the other hand, it is also desirable to have some overlap between the range of alignment scores that can be generated randomly with non negligible probability, and the range of alignment scores that are actually mapped to non null interaction strengths. In this way a range of weak interactions can be generated randomly with non negligible probability, thus facilitating the "probing" from the part of the evolutionary process, of the effect of the presence of the corresponding connections between the devices encoded in the genome. We have thus a scenario where, given a genetic alphabet size and a set of scoring matrices, the value of i_{min} is determined as the upper limit of the range of alignment scores that are generated with non negligible probability aligning randomly generated pairs of sequences over the given genetic alphabet and using the given scoring matrices. The value corresponding to a "non negligible probability" depends on the estimate of the number of pairs of sequences associated with terminals in a typical evolving network. A value of probability between 10^{-2} and 10^{-3} can be considered reasonable for the experiments reported in this thesis[12].

The choice of the alphabet size and of the scoring matrices is an iterative process. Given the number n_s of alignment score values required by the kind of network that must be evolved, we start by assigning tentative values to the alphabet size and to the scoring matrices. Then, we determine the probability distribution of the alignment scores of pairs of randomly generated sequences. We use this probability distribution to assign i_{min} so as to obtain the slight overlapping of the range of the n_s active alignment scores with the range of random scores, as described

[12] One could object that we should consider not only the interaction between *pairs* of sequences, but the interaction between *triples* and, more generally, between n-*tuples* of randomly generated sequences. We must consider, however, that Darwinian evolution is driven by the active process of selection. Thus, we can expect artificial evolution to be endowed with the power not only to counter, but indeed to *exploit* the existing entropic effects, provided these effects are not overwhelming. The restriction of the limitation of the interference to the case of pairs of sequences has the double goal of ensuring that the random component is neither overwhelming, nor practically absent. The experiments of sequence alignment described in the next chapter are intended as a mean to probe if the remaining randomness is sufficient for evolution to kick off and proceed, and if the process of selection is able to keep under control and to exploit this residual randomness.

above. There is however the problem that the local alignment score that can be expected from two random sequences depends on the length of the sequences. Hence we must estimate the length of the typical sequence associated to a terminal of an evolving network. This estimate depends on the kind of network that must be evolved and is in general difficult to obtain. A heuristic strategy is to assign the number n_t of terminals to which each terminal must be capable to independently connect through a wire (i.e., a connection with infinite conductance). This requires the possibility of generating n_t independent alignment scores exceeding the i_{max} limit (remember that $i > i_{max}$ is the range of alignment scores that is associated with the maximum strength connection). If c_{max} is the maximum value existing in the scoring matrices, a sequence capable of generating these n_t independent alignment scores has a minimum length l equal to $n_t\, i_{max}/c_{max}$. Since i_{max} follows from the relation $i_{max} = i_{min} + n_s - 1$, which requires the knowledge of i_{min}, we assume an initial value for our first iteration of $i_{min} = 0$, from which we obtain an initial estimate of i_{max} and of l. This value of l is used to generate the distribution of probability of the alignment score of randomly generated sequences, from which an improved estimate of i_{min} is obtained and reinserted in the computation loop, until a stable estimate of i_{min} and, therefore, of i_{max} is obtained. This value of i_{max} determines the minimum length i_{max}/c_{max} of a sequence that can produce an alignment score corresponding to a connection with maximum strength. If this length is not judged excessive for the implementation (in particular, for the computation time of the alignment scores and the size of the resulting genome), the alphabet length, scoring matrices, and the value of i_{min} obtained can be used in the evolutionary experiments, otherwise the tentative values of the alphabet size and of the scoring matrices are corrected and the computation is restarted.

In practice, once the tentative alphabet size and scoring matrices are assigned the first iteration of this cycle often gives already a good idea of the suitability of the choice. The only problem lies in the generation of the probability distribution of the alignment scores for pairs of randomly generated sequences, since the analytical determination of this distribution given the alphabet size and scoring matrices is still an open problem. It is known, however, that the distribution is not Gaussian and belongs instead to the class of the so-called extreme value distributions, which are asymmetric and have a long tail in the high score range (Mount, 2001).[13] For our purpose it is sufficient to estimate the

[13] Random variables characterized by an extreme value distibution are obtained, for

form of these distribution by generating sets of random sequences.

As an example of application of this approach in the context of the evolution of analog electronic circuits, let us consider as our tentative choice the scoring matrices shown in Figure 3.17 with an alphabet size $|\mathcal{G}| = 26$. The maximum substitution score in this case is $c_{max} = 5$. The arguments and examples of the previous subsection have shown that a value of $n_s \approx 100$ is typically sufficient to span several decades of interaction strength. Let us assume also that number of terminals to which each terminal must be capable to connect independently through a wire is $n_t = 2$. We thus obtain a first iteration value of typical sequence length of $l = n_t\, n_s / c_{max} = 40$ characters. Figure 3.20 shows the estimate of the probability distribution of the values of local alignment scores of randomly generated pairs of sequences of length 40 built on an alphabet with size $|\mathcal{G}| = 26$ using the scoring matrices shown in Figure 3.17. Assuming a value of probability density of about 10^{-3} as the upper limit of the range of random scores, we obtain a first iteration value of $i_{min} \approx 30$, which gives a value of $i_{max} \approx 130$, and a length $i_{max}/c_{max} \approx 26$ for a sequence that can produce an alignment score corresponding to a connection with maximum strength. Using the dynamic programming algorithm, the local alignment score of pairs of sequences having lengths in this range can be computed in reasonable time on present day computing machines. Moreover, the memory occupation of a genome with some thousand sequences in this range of length appears also reasonable. Hence, this choice of alphabet with size $|\mathcal{G}| = 26$ using the scoring matrices appears acceptable. Figure 3.21 (top) shows the probability distribution obtained with the same structure of scoring matrix, but reducing the alphabet size to $|\mathcal{G}| = 20$. Comparing this distribution with that relative to $|\mathcal{G}| = 26$ shown in Figure 3.20 reveals that the first iteration estimate of i_{min} has increased from $i_{min} \approx 30$ to $i_{min} \approx 40$, which corresponds to an increase of i_{max}/c_{max} from about 26 to about 28 characters. This slight increase appears acceptable in terms of resulting memory and computation time. Figure 3.21 (bottom) shows the probability distribution obtained by reducing the alphabet size still further to $|\mathcal{G}| = 12$. With this reduction of alphabet size the average of the entries of the scoring matrices is only slightly negative. Now the first iteration estimate of i_{min} has increased

example, when a series of batches of random variables are drawn from a sufficiently well-behaved distribution, and the maximum value of each batch is retained. The definition of sequence alignment score corresponds manifestly to this kind of process, since a whole collection of alignment values corresponds to each pair of sequences, with the alignment score defined as the maximum attainable alignment value.

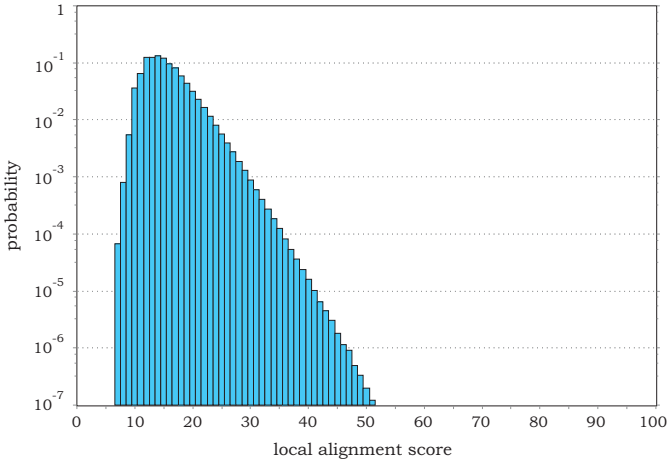


Figure 3.20: *The estimate of the probability distribution of the local alignment score of pairs of randomly generated sequences of length 40 built on an alphabet with size $|\mathcal{G}| = 26$ using the scoring matrices shown in Figure 3.17. This histogram and those shown in the subsequent figures in this section were obtained generating 100,000,000 pseudo-random pairs of sequences. Despite the long tail of the distribution, the probability drops rapidly in the high score range.*

to $i_{min} \approx 70$. This means that now the effects of random alignments "waste" the equivalent of about 14 matching characters, with a value of i_{max}/c_{max} increased to about 34. This makes this choice of alphabet size less appealing than the previous ones. Moreover, with this large value of the first iteration estimate of i_{min}, further iterations are advisable to obtain a better estimate of i_{min} and i_{max}.

Figure 3.22 shows the probability distributions relative to the choice of the scoring matrices illustrated in Figure 3.18. Now the maximum substitution score is $c_{max} = 4$, which leads to an increase to 50 characters of the estimated typical sequence length for $n_t = 2$. The case $|\mathcal{G}| = 26$ appears still acceptable, but now already with $|\mathcal{G}| = 20$ the range of random alignment scores extends above 60 characters, which makes this choice of scoring matrices less suited to small alphabets.

Figure 3.23 shows the probability distributions relative to the choice

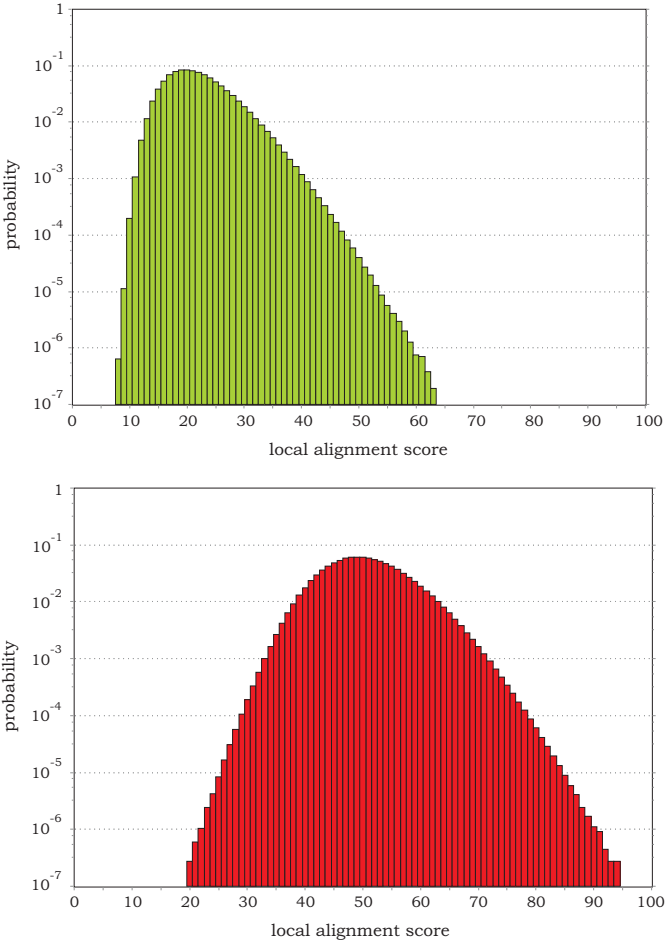


Figure 3.21: *The estimate of the probability distribution of the local alignment score of pairs of randomly generated sequences of length 40 built on an alphabet with size $|\mathcal{G}| = 20$ (top) and $|\mathcal{G}| = 12$ (bottom) using scoring matrices with the structure shown in Figure 3.17. Comparing these plots with that for the case $|\mathcal{G}| = 26$ illustrated in Figure 3.20, we see that the peak moves progressively toward larger values of alignment score and broadens, so that progressively larger values of interference can be expected.*

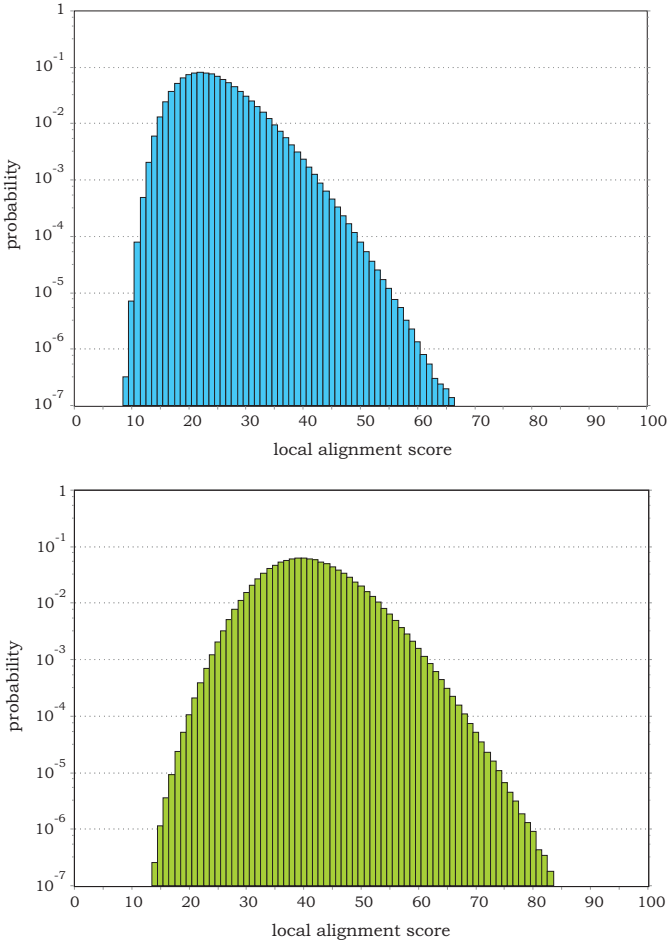


Figure 3.22: *The estimate of the probability distribution of the local alignment score of pairs of randomly generated sequences of length 50 built on an alphabet with size $|\mathcal{G}| = 26$ (top) and $|\mathcal{G}| = 20$ (bottom) using scoring matrices with the structure shown in Figure 3.18.*

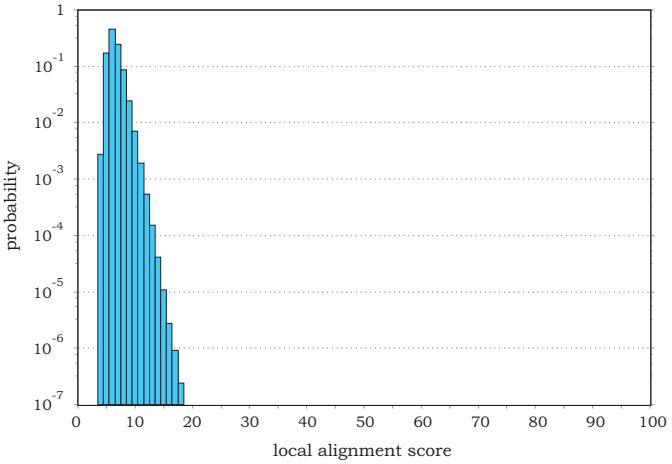


Figure 3.23: *The estimate of the probability distribution of the local alignment score of pairs of randomly generated sequences of length 100 built on an alphabet with size $|\mathcal{G}| = 26$ using scoring matrices with the structure shown in Figure 3.19. The small maximum positive value of substitution score and the abundance of negative scores in the scoring matrices keeps small the upper limit of the range.*

of the scoring matrices illustrated in Figure 3.19 and to an alphabet size $|\mathcal{G}| = 26$. The maximum substitution score has further reduced to $c_{max} = 2$, which leads to an increase to 100 characters of the first iteration estimate of the typical sequence length for $n_t = 2$. Thanks to the absence of large positive substitution scores, and to the balancing of the existing positive scores with many negative ones, the random alignment score region is limited to slightly more than 10 characters. This small value is partially compensated by the effect of the small value of c_{max}, which gives a value of i_{max}/c_{max} of about 55 characters.

Summing up, the analysis conducted above and the examples presented suggest the choice of a genetic alphabet size greater than about 20, and the adoption of a set of scoring matrices following the pattern of those shown in Figure 3.17 and Figure 3.18, with a sufficiently large maximum positive substitution score and a sufficiently negative average of the entries of the scoring matrices. The suitability of this kind of

choice will be further probed in the experiments of sequence matching described in the next chapter (and also, of course, in the experiments of analog network evolution reported in the same chapter). The exploratory experiments performed with the evolutionary system suggest in any case that the behavior of the evolutionary experiments is not too sensitive to the choice of the alphabet size and of the scoring matrices, provided the general guidelines given above are followed.

3.4.6 Interaction silencing

In the previous chapter (Section 2.8) it was pointed out that an evolutionary system for analog networks which uses pairs of character sequences to determine the interaction between the devices, is at risk of being plagued by the problem of interference between the sequences. When the sequence interaction is implemented using the local alignment technique detailed above, the judicious choice of the genetic alphabet and of the scoring matrices advocated in the previous subsection in order to keep at bay the effect of random interactions is a first tactic against the excessive increase of the interferences. It is clear, however, that above a certain level of complexity of the network, the number of devices, terminals, and associated sequences will overwhelm this kind of provision and will, for example, frustrate the evolution of networks composed of a large number of weakly connected devices.

This problem can be addressed in several ways. A general approach to the problem can be based on the assumption that two device terminals must be considered for connection only if the associated sequences comply with some specific condition. Subsection 3.5.3 below will describe the version of this approach that corresponds to the *compartmentalization* of the network. In that case, the assumption is that the default condition for a pair of terminals is the absence of connection, and that the presence in the sequences associated with the terminals of a common motif defining the terminals as belonging to the same compartment is required for their being considered for connection. The present subsection describes the complementary approach, that we call *interaction silencing*. This approach is based on the assumption that the default condition for a pair of terminals is the potential existence of an interaction, and that the presence in the sequences associated with the terminals of specific motifs has the effect of the silencing of the potential interaction. This determines a mechanism that has some similarity with the post-transcriptional silencing of genes operating in

biological GRNs (Lewin, 2004; Meister and Tuschl, 2004).

A possible implementation of interaction silencing (reminiscent of the phenomenon of RNA interference in biological GRNs brought about by complementary short RNA strands) is illustrated in Figure 3.24. For each letter of the genetic alphabet a "complementary" character is defined, chosen between those that have a large negative substitution score relatively to that letter in the local sequence alignment scoring matrices. When the sequence associated with a terminal contains a substring of sufficient length (in the sense considered in Subsection 3.1.2 while discussing the length of the tokens) that matches exactly the complementary substring in the sequence associated with another terminal, the two terminals will not be considered for connection, irrespective of the value of interaction strength between the two sequences. The minimal length of the matching substrings that determine the silencing of the interaction is chosen according to the same criteria used to select the length of the tokens and described in Subsection 3.1.2, which ensure that randomly produced silencing is not too frequent. Typically, a minimal length of just a few characters will be sufficient to avoid an excessive amount of randomly generated silencing.

Note that the use for interaction silencing of the exact matching of substrings of complementary characters that are defined as badly matched relatively to the local alignment parameters, ensures that these substrings will not produce a positive value of interaction strength according to the sequence interaction map. Thus, the sequence interaction determined by the local alignment algorithm and the mechanism of interaction silencing just described can coexist with minimal reciprocal disturbance. Note that once again the local nature of the sequence alignment is instrumental in allowing the simultaneous presence of fragments of genome with independent roles in the determination of the network connectivity. The search for exactly matching complementary substrings entails a computational complexity that is linear in time relatively to the length of the sequences (Gusfield, 1997). Thus, the computational complexity of the decoding of a genome where interaction silencing is widespread, is reduced relatively to the same genome where this mechanism is not implemented, since the sequence interaction map (whose computational complexity is quadratic relatively to the length of the sequences) need not be applied to pairs of sequences whose interaction has been silenced. To facilitate the evolutionary emergence of interaction silencing, it is useful to define an additional

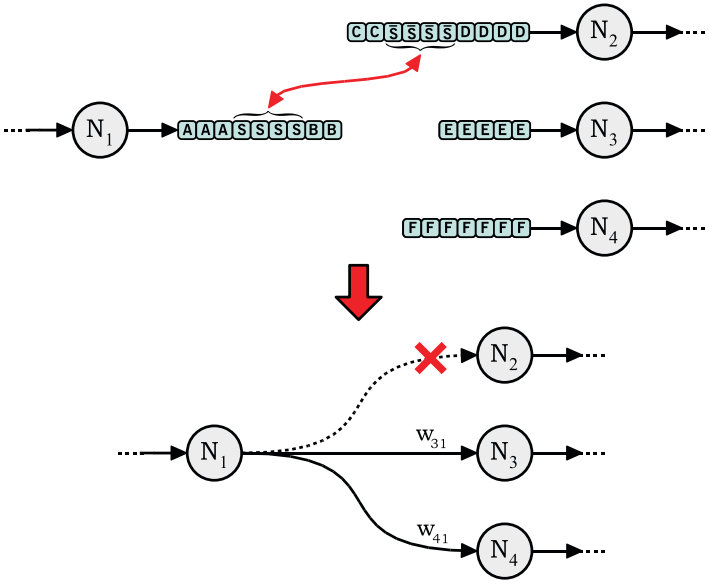


Figure 3.24: *Interaction silencing by complementary exact matching of motifs in the sequences associated with the terminals whose interaction must be silenced. For each letter s of the genetic alphabet a "complementary" character $\overline{s}$ is defined, chosen so as to minimize the disturbance of interaction silencing with the local alignment that determines sequence interaction. When the sequence associated with a terminal contains a substring of sufficient length that matches exactly the complementary substring in the sequence associated with another terminal, the two terminals are not considered for connection, irrespective of the value of interaction strength between the two sequences.*

genetic operator of complemented chromosome fragment duplication, where a genome fragment is chosen at random, duplicated, and the duplicate is complemented and randomly inserted in the genome.

3.4.7 Genetic code

A final remark can be made about the different constraints imposed on the genetic alphabet by the requirements of the sequence interac-

tion mapping and by those of the interaction silencing. We said previously that the sequence interaction map must be a many-to-one map, and allow a gradual transition from the absence of interaction to the maximum magnitude of interaction strength. The interaction silencing (and the evolvable compartmentalization described below in Subsection 3.5.3), on the other hand is based just on an on-off threshold mechanism that does not require any graduality. For this reason the two functionalities can be implemented with different techniques, such as local sequence alignment for the former and exact string matching for the latter. We have seen in Subsection 3.4.5 that the genetic alphabet used with local sequence alignment cannot bee too small, to avoid the excessive presence of high-valued random interactions. On the other hand, for exact string matching this problem is of much less concern, since - like in the case of the tokens examined in Subsection 3.1.2 - the probability of randomly generating a string that matches exactly a given string of length $|t|$ with an alphabet of size $|\mathcal{G}|$ is $|\mathcal{G}|^{-|t|}$. This value can be easily reduced to negligible values by increasing sufficiently $|t|$, even with small genetic alphabets.

From these observation it follows that would be conceivable to use in our evolutionary system a small genetic alphabet $|\mathcal{G}|$, and implement interaction silencing (and evolvable compartmentalization) in terms of exact matching between (possibly complementary) sequences of nucleotides. In order to obtain a larger alphabet for the implementation of device interaction, n-tuples of nucleotides could be univocally associated with the elements of a larger interaction alphabet $|\mathcal{A}|$. The interaction map would be thus defined between pairs of sequences of n-tuples of nucleotides, interpreted as sequences from this larger alphabet. This scenario is clearly very similar to that observed in existing biological systems, where the genetic code puts into correspondence triplets of nucleotides (which belong to an alphabet of size $|\mathcal{G}| = 4$) with amino acids (which form an alphabet of size $|\mathcal{A}| = 20$), and where interaction silencing appears to be implemented in terms of (almost) exact matching of sequences from the four-letter RNA alphabet, whereas protein-mediated gene interaction works in more gradual terms and using the larger amino acid alphabet of 20 letters.

3.5 External connections

In Section 2.9 the problem of the exchange of signals between an evolving system and its environment was considered in general and abstract terms. The result of the analysis carried out in that section was that a sequence-based interaction map can be used not only to determine the connections within the system, but also to define the connections crossing the boundary of the evolving system. The present section puts those conclusions into practice, showing how this extension of the sequence-based approach to the external connections can be actually defined and implemented.

The basic idea for the establishment of the connections of an evolving analog network with its environment is that the environment of an evolving analog network is still an analog network, and is therefore constituted by devices possessing terminals, some of which may be connected to the evolving analog network. Thus, in order to use a sequence-based device interaction map for establishing the connections, we only need to associate sequences with the terminals of the external devices that might possibly be connected to the evolving network. For example, the external devices for an analog electronic circuit might be a power supply and a resistive load, while the external devices for an artificial neural network controlling a robot might be the sensors and the actuators of the robot.

Note that not all terminals of the devices composing the external network must necessarily be given the possibility to connect to the evolving network. For example, the power supply of an electronic circuit or the sensing circuitry of a robot can be complex networks, with only a small subset of all the terminals of these networks considered as outputs that can be possibly connected to the powered electronic circuit or to the control system of the robot. On the other hand, the possibility for the evolving network to connect to some prespecified terminals of the external network must not be interpreted as a coercion to connect to all these terminals. Evolution must instead be left free to select which of those prespecified terminals are actually connected to the evolving network. This corresponds to leaving the possibility that a kind of elementary "feature selection" is performed on the raw collection of terminals available for connection, just like a living organism "selects" the external signals that it produces and to which it is responsive. A brief reflection shows that this possibility follows directly from the nature of the sequence-based mechanism of establishment of the connections

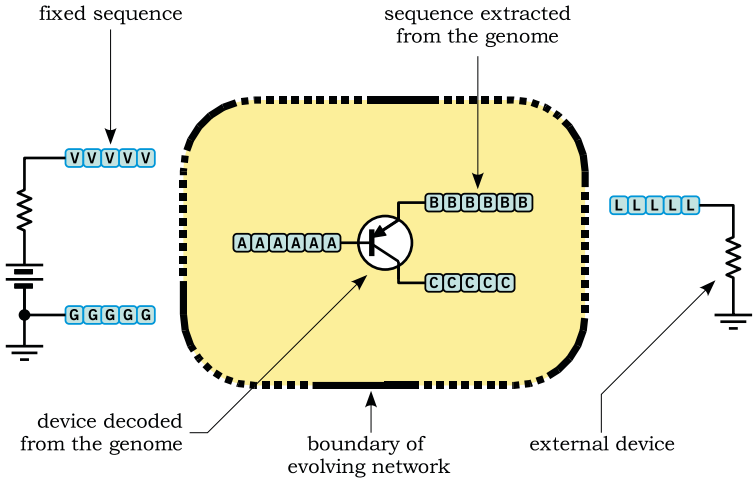


Figure 3.25: *The evolution of connections between the preassigned external network and the evolving network can be obtained associating a fixed sequence of characters defined by the experimenter with the terminals of the external devices. The device interaction map used to assign the strength of the interaction between the terminals of the evolving network can than be applied also to pairs formed with the fixed sequences. Note that not all the terminals of the devices composing the external network must be necessarily given the possibility to connect to the evolving network.*

between terminals, since the evolved interaction strengths include values corresponding to the absence of any direct connection[14] between two terminals. We can therefore conclude that to connect the evolving network to the external network in the spirit of the auxiliary evolvability conditions given in the previous chapter, we need only devise a strategy of association of sequences of characters with the terminals of the external network that must be connectible to the evolving network. As shown in the next sections, this can be done in many different ways.

3.5.1 Fixed sequences

The first solution that comes to mind to associate a sequence of characters with each external device terminal that must be given the possibility of connecting to the evolving analog network, consists in letting the experimenter define a collection of predefined fixed sequence and associate them with the connectible terminals (Figure 3.25). Once the fixed sequences are associated with the connectible terminals of the external devices, the connection strategy for the devices decoded from the genome which is based on the use of a device interaction map can be applied also to the external devices, and the evolution of these connections becomes possible.

Although at first sight very simple, this approach has the disadvantage of enlarging the collection of parameters that must be assigned by the experimenter. Moreover, it requires a certain caution in the choice of the sequences. For example, we must ensure that the structure of the fixed sequences permits the establishment of a connection with infinite conductance towards the external components, should this connection be required to achieve the desired functionality of the whole network. This means that the fixed sequences must ensure the possibility of exceeding the value of sequence alignment score that is associated with the maximum finite interaction strength value by the network specific interaction map. As explained in the previous section, this possibility depends, among other things, on the choice of the scoring matrices, and requires an adequate length of the sequences involved (remember from Subsection 3.4.5 that if the maximum substitution score is c_{max}, the maximum alignment score that can be realized by a sequence of length l is $c_{max}\, l$). It follows that the choice of the fixed sequences can be a complicated task, which opens the door to the suspicion that a possible failure of the evolutionary process is due to a poor choice of the fixed sequences associated with the external components.

3.5.2 Transducers

The attribution by the experimenter of fixed sequences of characters to some terminals of the external devices described in the previous section, entails another potential problem. Each sequence must be capable of generating all the interactions required by the evolved network in order to display the expected functionality. These interactions

[14] See footnote 4 on page 69.

can be manifold for some of the external connected terminals. For example, when a power supply is connected to an electronic circuit, its terminals are typically connected to many of the devices that form the circuit. Obviously, we can hardly expect the experimenter to anticipate the number of connections that each external terminal needs to establish, and to design the associated sequences accordingly. This difficulty is not so great as might seem at first sight, since evolution can overcome it by establishing a connection between the external terminal and a terminal of a device belonging to the evolving circuit, and then use the terminals of that device to extend the connection to the rest of the evolving circuit.

This kind of evolved "device bridging" function, however, can be facilitated by defining a special *transducer device* (or simply *transducer*) whose explicit role is the establishment of a connection linking the external world to the evolved system (Figure 3.26). As for the other devices of the network, the transducer is identified in the genome by a specific token, and is characterized by one terminal that is considered as belonging to the external network, and one that is considered as belonging to the evolving network. The association of sequences of characters extracted from the genome proceeds exactly like for the other devices, including the possibility of associating more than one sequence of characters to one terminal. An advantage of explicitly defining this new kind of device is that by duplicating and mutating the representation of an existing transducer in the genome – which, as explained in the next section, is easily done with the available genetic operators – several independent connections can be easily established with an external device.

3.5.3 Evolvable compartmentalization

The action of the transducer devices described above bears some similarity to the function of a trans-membrane protein transmitting a signal across the membrane that separates a cell from the surrounding environment, or across the membrane of some subcellular compartment or organelle (Lewin, 2004; Krauss, 2003). The sequences of characters associated with the terminals of a transducer belonging to the external network and to the evolved network have a correspondence in the distinct domains that the trans-membrane protein has on the two sides of the membrane. The sequences of characters of a transducer are encoded in the genome and, correspondingly, the characteristics of the

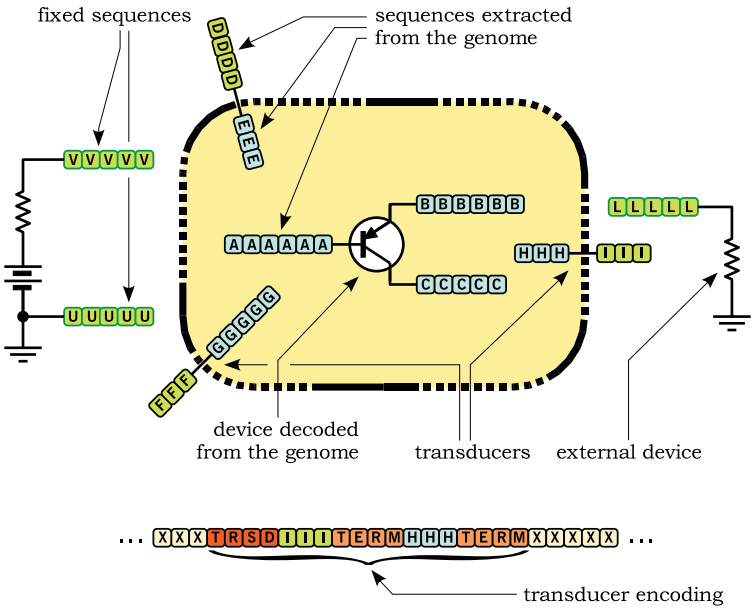


Figure 3.26: *The evolution of the connections between the preassigned devices of the external network, and those of the evolving network can be facilitated defining specialized transducer devices. A transducer, like any other device, is represented in the genome using a specialized token. In the fragment of genome shown here (bottom) the token* `TRSD` *identifies the transducer, and tokens* `TERM` *delimit the sequences of characters that must be associated with the terminals of the transducer. Contrary to the other devices encoded in the genome, however, one of the terminals of a transducer belongs to the evolving network, while the other belongs to the external circuit, thus permitting the establishment of a connection between the two environments.*

domains of a trans-membrane protein are encoded in the genome of the cell, and both can be induced by evolution to adapt to the fixed nature of the external signals.

Note that trans-membrane proteins and the other mechanisms of signal transduction (Krauss, 2003; Lewin, 2004) are used by cells not only to exchange signals with their non-living external environment,

but also to exchange signals between different compartments within a cell, and to communicate with other cells of a cellular colony or of a multicellular organism. In the same way, transducer devices can be used to evolve compartmentalized or "multicellular" analog networks. For example, a simple way of obtaining a compartmentalized network would be to assume each artificial chromosome as specifying a separated compartment of the network.

Another approach, which leaves more freedom to evolution in determining a compartmentalized network was anticipated in subsection 3.4.6 in the context of the discussion on interaction silencing. The idea is to consider device terminals for connection only if the associated sequences comply with some specific condition, for example, the presence in the sequences associated with the terminals of a common motif defining the terminals as belonging to the same compartment. This approach differs from that based on transducers in that the motifs that determine the compartments are evolved rather than predefined like the transducer tokens.

Figure 3.27 illustrates a possible implementation of this idea. The letters of a subset of the genetic alphabet are defined as having a large negative substitution score relatively to all the letters of the alphabet (including the letter itself) in the local sequence alignment scoring matrices. When the sequence associated with a terminal contains a substring of sufficient length composed of letters in this special subset and matching exactly a substring in the sequence associated with another terminal, the two terminals are considered as belonging to the same compartment. In this case the sequence interaction determined by the local alignment score of the sequences is calculated, otherwise the corresponding terminals are assumed as not directly connected. The interaction between two distinct compartments is brought about by devices that have terminals (either distinct or not) belonging to more than one compartment.

The use of the exact matching of substrings of characters that are defined as badly matched relatively to the local alignment parameters, ensures that these substrings will not produce a positive value of interaction strength according to the sequence interaction map. Thus, the sequence interaction determined by the local alignment algorithm and the mechanism of evolvable compartmentalization can coexist with minimal reciprocal disturbance. Note that a simple variant of the exact substring matching algorithms (which have linear computational complexity relatively to the length of the sequences) can be used to

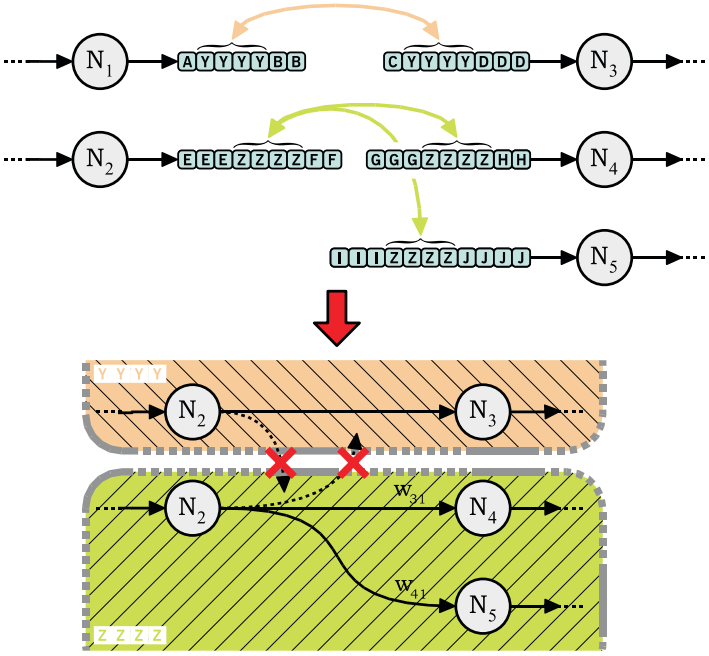


Figure 3.27: *A mechanism of evolvable compartmentalization of the network can be based on the presence of evolvable motifs contained in the sequences associated with the terminals of the devices, rather than predefined genome tokens. Two terminals are considered as belonging to the same compartment only if the sequence associated with one terminal contains a substring of sufficient length composed of letters belonging to a special subset of the genetic alphabet and matching exactly a substring in the sequence associated the other terminal. In this case the sequence interaction determined by the local alignment score of the sequences is calculated and the connection possibly established, otherwise the corresponding terminals are assumed as not directly connected.*

check for the condition of membership of the same compartment, before the sequence interaction mapping (which has quadratic computa-

tional complexity) is possibly applied to the sequence. The presence of a genetic operator of genome fragment duplication facilitates the evolutionary reorganization of the compartmentalized network structure, although some bootstrap problem can be envisaged and could require the seeding of the initial population genomes with suitable substrings. The issue of compartmentalization and of the modular structure of the evolved networks will be considered again from a more abstract point of view in Chapter 5.

3.5.4 I/O ports

The approach to the establishment of a connection between distinct compartments that is based on transducers requires the definition of the fixed sequences that must be associated with the connectible terminals of the external network. When the external network and the evolving network are the only existing compartments, there is the possibility of coalescing the fixed sequences and the transducers into a unique device, which we will call *I/O port* (Figure 3.28). The idea is that the connectible terminal of the external device and the transducer terminal belonging to the external network are assumed as always being connected through a wire. Hence, only one of the transducer sequences remains and must be extracted from the genome, whereas the fixed sequence associated with the terminal of the external device disappears, and no longer needs to be defined.

Note that now each external node that is assumed as connectible to the evolving circuit must be associated with a particular I/O Port device, and must therefore have a distinct device token. For example, the external circuit shown in Figure 3.28 has three terminals marked with the labels A, B, and C which are assumed as connectible, and this requires the definition of three tokens `IOPTA`, `IOPTB`, and `IOPTC`. In the decoding of a genome into an analog network, the fragments of genome delimited by the I/O port tokens and by the terminal tokens are extracted and associated with the correspondingly labeled terminals of the external devices. At this point, the external devices are considered as devices belonging to the evolving network and the device interaction map is applied to determine the strength of the interaction between all pairs of terminals that have a sequence of characters associated with them.

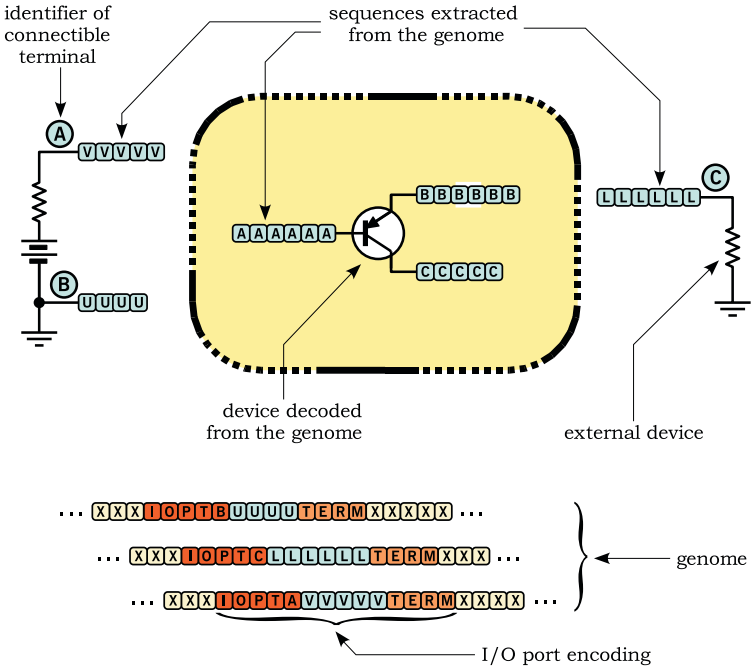


Figure 3.28: *An example of application of the technique of connection of external devices to the evolving circuit by means of* I/O *ports associated with external terminals marked as connectible to the decoded circuit. The fragment of genome represented above contains the tokens* `IOPTA`*,* `IOPTB` *and* `IOPTC` *corresponding to the three connectible external terminals. The fragments of the genome delimited by these tokens and by the* `TERM` *token are associated to the terminals of the external devices.*

3.6 Genetic operators

The genetic representation for analog networks described in this chapter permits the execution of many kinds of reorganizations of the genome without compromising its decodability. The only practical limit that must be imposed on the genome concerns the length of sequences associated with the terminals and the parameters of the devices, in order

to ensure that the sequence alignment algorithm that constitutes the core of the device interaction map remains executable in reasonable time and memory bounds. Apart from this limit, the fragments of the genome coding for components can be of variable length and be located anywhere within the genome. The genome itself is obviously a variable length genome. This opens the way to the introduction in our evolutionary algorithms of several genetic operations that are seldom used in artificial evolution experiments (although they are known to be common in the evolution of biological genomes (Graur and Li, 2000; Shapiro, 2002)). In particular, the genetic representation introduced above allows

- Operations on single nucleotides, such as insertion, deletion, and substitution.
- Operations on chromosome fragments, such as duplication, duplication of complement, deletion, transposition, and insertion of device descriptors.
- Operations on chromosomes, such as duplication and deletion of single chromosomes, and crossover (reciprocal recombination (Graur and Li, 2000, p. 29)) of pairs of chromosomes..
- Operations on the whole genome, such as duplication and trimming.

Note that from the point of view of the genetic operators each chromosome is just a sequence of characters where the tokens for devices, terminals and parameters have no special meaning. Therefore, the tokens are *not* protected from the action of the genetic operators, whose action can invalidate any device descriptor present in the genome, making that particular descriptor undecodable.

Let us now consider one by one the genetic operators listed above.

3.6.1 Single nucleotide mutations

Nucleotide insertion A random character belonging to the genetic alphabet is inserted, with probability p_{ni}, between each pair of adjacent nucleotides existing in the genome.

Nucleotide deletion Each nucleotide in the genome is deleted with probability p_{nd}.

Nucleotide substitution Each nucleotide in the genome is substituted, with probability p_{ns}, with a random character belonging to the genetic alphabet.[15]

3.6.2 Chromosome fragment mutations

Chromosome fragment duplication For each chromosome, with probability p_{f2}, two random points are chosen and the intervening genome fragment is duplicated and the duplicate is inserted at a randomly chosen point of a randomly chose chromosome belonging to the genome.

Complemented chromosome fragment duplication For each chromosome, with probability p_{fc}, two random points are chosen and the intervening genome fragment is duplicated; the duplicate is then complemented and inserted at a randomly chosen point of a randomly chose chromosome belonging to the genome.

Chromosome fragment transposition For each chromosome, with probability p_{ft}, two random points are chosen and the intervening genome fragment is transferred at a randomly chosen point of a randomly chosen chromosome belonging to the genome.

Chromosome fragment deletion For each chromosome, with probability p_{fd}, two random points are chosen and the intervening genome fragment is deleted.

Device insertion For each chromosome, with probability p_{di}, the descriptor of a device randomly selected in the device set is inserted at a randomly chosen point. The sequences of characters associated with

[15] The most direct (and most frequently adopted) technique for the implementation of single nucleotide mutations on a computer consists in generating a pseudo-random bit biased according to the required mutation probability for each nucleotide insertion, deletion, and substitution. When the mutation probability p is very small and the length n of the genome is large, however, this direct approach leads to the generation of an inordinate amount of calls to the routine producing the pseudo-random bits, with only a few calls leading to an actual nucleotide mutation. To avoid wasting computational resources on "useless" calls to the pseudo-random bit routine, it is preferable in this case to generate for each kind of nucleotide mutation a pseudo-random integer m drawn from a binomial distribution of n trials, each of probability p (for example, Press et al., 1992, p. 295), and then to generate m random integers to be interpreted as the indexes of the nucleotides that must be mutated. Although the possibility of generating the same index more than once leads to a mutation probability different from p, the difference is usually negligible for small values of p.

the terminals and the evolvable parameters of the devices can be randomly generated or can be obtained by sampling the sequences associated with the terminals of the devices existing in the genome.

3.6.3 Whole chromosome mutations

Chromosome duplication Each chromosome is duplicated with probability p_{c2}. The duplication can either append to the chromosome a copy of itself, or create a new chromosome.

Chromosome deletion Each chromosome is deleted with probability p_{cd}.

Crossover (reciprocal recombination) In the simplest implementation of the crossover operator, the genomes of two individuals (the "parents") are first compared. If the number of chromosomes in the two genomes is the same, the chromosomes are paired and, for each pair, with probability p_{cx}, one point is randomly chosen within each chromosome, and the fragments of chromosomes thus determined are swapped to generate the new chromosomes. The limit of this kind of crossover operator is that, when applied to chromosomes that – like those of the genome used in this work – do not have a fixed structure and length, it tends to produce macro-mutations rather than the recombination of homologous chromosome fragments (Harvey, 1995). It is therefore advisable to redefine the crossover operator so as to favour the recombination of fragments of chromosome that can be considered homologous.

A possible candidate for this role of homologous crossover operator starts by choosing randomly one tentative crossover point within one of the chromosomes that must be recombined. A fragment of chromosome of a predefined length l_m belonging to the neighborhood of the selected point is then considered as a template and a fragment sufficiently similar to the template is searched in the other chromosome. The search for a fragment similar to the template can start with the search of a fragment exactly matching a substring of the template and then evaluating the global alignment score of the whole template with the tentative matching region thus identified. If a fragment sufficiently similar to the template is found, a crossover point in this second chromosome corresponding in the template to the tentative crossover point previously selected in the other chromosome is identified, and the

fragments of chromosomes thus determined are swapped to generate the recombined chromosomes. Thus, the recombination is performed only in presence of a reasonable matching between the fragments of genome adjacent to the crossover point (Harvey, 1995). This finds a correspondence in the recombination of biological genomes, where the recombination of poorly matched DNA sequences is prevented by specialized molecular proofreading machinery (Alberts et al., 2002). Since the actual execution of the recombination requires the existence of a certain similarity between the paired chromosomes, this operation will be seldom performed if the population gene pool is highly heterogeneous. However, the selection built in the evolutionary process tends to rapidly reduce the diversity of the population and to produce a degree of genetic homogeneity that – both in the natural and in the artificial case – permits the actual accomplishment of most of the attempted recombinations of chromosomes (Harvey, 1995; Goldberg, 2002).[16]

3.6.4 Whole genome mutations

Genome duplication The whole genome is duplicated with probability p_{g2}. The duplication can either append to each existing chromosome a copy of itself, or create a new chromosome for each existing one.

Genome trimming The application of the genetic operators described so far can result in the transformation into non coding genome of fragments of genome previously coding for devices. The presence of this non coding genome does not prevent the decoding of the remaining genome, and can even conceivably play the role of an evolutionary useful repository of genetic fragments and pseudo-genes. To test this last hypothesis, it is useful to be able to make a comparison with the performances of a genome that is periodically freed from most of the non coding genome. To this end, a genome trimming operator is defined, which eliminates from the genome of an individual all the non coding genome except possibly a short fragment of predefined or random length that is retained in order to space the device descriptors and leave in the genome some non-coding genome that can be possibly recruited by evolution. At each generation the genome trimming operator is applied to each individual with probability p_{gt}, and to all individuals with probability p_{pt}. Note that the procedure of genome decoding automati-

[16] Figure 4.47 on page 186 permits a visual appreciation of the degree of homogeneity of an evolved population.

cally identifies the coding regions of the genome, and this information can be used to implement the genome trimming operation with minimal additional computational effort.

3.6.5 Generation of random populations

Although not strictly a genetic operator, the generation of a population of random individuals is an operation that is traditionally required to start the evolutionary process. This can be easily obtained by performing one or more operations of device insertion on a background of random nucleotides. A method of seeding evolution with nonrandom genomes derived from existing circuits will be briefly described in the next chapter.

3.7 Summary

In this chapter we have specified all the elements required to actually implement an evolutionary system for analog networks complying with the guidelines given in the previous chapter. We have defined a genetic representation for the devices of an analog network which tolerates major reorganizations of the genome; we have defined a way to determine and evolve the connections between the devices of the network and between the evolved network and the external network (or other evolved compartments and networks) which is based on an abstract definition of the interaction between fragments of genome; we have defined a set of genetic operators that can reorganize the genome and change the composition and the structure of the network, either gradually or abruptly. We have in particular specified a definite implementation of the interaction between fragments of the genome in terms of local sequence alignment (note, however, that other implementations of this mapping could be conceived in the framework of the approach proposed in this thesis). We are thus ready to proceed in the next chapter to the examination of the behavior of this evolutionary system and to its application to the evolution of analog networks.

Chapter 4

Experiments[1]

Overview

This chapter describes three series of evolutionary experiments conducted with the artificial evolutionary system introduced in the previous chapter. First, a series of ***sequence matching*** *experiments is devoted to the evolution of genomes realizing given local alignment scores relatively to a preassigned collection of sequences. In the analog network perspective these experiments give some information about the possibility and the effort required to evolve alignments between sequences of nucleotides, complementing the investigations about the properties of sequence alignment based on theoretical considerations that were made in the previous chapter. A series of* ***network matching*** *experiments extends this investigation to the case of alignments of sequences interpreted as interactions between the terminals of a collection of preassigned analog network devices, with the aim of generating the values of interaction strength that characterize simple analog networks of known functionality. These preliminary explorations are followed by a series of actual* ***analog network evolution*** *experiments, where the evolutionary process is required to synthesize analog networks realizing a preassigned functionality. The chapter closes with a short discussion on the results obtained, including a survey of other applications and experiments that can be performed with the system.*

[1] Parts of this chapter were published in (Mattiussi and Floreano, 2004b) and (Mattiussi and Floreano, 2004a).

4.1 Sequence matching experiments

The evolutionary system for analog networks described in the previous chapter is based on the association of sequences of characters extracted from the genome with the terminals of the devices that constitute the network. The interaction between the terminals of the devices is determined by a mapping from pairs of sequences of characters to values of interaction strength, and this mapping is based on the local alignment of the sequences.

An analog network is typically composed of several devices whose terminals are selectively pairwise connected. Given the devices of the network, the strength of the interactions between the terminals determines the behavior and the functionality of the network. This means that one of the attributes that endow our evolutionary system with the potentiality to synthesize analog networks with given functionality is the possibility to evolve collections of sequences having the required values of interaction.

In the previous chapter the properties of the local alignment of sequences were analyzed and the parameters of the sequence alignment algorithm were assigned so as to ensure the possibility to generate both small and large values of alignment score. In the present section we extend that analysis exploring the possibility to *evolve* preassigned values of alignment score. More precisely, our goal is now the evolution of a collection of sequences characterized by given alignment scores relatively to a collection of fixed sequences. In other words, the target of evolution is not the synthesis of an analog network but the generation of a collection of sequences that match in a predefined way a series of preassigned *target sequences*. The evolutionary experiments start with the assignment of the genetic alphabet and of the scoring matrices for the sequence alignment algorithm. Then, the number and length of the target sequences is chosen, along with the *target alignment score* for each target sequence. The goal of evolution is to generate a genome that matches the target sequences in the required way, so that the local alignment score of the genome relatively to the target sequences corresponds to the required target alignment scores (Figure 4.1).

4.1.1 Experiment 1: High-strength matching

The goal of Experiment 1 is the evolution of a genome that matches all the target sequences with alignment scores that are near the max-

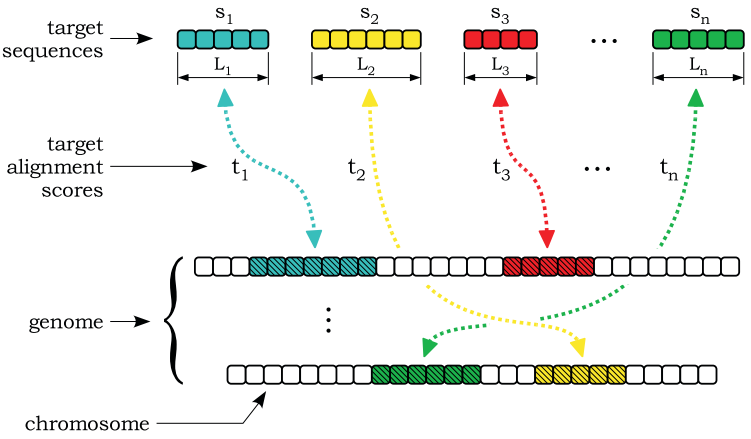


Figure 4.1: *In the series of sequence matching experiments the goal of the evolutionary process is the generation of a genome that realizes a set of preassigned alignment scores (called* target alignment scores*) relatively to a collection of preassigned target sequences.*

imum values achievable with the given length of target sequences and the given scoring matrices. In the analog network perspective, this corresponds to the evolution of a genome that has a "strong interaction" with all the target sequences. The idea is to verify the possibility of achieving such an interaction, and to estimate the number of generations required to obtain it.

Genetic alphabet The genetic alphabet is denoted by $\mathcal{G}$, and its size is denoted by $|\mathcal{G}|$. In the current experiment

$\mathscr{V}$ size of genetic alphabet $|\mathcal{G}| = 26$

Scoring matrices The scoring matrices used in this and all the subsequent experiments have the structure of the matrices shown in Figure 3.17 on page 89, that is, the substitution matrix is a symmetric circulant matrix, and the indel score is the same for all the letters of the genetic alphabet. We can therefore limit the representation of those matrices to the first row s_1 of the substitution matrix S and to a generic

element d_i of the indel vector $\mathbf{d}$, as illustrated in Figure 3.18 on page 91. In the current experiment

- first row of substitution matrix
  $\mathbf{s}_1 = (\ 5,\ 2,\ 1,\ 0,\ -1,\ -2,\ -5,\ \ \ldots\ \ -5,\ -2,\ -1,\ 0,\ 1,\ 2\ )$
- generic element of indel vector $d_i = -3$

Target sequences A collection of n_t target sequences of length l_i, $i = 1, \ldots, n_t$ is randomly generated[2] from the genetic alphabet. In the current experiment

- number of target sequences $n_t = 10$
- length of target sequences $l_i = 20$ for all i

Fitness function The fitness function of an individual is calculated as follows. The local alignment score of the i-th sequence relatively to each chromosome of the genome of the individual is calculated, and the maximum alignment score is assumed as representing the alignment score $\widetilde{t}_i$ of that particular target sequence relatively to the whole genome. This gives a vector $\widetilde{\mathbf{t}} = (\widetilde{t}_1, \ldots, \widetilde{t}_{n_t})$ that must be compared to the vector of the target alignment scores $\mathbf{t} = (t_1, \ldots, t_{n_t})$. The fitness of the individual is the negative of the distance between $\mathbf{t}$ and $\widetilde{\mathbf{t}}$ induced by the $\|\cdot\|_1$ norm, divided by the number of target sequences, that is,

$$f = -\frac{1}{n_t} \sum_{i=1}^{n_t} \left| t_i - \widetilde{t}_i \right|$$

This corresponds to a fitness that is negative when some of the targets are not matched as required, and is zero – the maximum fitness value – only when all the target alignment scores are attained. In the current experiment

- target alignment score $t_i = 70$ for all i

Note that with the given substitution matrix, which has a maximum substitution score corresponding to $s_{11} = 5$, the maximum alignment score that can be attained relatively to a target sequence of length $l_i = 20$ is $s_{11} \cdot l_i = 100$. Since only the alignment of a character with itself

[2] Henceforth, we use the shorthand expression "randomly generated" to mean "randomly generated with a uniform distribution of probability on the set of elements".

gives the maximum substitution score, this maximum alignment score is attained only if the genome contains a substring that exactly matches the target sequence. Asking evolution to achieve a value of $t_i = 70$, on the other hand, corresponds to requiring the generation of a value that is close enough to the maximum attainable alignment score to make evolution interesting, but without imposing an exact matching and the corresponding absence of redundancy.

Evolutionary algorithm and parameters The evolutionary algorithm used in the current experiment is a standard generational genetic algorithm that uses tournament selection, elitism (Bäck, 1996; Bäck et al., 2000a,b), and the genetic operators described in the previous chapter. The actual values of the parameters are given below. Parameters corresponding to probabilities of genetic operators that were mentioned in the previous chapter and are not listed here are assumed as having zero value.

✓ population size	$n_p = 100$
tournament size	$n_r = 5$
elite size	$n_e = 1$
✓ probability of nucleotide substitution	$p_{ns} = 0.001$
probability of nucleotide insertion	$p_{ni} = 0.001$
probability of nucleotide deletion	$p_{nd} = 0.001$
✓ probability of fragment duplication	$p_{f2} = 0.001$
probability of fragment deletion	$p_{fd} = 0.001$
probability of fragment transposition	$p_{ft} = 0.001$
✓ probability of chromosome duplication	$p_{c2} = 0.001$
probability of chromosome deletion	$p_{cd} = 0.001$
✓ probability of genome duplication	$p_{g2} = 0$
✓ probability of chromosome crossover	$p_{c2} = 0.1$
number of matching characters for crossover	$l_m = 10$

Initial population Each individual of the initial population is generated with a genome composed of n_{ic} chromosomes of length l_{ic} randomly generated from the genetic alphabet. In the current experiment

✓ number of chromosomes in the initial genome	$n_{ic} = 10$
✓ length of chromosomes in initial genome	$l_{ic} = 20$

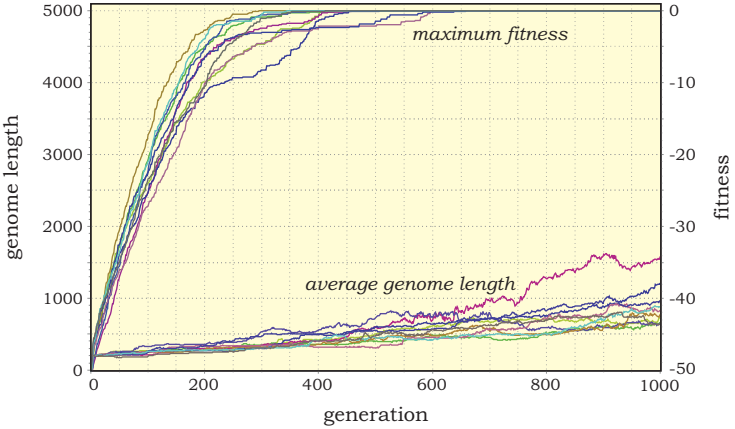


Figure 4.2: *The result of ten repetitions of Experiment 1, aimed at the evolution of a genome characterized by high local alignment scores relatively to a collection of preassigned target sequences. The maximum fitness curves show that all the runs eventually attained a fitness of zero, corresponding to the evolution of a genome matching the target sequences in the required way. The curves of the average genome length witness the action of the genetic operators and the corresponding variation of the genome length during evolution.*

Results Figure 4.2 shows the results of a series of ten repetitions of this evolutionary experiment. The curves of the maximum fitness show that with the given parameters, the evolutionary system was consistently able to evolve a genome achieving the required local alignment scores relatively to the target sequences. The number of generations required on average to evolve the required alignment scores is of the order of some hundreds. This is a first sign that the evolutionary system based on sequence alignment presented in this thesis cannot be expected to produce significant results in just a few generations, although, as is suggested by the results of Experiment 7 below, increasing the population size can help to reduce the required number of generations. We must realize that the probability of the simultaneous occurrence of multiple beneficial nucleotide mutations on a genome can be expected to be very small (Conrad, 1990). Consequently, we can expect to observe phases of the evolutionary process where the

progress of the best individual proceeds at the pace of one "improved nucleotide" per generation. Of course, the existence of operators that alter the genome more radically – in particular the presence of operators of recombination of the genetic material of distinct individuals – gives evolution the possibility to proceed on average at a faster pace, especially if the size of the population is not too small.

The curves of average genome length[3] shown in Figure 4.2 reveal a collective upward trend. This is not surprising in view of the fact that the sequence alignment is local and, therefore, the addition of genome fragments that do not belong to the regions matching the target sequences has no effect on the alignment and, consequently, on the fitness. In particular, the duplication of a chromosome does never perturb the fitness, whereas the deletion of a chromosome has typically a negative impact on the fitness, except in those cases where the deleted chromosome does not contain a matching region. For this reason, we cannot judge from this experiment if the presence of operators of genome duplication plays a role in the attainment of the required matching that is observed in all runs of the experiment, or if the increase in the genome length is simply a consequence of this asymmetry of the duplication and deletion operators relatively to their impact on the fitness. Experiment 3 will be devoted to the analysis of this question.

4.1.2 Experiment 2: Mixed-strength matching

Experiment 1 was aimed at the evolution of a genome "interacting" strongly with the target sequences. As discussed in the previous chapter (Subsection 3.4.5), apart from the evolution of strong interactions we must also be concerned with the possibility of avoiding unwanted interference between sequences. This corresponds to the possibility of achieving small sequence alignment scores for all the pairs sequences that must not interact. In this perspective, the goal of Experiment 2 is the evolution of a genome that matches half of the target sequences with alignment scores that are near the maximum values achievable with the given length of target sequences and the given scoring matrices, and the remaining half with alignment scores that are near the minimum feasible value of alignment score. In the analog network

[3]The average genome length and average number of devices decoded from the genome mentioned here and in the subsequent sections refer to values calculated by averaging at each generation over the individuals of the population.

Figure 4.3: *The result of ten repetitions of Experiment 2, aimed at the evolution of a genome characterized by large local alignment score relatively to half of the collection of target sequences, and by a small local alignment score relatively to the remaining target sequences. In the analog network perspective, where the value of sequence alignment is transformed into a value of interaction strength between terminals, this corresponds to the ability of evolving some strong interaction with some terminals, while keeping low the interference with other terminals. The maximum fitness curves show that this kind of mixed strength interaction can indeed be evolved, since all the runs eventually attained a fitness of zero, corresponding to the evolution of a genome matching the target sequences in the required way.*

perspective, this corresponds to trying to evolve a genome that has a "strong interaction" with half of the target sequences but a "weak interaction" with the remaining half. This experiment uses the same parameters as Experiment 1, except for the target alignment scores, which are assigned the following values

$$\text{target alignment scores} \qquad \begin{array}{ll} t_i = 20 & \text{for } i = 1, \ldots, 5 \\ t_i = 70 & \text{for } i = 6, \ldots, 10 \end{array}$$

Results Figure 4.3 shows the results of a series of ten repetitions of this evolutionary experiment. The graphs reveal that evolution proceeds in much the same way as in Experiment 4.2. This means that the

evolutionary system can evolve a genome realizing strong interactions with a collection of preassigned sequences while maintaining a low interference relatively to another collection of preassigned sequences.

The comparison of the maximum fitness curves of Figure 4.3 with those of Figure 4.2 reveals that the average number of generations needed to attain the required mixed-strength matching is smaller than that required by the high-strength matching, suggesting that the attainment of high alignment scores is evolutionarily more challenging than the control of the randomly generated alignment scores. We must note, however, that the value $t_i = 20$ assigned as low-valued target alignment score has a relatively small probability of being obtained from pairs of randomly generated sequences of length 20, given the genetic alphabet and the scoring matrices used in this experiment. This is suggested by the histogram of the estimated probability distribution of randomly generated alignment scores for the same evolutionary setup shown in Figure 3.20 on page 95. This histogram indicates that, especially with target sequences of length $l_i = 20$, the peak of randomly generated scores can be expected to be attained for values smaller than $t_i = 20$. Therefore, with the given choice of the genetic alphabet and scoring matrices, this alignment score appears as relatively protected from the effect of random interferences and, in any case, in a range that permits evolution to effectively counter the existing random interferences. Although these conclusions apply only to this admittedly very simple evolutionary setup, we will tentatively adopt this value as the alignment score i_{min} associated with the minimum non-null interaction strength in the experiments of network matching and network evolution described in the forthcoming sections.

4.1.3 Experiment 3: Disabling duplication

It was noted in commenting the results of Experiment 1 that the observed presence in the experiments of an upward trend in the average genome length does not in itself imply that the operations of genome duplication are actually required to successfully evolve a genome realizing the required matching with the target sequences. To explore this issue, the current experiment replicates Experiment 1 but with the duplication operators disabled. In order to keep balanced the rate of operators that add and the rate of operators that subtract nucleotides from the genome, the transposition and deletion operators are also disabled. This corresponds to assigning

✓ probability of fragment duplication	$p_{f2} = 0$
probability of fragment deletion	$p_{fd} = 0$
probability of fragment transposition	$p_{ft} = 0$
✓ probability of chromosome duplication	$p_{c2} = 0$
probability of chromosome deletion	$p_{cd} = 0$

All the other parameters have the values that were assigned in Experiment 1.

Results Figure 4.4 shows the results of ten repetitions of the experiment with the duplication operators disabled. The bottom panel shows a detail of the maximum fitness curves, revealing that in most runs the evolutionary process gets stuck on a less than optimal value of fitness. Note that the required matching was not achieved in a number of generations that is four times that considered in the repetitions of Experiment 1 reported in Figure 4.2. Apparently, with the genetic duplication operators disabled evolution is unable to produce a genome matching the target sequences with the required alignment score.

To ascertain the reason of the phenomenon revealed by Figure 4.4, the genome of the best individuals of the apparently stalled runs was examined. The analysis revealed that the cause of the evolutionary standstill is the overlap of matching regions of the genome relative to distinct target sequences. Figure 4.5 illustrates a typical example of this phenomenon. Here, two chromosomes are committed with largely overlapping matching regions to the matching of two target sequences each. The result is that, beyond a certain point, every improvement in the alignment of the multiply matched chromosomes relatively to one of the target sequences that they match, is detrimental to their alignment relatively to the other matched target sequence. The phenomenon is represented schematically in Figure 4.6 (top). Note that a similar phenomenon can be observed in biological genomes, where it is called *gene overlap* and where it is reputed to slow the rate of evolution (Graur and Li, 2000, p 294).

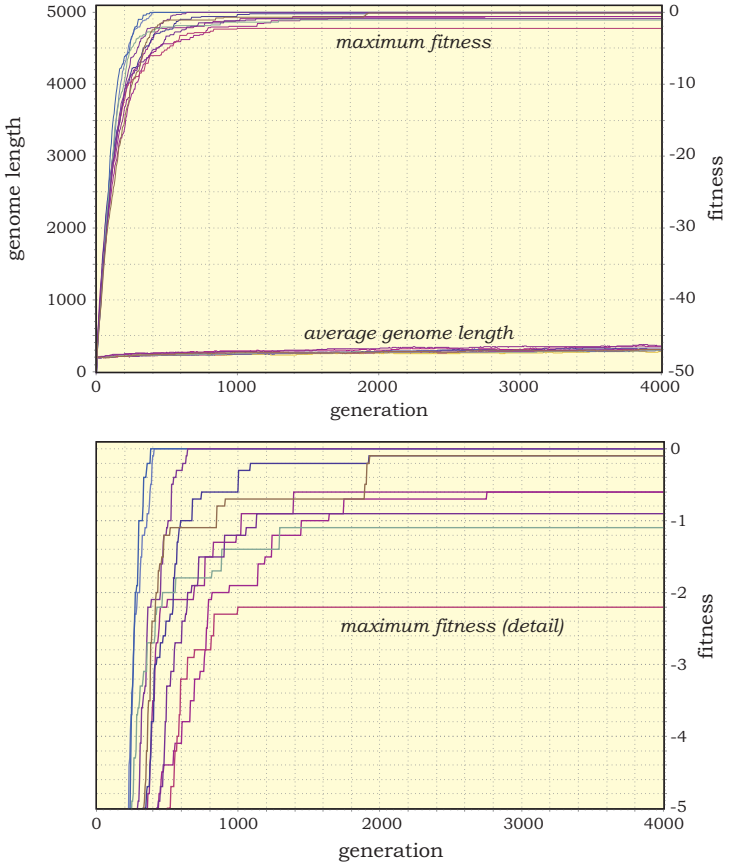


Figure 4.4: *The result of ten repetitions of Experiment 3. This experiment has the same goal as Experiment 1 (whose results are documented in Figure 4.2), except that the genome duplication operators that were active in Experiment 1 are now disabled. Although the number of observed generations was increased fourfold relatively to the runs shown in Figure 4.2, the maximum fitness curves reveal that with duplication disabled evolution apparently stalls without realizing the required matching with the target sequences.*

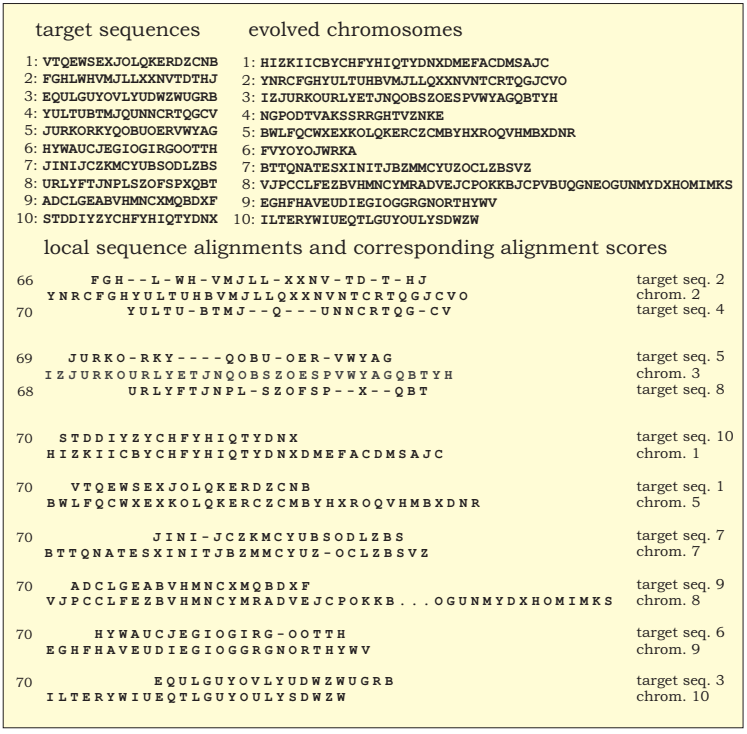


Figure 4.5: *An example of genome evolved to match a collection of target sequences with the operators of genome duplication disabled. The figure also shows the alignment of the chromosomes relatively to the target sequences, along with the corresponding alignment score. Here, chromosomes 2 and 3 are committed to the matching of two target sequences each, and the matching regions for the two overlap. The result is that evolution gets stuck without attaining the required alignment score (70 in this particular case) relatively to some of the target sequences, since the improvement of the matching of these chromosomes relatively to one of the target sequences that they match degrades the matching relatively to the other target sequence. As a result of the double matching of chromosomes 2 and 3, chromosomes 4 and 6 are not involved in the realization of the target sequence matching and do not appear in the alignments show in the figure. Note that these chromosomes are shorter than the other chromosomes, probably as a consequence of their being free from any evolutionary constraint.*

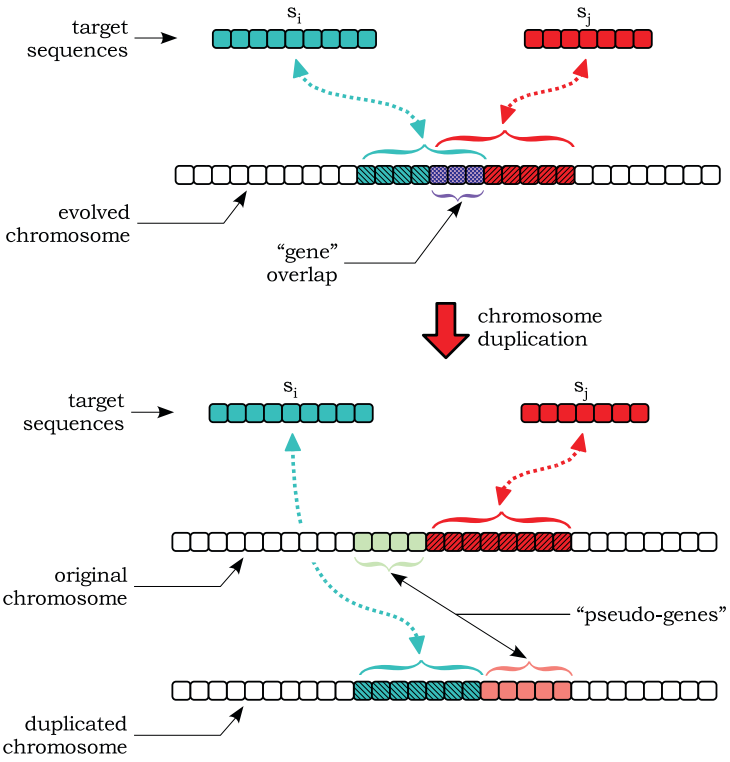


Figure 4.6: *A schematic representation of the phenomenon of gene overlap that can be observed in the evolved genome shown in Figure 4.5. In gene overlap, two distinct genes share a fragment of genome, so that any mutation in this fragment has potentially an effect on the function of both genes. In the case of sequence matching, gene overlap corresponds to a fragment of genome that is required to match two or more target sequences. The consequence is that when only the region of the genome that is in common remains to be evolved in order to achieve the required matching, the improvement of the matching relatively to one sequence can degrade the matching relative to the other. This fact can slow evolution and eventually bring it to a standstill. Gene overlap can be resolved by duplication of the genome fragments containing the overlapping genes followed by specialization of each gene for a distinct function (bottom). Typically the fate of the instances of the duplicate regions that are no longer used for the matching is to degenerate into "pseudo-genes".*

4.1.4 Experiment 4: Re-enabling duplication

Figure 4.6 (bottom) illustrates schematically how the presence of operators of genome duplication can solve the problem of gene overlap observed in Experiment 3. The duplication of the genome fragment involved in more than one matching creates two distinct instances of each matching region. The evolutionary process can then operate independently on the two instances, and make them evolve to match only one target sequence. To corroborate this hypothesis, in this experiment we continue the evolutionary runs of the previous experiment but re-enabling the duplication, transposition and deletion operators that where previously disabled, by putting

✓ probability of fragment duplication	$p_{f2} = 0.001$
probability of fragment deletion	$p_{fd} = 0.001$
probability of fragment transposition	$p_{ft} = 0.001$
✓ probability of chromosome duplication	$p_{c2} = 0.001$
probability of chromosome deletion	$p_{cd} = 0.001$

All the other parameters are kept unchanged from Experiment 3, reinstating thus fully the setup of Experiment 1.

Results Figure 4.7 shows that after the re-enabling of the duplication operators, evolution is no longer stalled and the required matching of the target sequences from the part of the genome is rapidly achieved in all the evolutionary runs. Inspection of the evolved genomes confirmed that the mechanism behind this phenomenon is indeed the one schematized in Figure 4.6. This corroborates the hypothesis that the presence of certain genome reorganization operators is instrumental to the success of an evolutionary string alignment process.

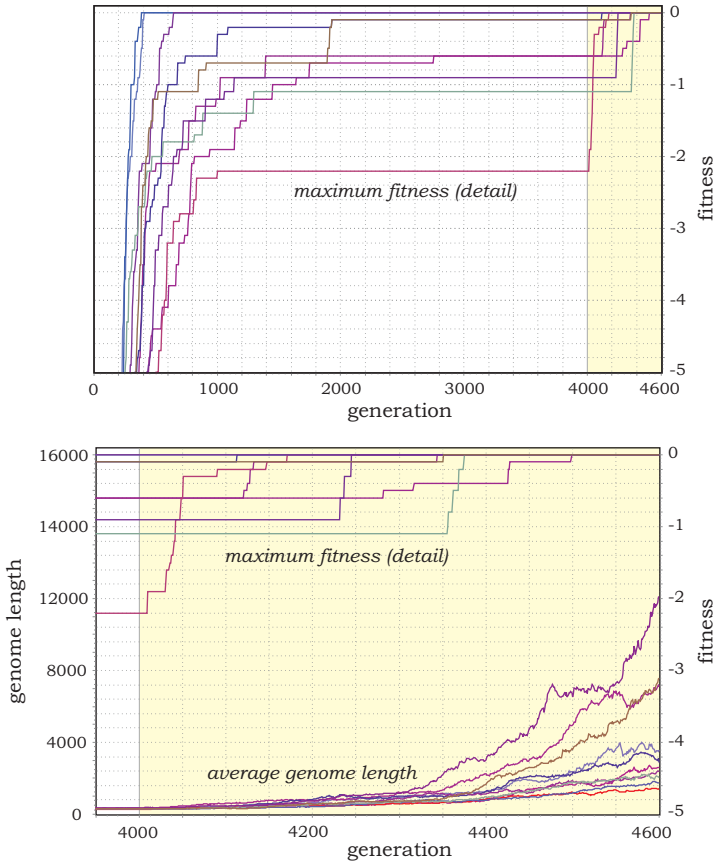


Figure 4.7: *Experiment 4 continues Experiment 3 (whose results are reported in Figure 4.4) but with the operators of genome doubling re-enabled. Up to generation 4000 the curves shown are those of Experiment 3. Genome duplication is disabled and several runs are stuck with less than the optimal null fitness as a consequence of gene overlap. At generation 4000 duplication is enabled and the new part of the experiment starts. The detail of the maximum fitness curves shown in the plots show that all the runs in which evolution was stalled in the absence of genome duplication are rapidly brought to success in the presence of genome duplication.*

4.1.5 Experiment 5: Varying the number of target sequences

Experiment 5 is meant to obtain an idea of the influence on the evolutionary process of the number of sequences that must be matched. Experiment 1 is repeated with all the parameters unchanged, except for a doubling from 10 to 20 of the number of target sequences, which gives

$$\text{number of target sequences} \qquad n_t = 20$$

and a corresponding doubling from 10 to 20 of the number of chromosomes in the individuals of the initial population

$$\text{number of chromosomes in the initial genome} \qquad n_{ic} = 20$$

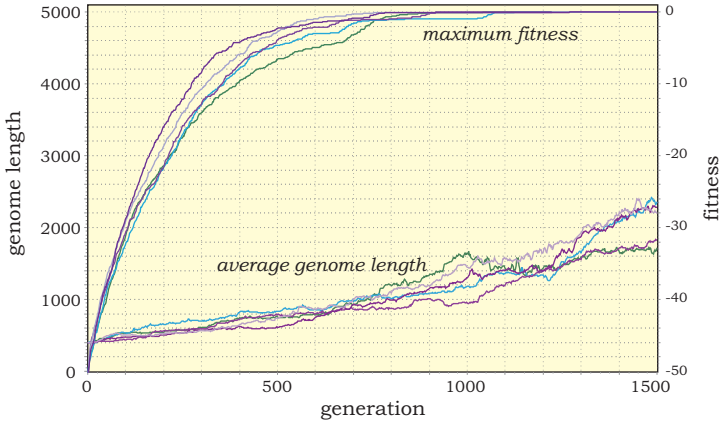


Figure 4.8: *The result of five repetitions of Experiment 5. This experiments has the same parameters and goal as Experiment 4.2 (whose results are documented in Figure 4.2), except that the number of target sequences is doubled with respect to Experiment 4.2. The maximum fitness curves show that all the runs eventually attained a fitness of zero, corresponding to the evolution of a genome matching the target sequences in the required way, although – not unexpectedly – more generations were on average required to obtain the desired result relatively to Experiment 1.*

Results Figure 4.8 shows the result of five repetitions of this evolutionary experiment. Comparing these curves with those relative to Experiment 1 (Figure 4.2) reveals that evolution is still successful in all the repetitions, and that the course of the evolution is similar to that observed in Experiment 1, except that – as could be expected – the average number of generations required to achieve the required matching has increased by some hundred generations due to the greater number of target sequences that must be matched by the evolved genome.

4.1.6 Experiment 6: Varying the length of the target sequences

The aim of Experiment 6 is to estimate the influence on the evolutionary process of the length of the sequences that must be matched. Experiment 1 is repeated with all the parameters unchanged, except for a doubling from 20 to 40 characters of the length of the target sequences and a corresponding doubling from 20 to 40 characters of the length of the chromosomes in the initial genome, and from 70 to 140 of all the target alignment scores. This gives

☑ length of target sequences	$l_i = 40$	for all i
☑ target alignment score	$t_i = 140$	for all i
☑ length of chromosomes in initial genome	$l_{ic} = 40$	

Results Figure 4.9 shows the result of five repetitions of this evolutionary experiment. The graphs show that evolution is still successful in all the repetitions of the experiment. Comparing these curves with those relative to Experiment 1 (Figure 4.2) reveals that – as could be expected – the average number of generations required to match the longer target sequences with larger target scores used in the current experiment, has increased relatively to Experiment 1, due to the presence of longer target sequences and higher target alignment scores that must be attained.

Summing up, Experiment 6 and Experiment 5 show the same qualitative behavior in the evolutionary process as Experiment 1, despite a doubling in the number or in the length of the target sequences and in the target alignment scores.

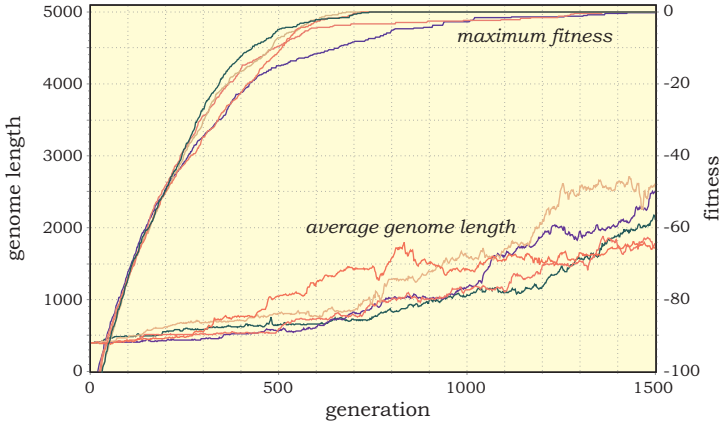


Figure 4.9: *The result of five repetitions of Experiment 6. This experiments has the same parameters and goal as Experiment 1 (whose results are documented in Figure 4.2), except that the length of the target sequences and the target alignment scores is doubled with respect to Experiment 1. The maximum fitness curves show that all the runs eventually attained a fitness of zero, corresponding to the evolution of a genome matching the target sequences in the required way. As could be expected, more generations were on average required to obtain the required matching relatively to Experiment 1.*

4.1.7 Experiment 7: Varying the population size

Experiment 7 investigates the influence of the population size on the evolutionary process. Experiment 1 is repeated with all the parameters unchanged, except for a tenfold increase in population size, which becomes

$$\text{population size} \qquad n_p = 1000$$

Results Figure 4.10 shows the result of five repetitions of this evolutionary experiment. As could be expected from the success of the experiments performed with a smaller population (Experiment 1), evolution achieves the desired result in all the repetitions of the experiment. Comparing these curves with those relative to Experiment 1 (Figure 4.2)

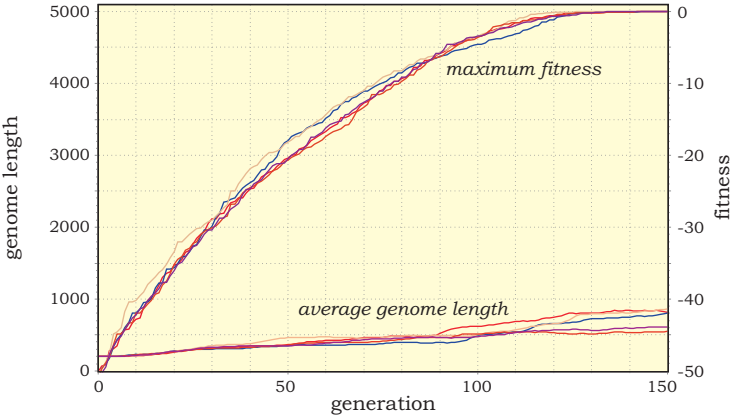


Figure 4.10: *The result of five repetitions of Experiment 7. This experiments has the same parameters and goal of Experiment 4.2 (whose results are documented in Figure 4.2), except that the number of individuals in the evolving population is increased tenfold with respect to Experiment 4.2. The maximum fitness curves show that the use of a larger population results in a significant decrease of the number of generations required to obtain the desired matching relatively to Experiment 1.*

reveals that the average number of generations required to achieve the required matching has decreased significantly. In other words, the size of the population seems to influence appreciably the length of the evolutionary runs required to attain a given sequence alignment score. Consequently, it can be expected to influence also the speed of the evolution of analog networks, although in that case the evolution of the sequence alignments is only one of the factors determining the rate of evolution and not necessarily the critical one.

4.1.8 Discussion

The goal of the sequence matching experiments reported in this section was the collection of information about the process of local sequence alignment in an evolutionary context. The setup was conceived so as to have some similarity with the actual sequence alignment that we can

expect to be required for the evolution of analog networks. Yet, the evolutionary setup and goal were chosen to be very simple, in order to avoid all the complexities of an actual analog network evolutionary experiment and simplify the interpretation of the results.

The results obtained in this series of experiments confirm that the values of genetic alphabet size and the scoring matrices that were obtained in the previous chapter on the ground of theoretical considerations, appear indeed capable of endowing the evolutionary system with the possibility of attaining the basic functionalities that are expected from it, namely, producing large alignment scores while keeping at bay unwanted interactions. Another interesting result is the observation that the presence of certain genome reorganization operators such as the operators of duplication of genome fragments is instrumental to the success of an evolutionary sequence alignment process. Still another information that we obtained from this series of experiments concerns the number of generations that we can expect to be necessary for the evolution of analog networks using population sizes of the same order of magnitude as those employed here. Given that the evolution of an analog network involves the selection of the devices composing the network, in addition to the generation of the connections between their terminals mediated by the sequence alignment, we can expect evolution times comparable or greater than those observed here.

Many more experiments could have been performed to probe in a systematic way the behavior of the evolutionary system relatively to the sequence alignment task. However, in consideration of the fact that the sequence matching task is a very different problem from the evolution of a network achieving a required functionality, it is preferable to proceed instead to a series of tests that bring the analysis closer to the final objective represented by analog network evolution.

4.2 Network matching experiments

The experiments of network matching reported in this section can be considered a variation of the sequence matching experiments described in the previous section. In the case of sequence matching experiments considered above, the experimenter has complete control over the target sequences and is free to assign their length, number, and composition. Evolution is then asked to produce the required matching between the evolved genome and this collection of fixed target sequences.

In an analog network perspective, this kind of matching is realized when the connections of the evolved analog network with the external devices is based on the use of fixed sequences associated with the terminals of the external devices (Subsection 3.5.1). Even adopting this technique for the external connections, however, most of the interactions present in an evolved network will be realized between the terminals of the devices decoded from the genome. Since those terminals will have associated with them sequences extracted from the evolved genome rather than fixed sequences, it is interesting to consider a set of experiments where a collection of predefined matchings must be realized between sequences of characters that are all evolved, rather than between evolved sequences and fixed sequences. In this case, evolution has the possibility to operate on both sequences in order to realize the required alignment score, instead of being permitted to modify only one sequence, with the other acting as a fixed constraint.

An evolutionary experiment requiring the matching between evolved sequences can be realized by defining a genome with a fixed number of chromosomes and assigning a matrix whose entries prescribe the alignment score that must be realized by evolution between each pair of distinct chromosomes. The required alignment scores could be randomly generated within a reasonable range of alignment values. There is however a way to generate the matrix of required alignment scores that is much more relevant to the evolutionary synthesis of analog networks – which is our actual final goal – than the random generation of required alignment scores.

The idea is to consider an existing analog network – let us call it *model-network* – for instance a textbook example of analog electronic circuit or neural network, and determine how this network could be obtained using the network representation based on sequence alignment defined in the previous chapter. Figure 4.11 illustrates the application of this idea to the case of a simple electronic circuit. All the non-resistive devices existing in the model network are considered as having a sequence of characters associated with each of their terminals (downward arrow in Figure 4.11). A population of individuals having a genome with a number of chromosomes equal to the number of terminals thus determined is created. The goal of evolution is to produce a collection of chromosomes whose pairwise sequence alignment score gives the interactions between the terminals that exist in the model network. In other words, the evolutionary process must find the sequences of characters that, when associated with the terminals of the

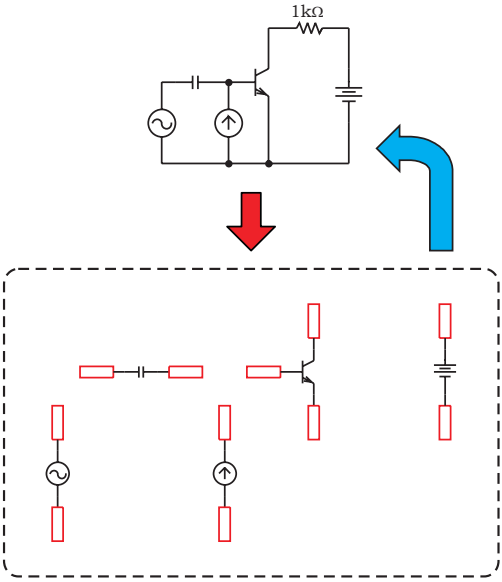


Figure 4.11: *In a network matching experiment a model network is given as a target of the evolutionary process (top). The goal is to generate a collection of sequences that, when associated with the terminals of the devices (bottom) determines between the terminals – via a predefined device interaction map – the terminal-to-terminal interactions of the model network and, thus, reconstructs the network from the devices (upward arrow). Note that in the case of electronic circuits considered here, the resistors determine the interaction between the terminals and, therefore, are not considered in the collection of devices that must be connected.*

devices, reconstruct – or "match" – the model network (upward arrow in Figure 4.11).

The formulation of a sequence matching evolutionary problem in terms of network matching has several interesting properties, besides the already mentioned fact that this approach permits the exploration of the issue of the evolution of alignments between sequences that are all evolved, instead of being partially fixed. The first property consists in the fact that the statistics of sequence alignments that are required

corresponds to that of actual, functional networks, instead of being arbitrarily assigned. This brings the matching experiments closer to the problem of evolving analog networks and makes the corresponding results more telling relatively to the behavior and the choice of the parameters for the evolutionary system applied to that problem. A second important property is the possibility to build a genetic representation of a network and use it to seed the evolutionary process, thus permitting the incremental evolution of pre-existing networks instead of being forced to start evolution from randomly generated networks. To obtain the genetic representation for a given analog network, it is sufficient to use it as model network in a network matching evolution. Once the sequences that match the model network are obtained, they must be simply combined with the convenient device, terminal, and parameter tokens (possibly with the addition of some randomly generated non-coding genome, in order to make the resulting genome more similar to actually evolved genomes) to obtain a genetic representation that will be decoded into the desired network. This genetic representation can then be used to seed the initial population of an evolutionary process, which will thus have as starting point the network that has been encoded in this way.

4.2.1 Network matching for electronic circuits

When the model network is a neural network the implementation of network matching is straightforward. The interactions that must be considered in this case are those between the outputs and the inputs of the neurons, and the strength of the interaction corresponds to the weight associated with the connection. As explained in Section 3.4, a value of sequence alignment between two device terminals is transformed into a value of interaction strength between the terminals through the action of the network-specific interaction map. To define an evolutionary network matching experiment for the model network, we must therefore simply associate with each weight of the model network – including the null weights that implicitly correspond to the absence of connections – the subset of values of sequence alignment that are transformed by the network-specific interaction map into that particular weight, and ask evolution to produce for the pair of chromosomes associated with the two terminals linked by that weight, a sequence alignment score belonging to the subset thus determined.

If the model network is an electronic circuit rather than a neural

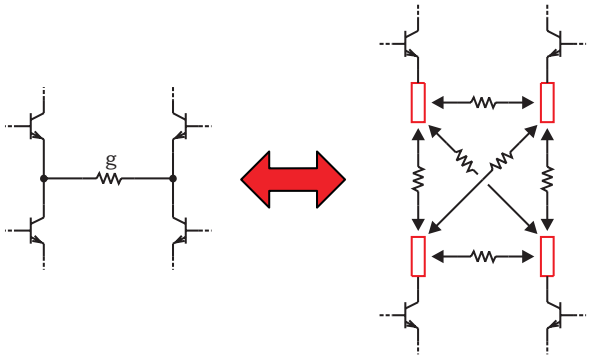


Figure 4.12: *In a network matching experiment using an electronic circuit as model network (left), there are conditions where the resistors that are present in the circuit do not determine univocally the value of each terminal-to-terminal interaction strength. Thus, either an arbitrary assignment is done by the experimenter, or the matching condition is enforced in terms of the equivalent conductance between the terminals that is determined by all the conductances produced by the alignments of the sequences associated with the terminals (right).*

network there is a slight complication in setting up a network matching problem. This is due to the fact that the model network does not determine univocally the interaction between each pair of terminals of the non-resistive devices. The source of this ambiguity can be appreciated considering the example shown in Figure 4.12, where two nodes of a circuit are connected through a resistor having a finite conductance value. We could be tempted to assume the value of conductance of the resistor as corresponding to the strength of the interaction between the terminals of the devices connected by the resistor. However, upon further thought we realize that it is not clear to which pair of terminals this value of interaction strength must be attributed. We cannot associate it with all pairs of terminals connected by the resistor, since the resulting equivalent conductance between the nodes would have a value corresponding to the parallel combination of several instances of that resistor, which does not correspond to the desired value. An obvious solution is to define the network matching problem in terms of the matrix of equivalent conductances between the terminals, rather than directly in terms of the pairwise matching of the sequences. In

the network evolution perspective, this formulation has the further advantage of producing a more realistic evolutionary scenario, since the alignment scores determined by the sequences interact in more complex ways than by requiring directly the matching of the alignment scores. However, it must be realized that this more realistic scenario includes also potentially more conditions that can stall the evolutionary process and, therefore, is probably less suited than the direct matching approach to the generation of the genetic representation of a given network.

4.2.2 Experiment 8: Matching a series of capacitors

The model network used in this first network matching experiment is an electronic circuit composed of a collection of ten capacitors connected in series (Figure 4.13, top). This corresponds to a model network with 10 devices and 20 device terminals (Figure 4.13, bottom). To reconstruct the model network from the collection of unconnected devices, each device terminal must be connected through a wire to exactly one terminal of another device, and have no direct connection with all other terminals. In terms of conductance, this corresponds to an infinite conductance between each terminal and one terminal of another device, and a null conductance between that terminal and all other terminals.

To see what this means in terms of required values of sequence alignment scores between the sequences that are considered as associated with the terminals, we must examine the definition of the network-specific interaction map. We remind from Section 3.4.3 that the network-specific interaction map $i \xrightarrow{N(i)} g$ associates sequence alignment scores i with conductance values g belonging to a discrete set $\{g_0, g_{min}, \ldots, g_{max}, g_\infty\}$, where g_0 is the zero-valued conductance, g_{min} is the minimum non null value of conductance, g_{max} is the maximum finite value of conductance, and g_∞ is the infinite-valued conductance. If i_{min} is the sequence alignment score associated with g_{min}, and i_{max} the alignment score associated with g_{max}, all the values of alignment smaller than i_{min} are associated with g_0, and all those larger than i_{max} are associated with g_∞. This means that each of the 20 sequences associated with a terminal of the devices composing the model network must have a sequence alignment score $i > i_{max}$ relatively to exactly one other sequence, and sequence alignment scores $i < i_{min}$ relatively to all the other sequences (note that the interaction of a terminal with

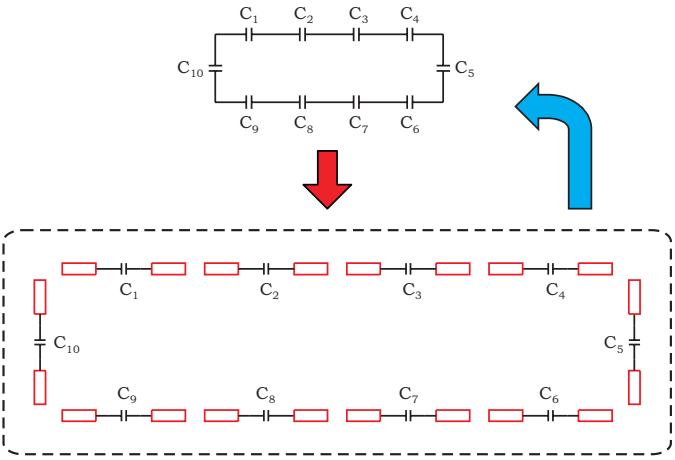


Figure 4.13: *The model network that must be matched in Experiment 8 is composed of 10 capacitors connected in series (top). The reconstruction of the model network from the collection of unconnected capacitors (bottom) requires the evolution of a collection of 20 sequences of characters that are considered as associated with the 20 terminals of the capacitors. Each sequence must interact strongly with just one other sequence, determining the infinite conductance value that corresponds to a direct connection between the terminals. Moreover, it must interact weakly with all other sequences, so as to determine a zero-valued conductance that corresponds to the absence of a direct connection.*

itself is not considered). Thus, the evolutionary problem constituted by the matching of the circuit shown in Figure 4.13 has some analogy with the mixed-value sequence matching problem of Experiment 2, since the two problems require the simultaneous generation of both small and large alignment scores. However, the network matching experiment is much more demanding and realistic, since it enforces both requirements simultaneously on each sequence, instead of enforcing the two kinds of alignment on different sequences.

The actual values of the parameters defining the network specific interaction map are the following.

Network-specific interaction map In the current and in the forthcoming network matching experiments we use a logarithmic quantization of the conductance values (see Appendix B and Subsection 3.4.3), with

- minimum non null conductance value $g_{min} = 10^{-6}\,\mathrm{S}$
  maximum finite conductance value $g_{max} = 1\,\mathrm{S}$
- alignment score associated with g_{min} $i_{min} = 20$
- number of conductance values per decade $n_d = 8$

This gives

- number of conductance values $\neq g_0$ and g_∞ $n_g = 49$
- alignment score associated with g_{max} $i_{max} = 68$
- base of exponential decoder $\alpha = 10^{1/8} \approx 1.33$

Figure 4.14 shows the resulting set of quantized values (apart from g_0 and g_∞) and the associated sequence alignment scores. The other functions and parameters necessary to fully specify the evolutionary experiments are described below.

Fitness function The fitness function of an individual is calculated as follows. Each chromosome is assumed as associated with a preassigned terminal of the non resistive devices of the model network. The association is the same for all the individuals of the population and does not change during the run. The local alignment score c_{ij} of the i-th chromosome relatively to the j-th chromosome ($i \neq j$) is computed. This value is transformed into a terminal-to-terminal conductance value using the network-specific conductance map described above and illustrated in Figure 4.14. Using standard techniques of circuit analysis, the terminal-to-terminal conductance values are transformed into the equivalent conductances $\widetilde{g}_{ij}^{eq.}$ between pairs of terminals. This equivalent conductance must be compared with the value of equivalent conductance $g_{ij}^{eq.}$ existing between the same terminals in the model circuit, which is computed in the same ways as $\widetilde{g}_{ij}^{eq.}$. To perform the comparison in terms of relative rather than absolute difference, both values are converted to a logarithmic index using the relations

$$t_{ij} = \min\left(\max\left(n_d \log_{10} \frac{g_{ij}^{eq.}}{g_{min}}\,,\, -1\right),\, n_g\right) \tag{4.1}$$

(alignment score) quantized resistance value

(68)1.00E+00 (67)1.33E+00 (66)1.78E+00 (65)2.37E+00
(64)3.16E+00 (63)4.22E+00 (62)5.62E+00 (61)7.50E+00
(60)1.00E+01 (59)1.33E+01 (58)1.78E+01 (57)2.37E+01
(56)3.16E+01 (55)4.22E+01 (54)5.62E+01 (53)7.50E+01
(52)1.00E+02 (51)1.33E+02 (50)1.78E+02 (49)2.37E+02
(48)3.16E+02 (47)4.22E+02 (46)5.62E+02 (45)7.50E+02
(44)1.00E+03 (43)1.33E+03 (42)1.78E+03 (41)2.37E+03
(40)3.16E+03 (39)4.22E+03 (38)5.62E+03 (37)7.50E+03
(36)1.00E+04 (35)1.33E+04 (34)1.78E+04 (33)2.37E+04
(32)3.16E+04 (31)4.22E+04 (30)5.62E+04 (29)7.50E+04
(28)1.00E+05 (27)1.33E+05 (26)1.78E+05 (25)2.37E+05
(24)3.16E+05 (23)4.22E+05 (22)5.62E+05 (21)7.50E+05
(20)1.00E+06

Figure 4.14: *The set of quantized resistor values used in the network matching experiments (and also in the circuit evolution experiments described in the next section). The positive integers in round brackets give the sequence alignment score associated with the resistance (and corresponding conductance) value by the network-specific interaction map. Sequence alignment scores i greater than $i_{max} = 68$ are associated with the infinite-valued conductance, which corresponds to the presence of a direct connection between two terminals. Sequence alignment scores i less than $i_{min} = 20$ are associated with the zero-valued conductance, which corresponds to the absence of a direct connection between two terminals.*

$$\widetilde{t}_{ij} = \min\left(\max\left(n_d \log_{10} \frac{\widetilde{g}_{ij}^{eq.}}{g_{min}}, -1\right), n_g\right) \tag{4.2}$$

The $\min$ and $\max$ operators in these formulas represent the fact that the number of quantized conductance values is finite, with g_0 and g_∞ representing the ranges of conductance $[0, g_{min}(\alpha{+}1)/2\alpha]$ and $[g_{min}(\alpha{+}1)/2, \infty)$, respectively. For this reason Equation 4.1 and Equation 4.2 clamp the logarithmic index for $g < (g_{min}/\alpha)$ to $t = -1$, and the logarithmic index for $g > (\alpha\, g_{max})$ to n_g. Applying Equation 4.1 and Equation 4.2 to the equivalent conductances between terminals gives two symmetric ma-

trices $\widetilde{\mathbf{T}} = (\widetilde{t}_{ij})$ and $\mathbf{T} = (t_{ij})$. The fitness of the individual is defined as the negative of the sum of the distance between the elements of $\mathbf{T}$ and those of $\widetilde{\mathbf{T}}$, taking into account the symmetry of the matrix and the irrelevance of the diagonal elements. This corresponds to

$$f = -\sum_{i<j} \left| t_{ij} - \widetilde{t}_{ij} \right|$$

The resulting fitness is negative when some of the logarithmic indexes $\widetilde{t}_{ij}$ of the evolved equivalent conductances do not match the logarithmic indexes t_{ij} of the model network, and is zero – the maximum fitness value – only when all logarithmic indexes are the same.

Genetic alphabet In the current experiment

- size of genetic alphabet $|\mathcal{G}| = 26$

Scoring matrices In the current experiment

- first row of substitution matrix

$$\mathbf{s}_1 = (\ 5,\ 2,\ 1,\ 0,\ -1,\ -2,\ -5,\ \ \ldots\ \ -5,\ -2,\ -1,\ 0,\ 1,\ 2\ )$$

- generic element of indel vector $d_i = -3$

Evolutionary algorithm and parameters The evolutionary algorithm used in the current experiment is a standard generational genetic algorithm that uses tournament selection, elitism, and the genetic operators described in the previous chapter. Since the number of chromosome in the genome must remain constant and equal to the number of terminals of the devices in the model network, the duplication of chromosomes is performed by appending a copy of the chromosome to the existing one, rather than by creating an additional distinct chromosome. The deletion of a chromosome, on the other hand, leads to the elimination of the individual from the population. The actual values of the parameters are listed below. Parameters corresponding to probabilities of genetic operators that were mentioned in the previous chapter and are not listed here are assumed as having null value.

- population size $n_p = 100$
- tournament size $n_r = 5$
- elite size $n_e = 1$

- probability of nucleotide substitution $p_{ns} = 0.001$
  probability of nucleotide insertion $p_{ni} = 0.001$
  probability of nucleotide deletion $p_{nd} = 0.001$
- probability of fragment duplication $p_{f2} = 0.01$
  probability of fragment deletion $p_{fd} = 0.01$
  probability of fragment transposition $p_{ft} = 0.01$
- probability of chromosome duplication $p_{c2} = 0.001$
  probability of chromosome deletion $p_{cd} = 0.001$
- probability of genome duplication $p_{g2} = 0$
- probability of chromosome crossover $p_{c2} = 0.1$
  number of matching characters for crossover $l_m = 10$

Initial population Each individual of the initial population is generated with a genome composed of n_{ic} chromosomes of length l_{ic} randomly generated from the genetic alphabet. Note that in the series of network matching experiments, the number of chromosomes must be equal to the number of terminals of the devices to which a sequence must be associated. In the current experiment

- number of chromosomes in the initial genome $n_{ic} = 20$
- length of chromosomes in initial genome $l_{ic} = 20$

Results Figure 4.15 shows the result of ten repetitions of Experiment 8. The curves of the maximum fitness reveal that in all the runs a set of sequences realizing the desired equivalent conductance between the terminals could be evolved successfully. Comparing these curves with those of Figure 4.2 obtained in Experiment 1, which requires the production of alignment scores of similar magnitude, we see that the average number of generations required to produce the intended result is comparable. The curves of the average genome length show instead a different trend in the two cases. Here, the genome length appears to stabilize, contrary to what happens in Figure 4.2. This is probably due to the fact that an increase in the length of the chromosomes in the current experiment tends to produce unwanted interactions, which are penalized in terms of fitness, whereas in Experiment 1 this penalization was absent.

Figure 4.16 shows an example of evolved genome realizing the required interactions. The parameters of the network-specific interaction

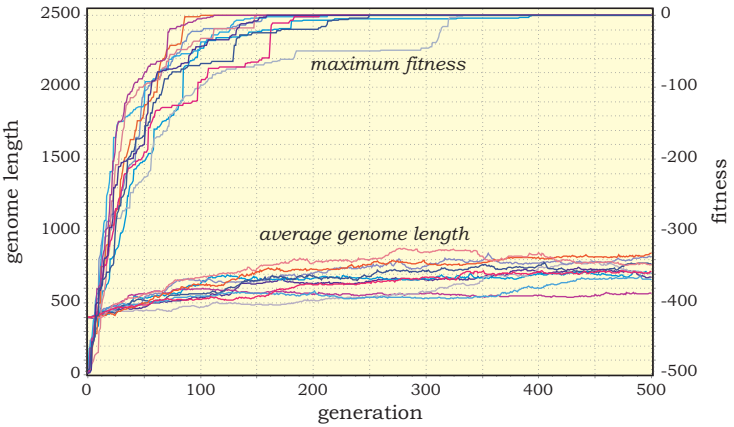


Figure 4.15: *The result of ten repetitions of Experiment 8. The goal of this network matching experiment is the evolution of a collection of sequences that, when associated with the terminals of the capacitors shown in Figure 4.13, reproduce the series connection of the capacitors in the model circuit. A negative value of fitness corresponds to an imperfect reproduction of the model circuit, and the null fitness value corresponds to an exact reproduction. The curves of maximum fitness reveal that all runs evolved successfully a collection of sequences realizing the required interactions.*

map used in this experiment specify that the sequence alignment score corresponding to g_{max} is $i_{max} = 68$ and the value corresponding to g_{min} is $i_{min} = 20$. The realization of a connection with infinite conductance between two terminals requires therefore an alignment score $i > 68$, and the absence of a direct connection between two terminals is realized by values of sequence alignment score $i < 20$. Figure 4.16 represents only the pairing of sequences that realize an alignment $i \geq 20$ and thus correspond to an actual connection existing between the corresponding terminals. The figure reveals that the only connections realized correspond to values of alignment with $i > 68$, and result therefore in ten connections having infinite conductance – ten wires. This is just the kind of interaction required to connect each capacitor to its neighbor in the model circuit of Figure 4.13.

Finally, note that this kind of experiment permits the generation of collections of sequences that can be used as fixed preassigned se-

Figure 4.16: *An example of genome evolved in the network matching experiment using as model network a series of 10 capacitors (Experiment 8, Figure 4.13). The genome is composed of 20 chromosomes, each constituting one of the evolved sequences for the network matching. To reproduce the model network, each evolved sequence must realize an alignment score greater than 68 with exactly one other sequence, and values smaller than 20 relatively to all other sequences. The figure shows the pairs of sequences realizing an alignment score greater than or equal to 20, and the corresponding alignment score. Note that all the scores are above 68, corresponding to a direct connection between terminals. The alignments of the sequences that produce the reported alignment score are not represented in the figure.*

quences to be associated with the terminals of the external devices as described in Section 3.5 on page 103, since the sequences obtained do not interact pairwise and yet permit the production of a connection with maximum interaction strength.

4.2.3 Experiment 9: Matching a resistor-capacitor series

In the previous experiment, the terminals of each capacitors were connected through a wire to the terminals of the neighboring capacitors. This has the consequence that the alignment score that must be realized by the evolved sequences to produce the connections is rather large (greater than 68 in the case of Experiment 8). Furthermore, the alignment score that must be realized does not prescribe an exact value of matching but consists in the entire range above i_{max}. It is therefore interesting to consider also the case where a prescribed value of interaction is required. This can be obtained modifying the model circuit of Experiment 8 shown in Figure 4.13 by inserting a resistor of finite conductance in the place of the resistors with infinite conductance – i.e, in

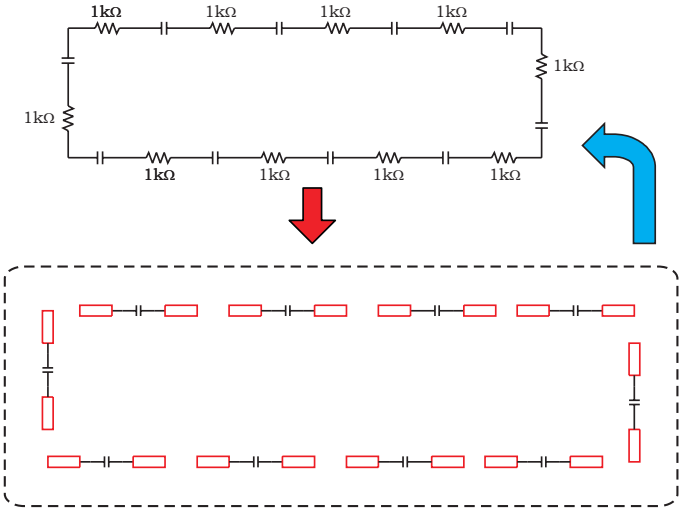


Figure 4.17: *The model network that must be matched in Experiment 9 is a series of capacitors whose terminals are connected through non null resistors (top). The reconstruction of the model network from the collection of unconnected capacitors (bottom) requires the generation of sequences each of which has a small alignment score with all but one of the other sequences, relatively to which it must realize exactly the alignment score corresponding to the connecting resistors.*

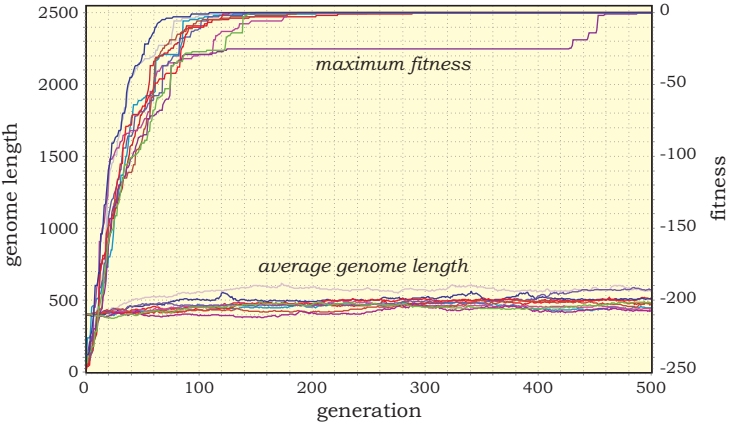


Figure 4.18: *The result of ten repetitions of Experiment 9, aimed at evolving the sequences required to reconstruct the model network shown in Figure 4.17. The maximum fitness curves are very similar to those shown in Figure 4.15, whereas the average genome length of the population appears appreciably smaller in the present case. This is a consequence of the smaller alignment scores required by the current experiment.*

place of the wires – connecting the capacitors. The resulting circuit is shown in Figure 4.17.

A series of evolutionary experiments aimed at matching this circuit was performed using the parameters of Experiment 4.13. Figure 4.18 shows the outcome of ten evolutionary runs. There seems to be no significant difference in the fitness performance with respect to the case of the previous experiment illustrated in Figure 4.15. The smaller sequence alignment scores that must be generated in this case appears instead to produce an appreciable difference in the resulting average genome length of the populations. This is confirmed by the inspection of the genomes of individuals attaining the optimal fitness. Figure 4.19 shows an example of this kind of genome. Note how this genome is shorter than that shown in Figure 4.16, and realizes exactly the alignment score corresponding to the value of the resistors present in the model network of this experiment (Figure 4.17).

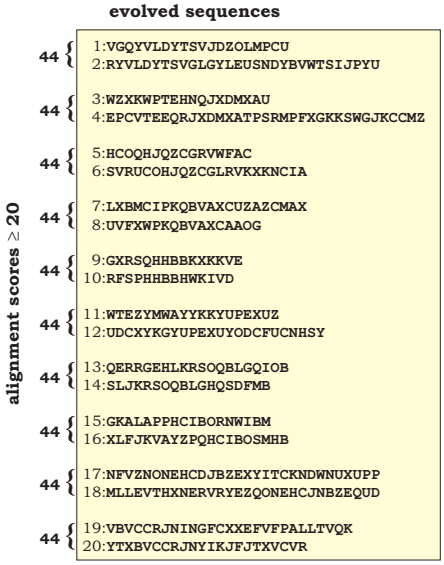


Figure 4.19: *An example of genome evolved in the context of Experiment 9 and realizing the perfect matching of the model network shown in Figure 4.17. Only the pairings realizing an alignment score greater than or equal to 20 are represented. Note that with the series of quantized resistance values shown in Figure 4.14 the resistance of 1kΩ present in the model network correspond to an alignment score of 44, which is exactly realized in all the cases.*

4.2.4 Experiment 10: Matching a typical analog electronic circuit

For the last network matching experiment the textbook example of electronic circuit shown in Figure 4.20 was chosen. The reason of this choice is to test the possibility of evolving a collection of sequences realizing the values of alignment with the kind of distribution that can be expected to be found in a typical circuit. For example, electronic circuits have typically a few power supply nodes that are connected to many device terminals, and the rest of the nodes that are only sparsely connected.

Apart from the model network, all the parameters used in this experiment are those of Experiment 4.13. Figure 4.21 shows the result of five evolutionary runs, with linear and logarithmic scales for the gener-

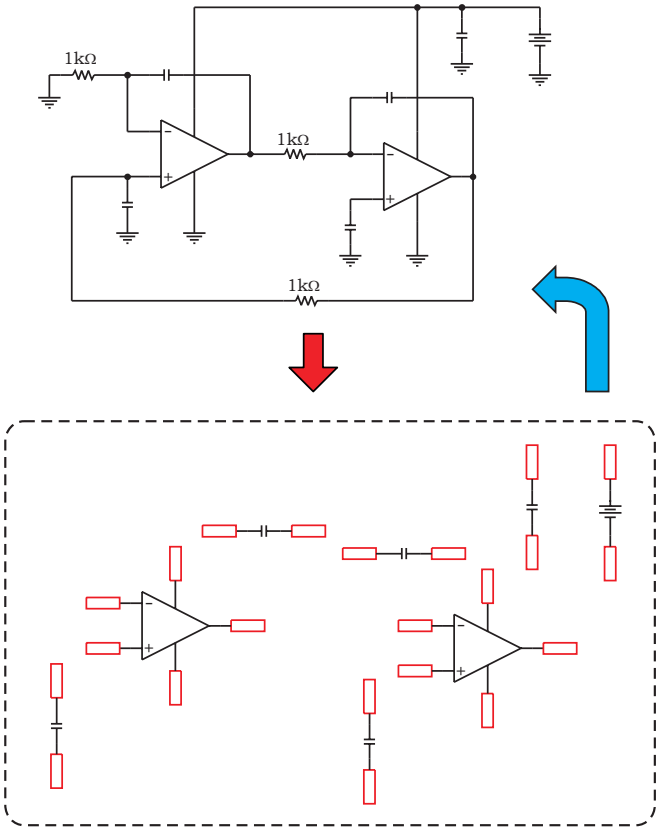


Figure 4.20: *The figure shows (top) the network that must be matched in Experiment 10. The goal of the experiment is to evolve one sequence of characters for each terminal belonging to a non-resistive device of the network (bottom), such that the interaction strength determined by the sequences through the device interaction map produces between the terminal the same equivalent resistance existing in the given network. Contrary to the artificial cases considered before, the interaction of the sequences thus generated has the statistical properties of a typical network.*

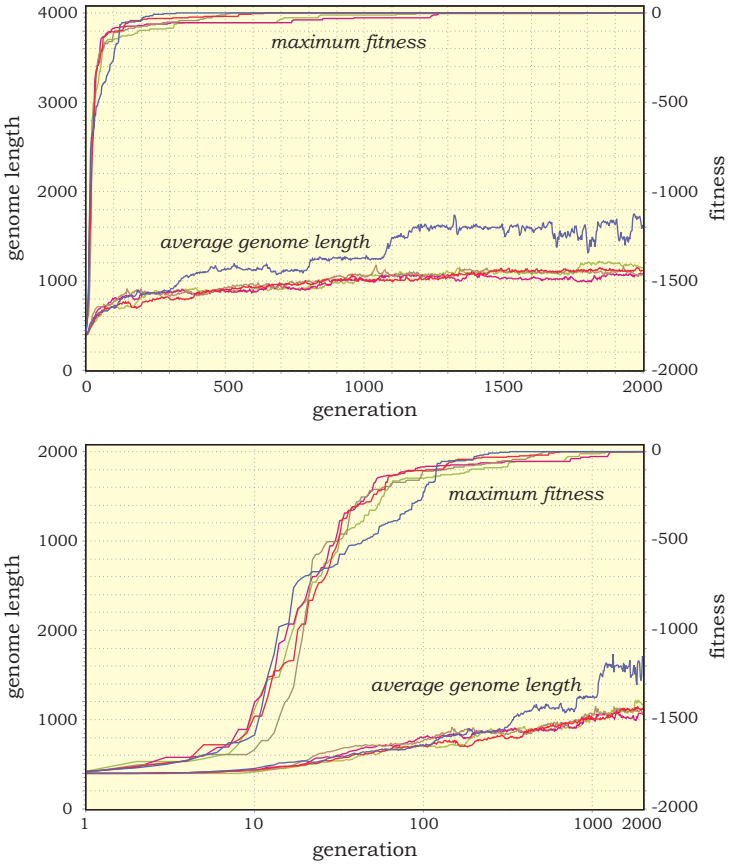


Figure 4.21: *The result of five repetitions of Experiment 10, aimed at the evolution of a genome whose chromosomes permit the reconstruction of the electronic circuit shown in Figure 4.20. The bottom panel shows the same results of the top panel, redrawn with a logarithmic generation scale, to permit a better appreciation of the initial phase of the evolutionary process. The greater complexity of the circuit that must be reconstructed relatively to the preceding examples results in a larger number of generations required to evolve the required sequences which, however, are successfully evolved in all runs.*

ation axis. The maximum fitness curves reveal that the required character sequences could be generated also in this more realistic network matching context and within a reasonable number of generations.

4.2.5 Discussion

The success of this small series of network matching experiments, and that of the preceding series of sequence matching experiments is encouraging relatively to the choice of the technique of local sequence alignment for the implementation of the sequence interaction map. From this success, however, does not ensue that the evolutionary system based on sequence matching must be able to evolve actual analog networks performing a given function. These results indicate nevertheless that at least some of the basic ingredients required to evolve analog network are within the system's reach, in particular, the possibility to match fixed sequences and to generate fragments of genome realizing high-valued sequence alignments while keeping interference under control. Moreover, these series of experiments have confirmed that the choice of parameters values based on the considerations of the previous chapter is reasonable. The information gained will be used in the next series of experiments, which will at last tackle the evolutionary task that actually interests us: that of evolution of analog networks.

4.3 Network evolution experiments

In the series of experiments presented in this section the approach proposed in this thesis is applied to the evolution of analog networks. First, a series of three experiments of evolution of analog electronic circuits in presented, followed by an example of evolution of a neural network. The performance of the electronic circuits corresponding to the individuals of the evolving populations in the experiments of circuit evolution was evaluated in simulation, using SPICE as circuit simulator (see Appendix C for additional details on SPICE and about its use in an evolutionary context).

Any electronic design textbook could have provided manifold examples of circuit functionality to be assumed as target for our experiments of electronic circuit evolution. However, in order to make our results comparable to those of alternative evolutionary approaches to the synthesis of analog networks, we based our experiments on three

problems taken from (Koza et al., 1999), where genetic programming is applied to this kind of evolutionary problem. Our choice is motivated by the fact that the results reported in that work (and in the more recent (Koza et al., 2003)) are arguably some of the most remarkable examples of evolutionary synthesis of electronic circuit existing in the literature. Moreover, the evolutionary experiments presented in those two books have the additional advantage of assessing the performance of the evolved circuits using the same circuit simulator used here.

From the set of problems considered in the above mentioned books, we restricted our choice to problems that can be formulated as *single-objective* problems, as opposed to multi-objective problems (Deb, 2001).[4] The reason for this limitation is on the one hand the desire to avoid in the interpretation of the first results obtained from our evolutionary system the complication that would have ensued from the use of a multi-objective evolutionary algorithm, and, on the other hand, from the questionable rationale behind the aggregation from the part of the experimenter of several disparate evolutionary targets into a single scalar-valued fitness function. For this reason the three examples of circuit evolution presented below are taken from (Koza et al., 1999), where the majority of problems can be formulated in single-objective terms, rather than from the more recent (Koza et al., 2003) where the focus has shifted on multi-objective problems. Of the single-objective problems considered in (Koza et al., 1999), we chose those that appeared to us as the most challenging for an evolutionary process, as witnessed by the less than perfect conformity of the results presented, with the assumed evolutionary target.

4.3.1 Experiment 11: Evolution of a voltage reference

The first experiment of network evolution is aimed at the synthesis of a voltage reference electronic circuit. Figure 4.22 shows the devices of the predefined external circuit, to which the evolved circuit must connect in order to produce the required functionality. The external circuit is composed of a variable voltage source V_c connected in series to a resistor R_c, and by a load resistor R_L.

The goal of the evolutionary experiment is the synthesis of a circuit

[4] Despite this choice, we remain convinced that actual, real-world circuits synthesis must be eventually formulated as multi-objective evolutionary problems. The final part of this section contains some further comments on the issue of multi-objective artificial evolution in general and in the context of network evolution in particular.

producing a fixed output voltage $V_o^* = 2\text{V}$ across the load resistor when the source voltage V_c varies in the range $4\text{V} \leq V_c \leq 6\text{V}$ and the circuit temperature T varies in the range $0°\text{C} \leq T \leq 100°\text{C}$. From these requirements we can derive the following fitness function for our experiment.

Fitness function Using the SPICE simulator the behavior of the circuit can be evaluated by assigning a fixed value of input voltage and circuit temperature, and using the simulator to compute the corresponding output voltage. The input voltage range was discretized by subdividing it into intervals of width $\Delta V_c = 0.1\text{V}$, resulting in 21 discrete values $\{V_{c_i}\} = \{4\text{V}, 4.1\text{V}, 4.2\text{V}, \ldots, 6\text{V}\}$. The temperature range was discretized into the five equispaced values of the set $\{T_j\} = \{0°\text{C}, 25°\text{C}, 50°\text{C}, 75°\text{C}, 100°\text{C}\}$. The output voltage $V_{o_{ij}}$ is computed in correspondence of each input voltage value V_{c_i} and circuit temperature T_j pair, and the difference $(V_{o_{ij}} - V_o^*)$ is calculated. A tolerance interval of width $2\Delta V_0 = 0.02\text{V}$ centered on V_o^* is defined, and the output voltage val-

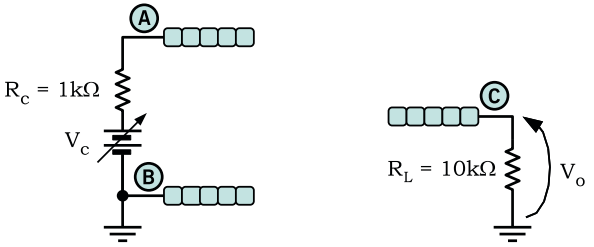


Figure 4.22: *The devices of the external network in the experiment of evolution of a voltage reference. The connection of the evolved network with the devices of the external network is based on the use of I/O ports. The external network is an electronic circuit composed of a variable voltage source V_c with a series resistance R_c, and by a load resistor R_L across which the output voltage V_o of the evolved network is measured. Note that, contrary to the case of the evolved network, the resistors of the external network are preassigned devices that are not obtained as a consequence of the interaction between sequences of nucleotides extracted from the genome. Note also that there is no I/O port associated with the wire connecting V_c with R_c. Consequently, the terminals connected by this wire cannot be connected directly to the evolved network by evolution.*

ues falling into that interval are not penalized in terms of fitness. This choice was made in order to have the resulting fitness function reflect a typical approach used in evaluating the results in electronic circuit design, where the performance of a circuits relatively to a given specification is judged acceptable without further qualification when it stays within a given range. This leads to the following definition for the error in correspondence of each input voltage and circuit temperature pair

$$\varepsilon_{ij} = \begin{cases} (V_{o_{ij}} - V_o^*)^2 & \text{if } |V_{o_{ij}} - V_o^*| \geq \Delta V_0 \\ 0 & \text{if } |V_{o_{ij}} - V_o^*| < \Delta V_0 \end{cases}$$

and to the following expression for the fitness f

$$f = -\sum_{i,j} \varepsilon_{ij}$$

This means that the values of fitness are negative when there is a discrepancy exceeding the tolerance ΔV_0 between the computed output voltage values $V_{o_{ij}}$ and the desired fixed output voltage value V_o^*. A null value of fitness signals the attainment of an output voltage that falls within the tolerance interval in correspondence of all the discrete input voltage and temperature values used in the simulations of the circuit.

External connections As illustrated in Figure 4.22, the technique adopted for the establishment of the connections between the external and the evolved circuits is the one based on the use of I/O ports. With this technique, some terminals of the external devices are marked for (potential) connection by associating with them a special kind of device that is represented in the genome and leads to the association with the marked terminals of sequences of characters extracted from the genome (see Subsection 3.5.4 and in particular Figure 3.28 on page 111 for the details).

Network-specific interaction map All the experiments of analog circuit evolution reported below use a the network-specific interaction map based on the same logarithmic quantization of the conductance values employed in the network matching experiments described in the previous section. We have in particular

minimum non null conductance value	$g_{min} = 10^{-6}\,\mathrm{S}$
maximum finite conductance value	$g_{max} = 1\,\mathrm{S}$

- alignment score associated with g_{min} $i_{min} = 20$
- number of conductance values per decade $n_d = 8$

From which follows

- number of conductance values $\neq g_0$ and g_∞ $n_g = 49$
- alignment score associated with g_{max} $i_{max} = 68$
- base of exponential decoder $\alpha = 10^{1/8} \approx 1.33$

The resulting set of quantized values (apart from g_0 and g_∞) and associated sequence alignment scores is shown in Figure 4.14 on page 144.

Genetic alphabet

- size of genetic alphabet $|\mathcal{G}| = 26$

Scoring matrices

- first row of substitution matrix
  $\mathbf{s}_1 = (\ 5,\ 2,\ 1,\ 0,\ -1,\ -2,\ -5,\ \ \ldots\ \ -5,\ -2,\ -1,\ 0,\ 1,\ 2\ )$
- generic element of indel vector $d_i = -3$

Evolutionary algorithm and parameters The evolutionary algorithm used in the current experiment is a standard generational genetic algorithm that uses tournament selection, elitism, and the genetic operators described in the previous chapter. The values of the parameters are listed below. Parameters corresponding to probabilities of genetic operators that were mentioned in the previous chapter and are not listed here are assumed as having zero value.

- population size $n_p = 100$
- tournament size $n_r = 5$
- elite size $n_e = 1$
- probability of nucleotide substitution $p_{ns} = 0.001$
  probability of nucleotide insertion $p_{ni} = 0.001$
  probability of nucleotide deletion $p_{nd} = 0.001$
- probability of fragment duplication $p_{f2} = 0.01$
  probability of fragment deletion $p_{fd} = 0.01$
  probability of fragment transposition $p_{ft} = 0.01$

✓ probability of chromosome duplication	$p_{c2} = 0.001$
probability of chromosome deletion	$p_{cd} = 0.001$
✓ probability of genome duplication	$p_{g2} = 0.001$
✓ probability of individual genome trimming	$p_{gt} = 0.01$
probability of population genome trimming	$p_{pt} = 0.01$
✓ probability of device insertion	$p_{di} = 0.01$
✓ probability of chromosome crossover	$p_{c2} = 0.1$
number of matching characters for crossover	$l_m = 10$

Device set The device set for the current experiment contains only one NPN bipolar junction transistor (the details of the SPICE transistor model used can be obtained from the SPICE deck relative to this experiment shown in Table D.1 on page 270 of Appendix D).

Initial population Each individual of the initial population is obtained by generating a genome composed of n_{ic} chromosomes. The genome contains the representation of n_{id} devices randomly chosen in the device set (in this experiment the device set contains only one element, but in the subsequent experiments it will contain more than one element), and the representation of n_{ip} copies of each kind of I/O port associated with the external circuit. The representation for devices and I/O ports is obtained appending to the token for the device or I/O port the required number of randomly generated sequences of nucleotides of length l_{it}, each delimited by a terminal token at its end. Fragments of genome representing distinct devices are separated in the genome by randomly generated spacer sequences of length l_{is}. Spacer sequences are also prepended at the start and appended at the end of each chromosome. In the current experiment

✓ number of chromosomes in the initial genome	$n_{ic} = 1$
number of devices in the initial genome	$n_{id} = 1$
number of copies of each I/O port in the initial genome	$n_{ip} = 1$
✓ length of sequences for terminals in the initial genome	$l_{it} = 20$
length of spacer sequences in the initial genome	$l_{is} = 20$

Results Figure 4.23 shows the result of 25 repetitions of Experiment 11. The interpretation of the figure is hindered by the overlap of some of

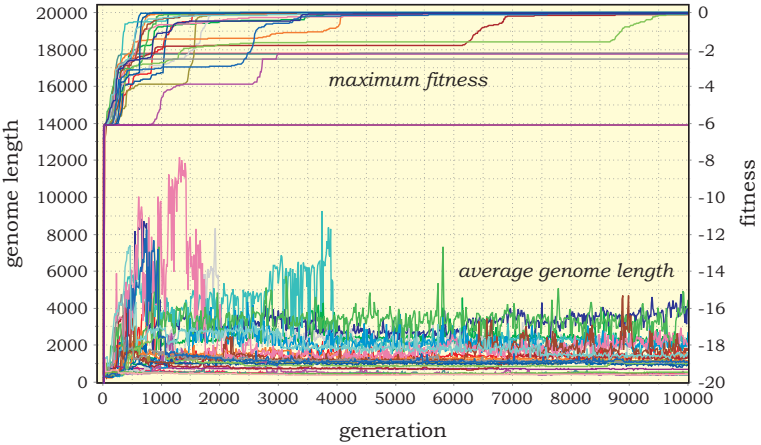


Figure 4.23: *The result of 25 repetitions of Experiment 11 aimed at the evolution of a voltage reference electronic circuit. A better performance corresponds to higher values of fitness, with a maximum value of zero corresponding to the exact realization of the required circuit function. The inspection of the tables of values corresponding to curves of the maximum fitness (top) reveal that after 10000 generations 18 of the 25 runs have produced at least one circuit realizing a fitness very close to zero. The curves of the average genome length (top) show that after an initial transient where several runs experienced a sustained genome growth, all the experiments settled on moderate or null rates of genome length growth. Some of the curves shown here are reproduced in a less cluttered context in Figure 4.24.*

the curves of maximum fitness for different experiments. An analysis of the maximum fitness data reveals that 18 of the 25 experiments attained a fitness very close to zero, corresponding to the evolution of a circuit that achieves a good regulation of the output voltage. We will examine later in more detail the kind of regulation behavior obtained in the successful runs. The analysis of data collected during the runs revealed that the runs that did not evolve a circuit with acceptable performance within 10000 generations corresponded to evolved circuits that had just one or two devices that were actually connected, and that runs that appeared as stalled in terms of maximum fitness could

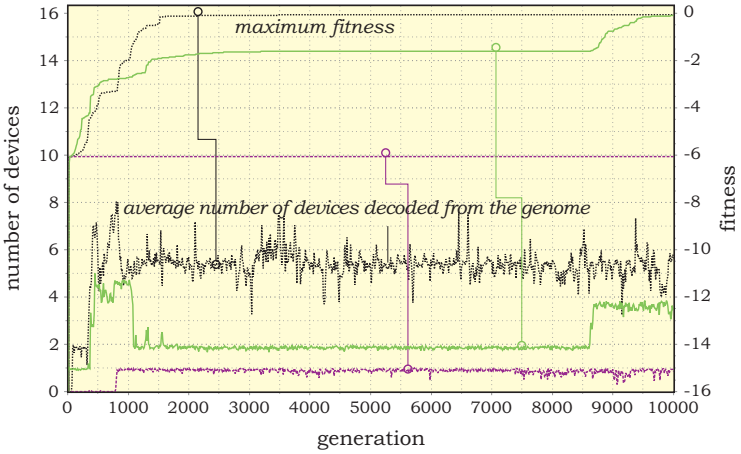


Figure 4.24: *The curves of maximum fitness and average number of devices in the genome for a selection of three of the 25 runs shown in Figure 4.23. This selection of runs is presented here to permit the appreciation of some typical phenomena observed during the runs, such as the temporary high number of devices in the initial generations of evolving populations, the correlation between an abrupt increase in the number of devices and a substantial increase in maximum fitness, and the insufficient number of devices in the individuals of the populations of the runs that did not evolve high fitness solutions within 10000 generations.*

overcome the impasse following an increase in the number of encoded devices. Figure 4.24 shows the relationship between the number of devices encoded in the genome and the maximum value of fitness during three of the 25 repetitions of the experiment, selected to illustrate this and other typical behaviors observed during the runs.

Figure 4.25 shows an example of best evolved voltage reference circuit from one of the successful runs. The performance of this circuit is illustrated in Figure 4.26. Figure 4.27 shows the best circuit produced after 80 generations using a population size of 640,000 in the experiment of evolution of a voltage reference with genetic programming (GP)

described in Chapter 50 of (Koza et al., 1999), and Figure 4.28 shows the schematics of the corresponding evolved circuit. Note that the results of the experiment performed in this thesis and of the experiments described in (Koza et al., 1999) cannot be directly compared, since the two sets of experiments use in some cases different sets of devices (for example, including PNP transistors in the GP case), different SPICE models for the devices, or different external circuits. The purpose of presenting side to side the results of the two set of experiments is to show that the evolutionary approach proposed in this thesis achieves on this kind of problem a performance comparable with the best results of evolution of analog electronic circuits existing in the literature. Moreover, the observation of the circuits evolved with the GP approach and of those evolved with the sequence alignment approach introduced in this thesis permits to appreciate the influence of the genetic encoding on the structure of the evolved circuit. In particular, the abundance of resistors connecting the terminals of the transistors in circuits evolved with the sequence alignment approach, for example that of Figure 4.25, reveals their crucial evolutionary role in the approach proposed in this thesis.

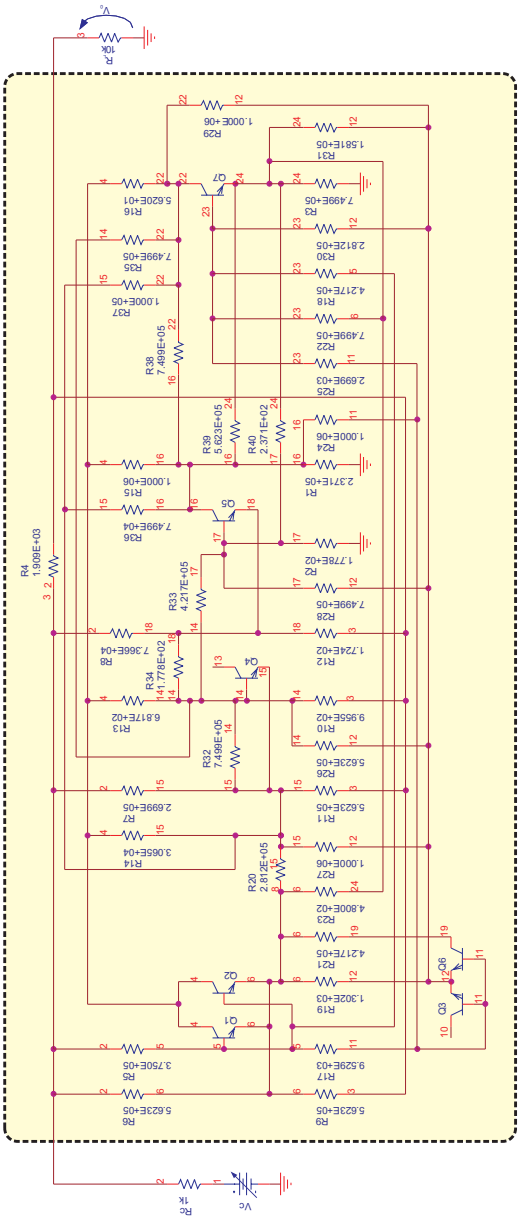


Figure 4.25: *An example of voltage reference circuit evolved in the context of Experiment 11. The circuit contains 7 devices that have been decoded form the genome (transistors Q1-Q7, inside the dotted box). Using the device interaction map decribed in the previous chapter, the terminals of these devices have been connected among themselves and with the terminals of the devices of the preassigned external circuit (drawn outside of the dotted box) that were marked for connection. Note that not all pairs of terminals is directly connected by a wire or by a resistor, showing that the sequence alignment approach does not lead necessarily to the connection of all pairs of device terminals. The SPICE input file corresponding to this circuit is shown in Table D.1 of Appendix D.*

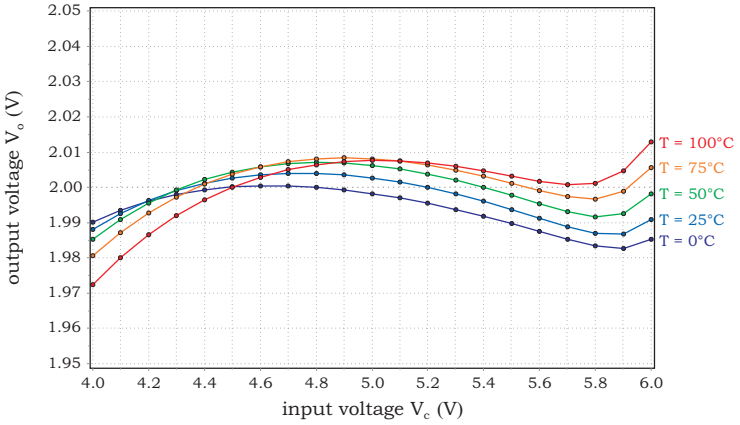


Figure 4.26: *The output voltage V_o of the evolved voltage reference circuit shown in Figure 4.25, plotted as a function of the input voltage V_c. The markers correspond to values of input voltage used to evaluate the circuit fitness. The different curves correspond to the five temperatures at which the circuit is tested. The curves show that the circuit achieves a remarkable performance, since V_o approximates to a few percent the desired fixed output voltage value of $2V$.*

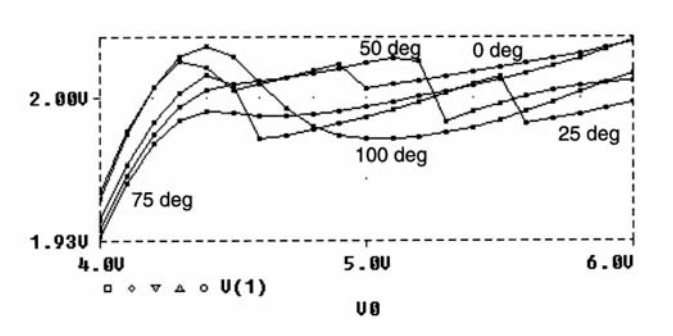


Figure 4.27: *Figure 50.9 from (Koza et al., 1999) shows the output voltage of the evolved voltage reference circuit shown in Figure 4.28. The circuit is evaluated at the same values of input voltage and circuit temperature used to plot the curves of Figure 4.26. The markers are drawn in correspondence of the input voltage values used to evaluate the circuit fitness. The different curves correspond to the five temperatures at which the circuit is tested.*

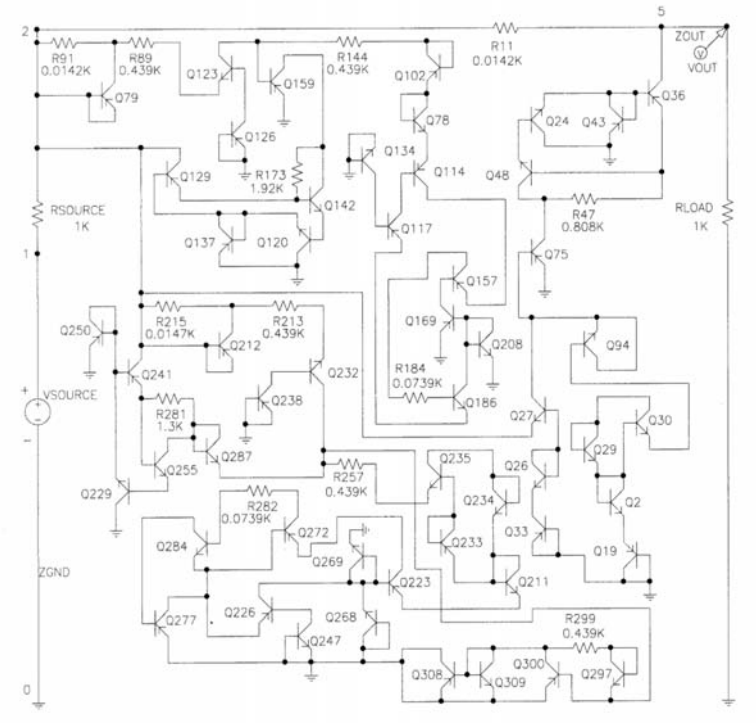


Figure 4.28: *Figure 50.8 from (Koza et al., 1999) shows a voltage reference circuit evolved with the genetic programming technique (GP). Relatively to the circuit evolved with the sequence matching approach which is shown in Figure 4.25, this circuit uses more active devices and less non-null resistors. This is a consequence of the different genetic representation used to evolve the two circuits. In the case of the circuit evolved with GP shown here, the encoding defines series of circuit transformation instructions that are used to build the circuit starting from an elementary initial circuit, and does not assign a special role to the resistors. In the case of the sequence matching encoding proposed in this thesis and used to synthesize the circuit shown in Figure 4.25, the focus is on the genetic representation of the devices and of their interaction via resistors connecting their terminals.*

4.3.2 Experiment 12: Evolution of a temperature-sensing circuit

The second experiment of network evolution is aimed at the synthesis of a temperature sensing electronic circuit. Figure 4.29 shows the devices of the predefined external circuit: it is composed of two fixed voltage sources,V_c and V_s connected in series to the resistors R_c and R_s, and by a load resistor R_L. The goal of the evolutionary experiment is the synthesis of a circuit producing across the load resistor an output voltage V_o that is proportional to the circuit temperature T in the range $0°\mathrm{C} \leq T \leq 100°\mathrm{C}$, with a proportionality coefficient of $\gamma = 0.1\mathrm{V}/°\mathrm{C}$. In other words, when the circuit temperature varies from 0°C to 100°C, the output voltage must vary linearly from 0V to 10V.

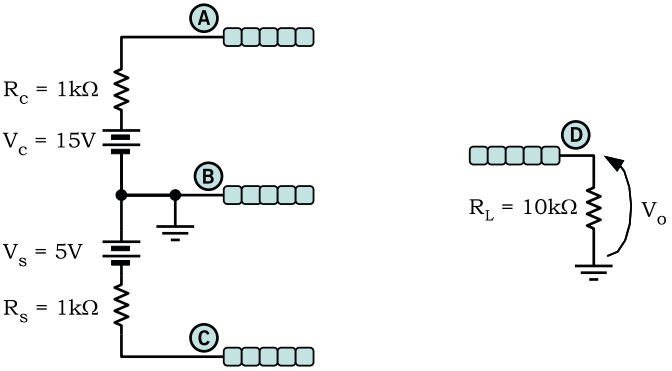


Figure 4.29: *The devices of the external network in the experiment of evolution of a temperature sensing electronic circuit. The connection of the evolved network with the devices of the external circuit is based on the use of I/O ports associated with the external devices. The external network is composed of two fixed voltage sources V_c and V_s, two series resistance R_c and R_s, and a load resistor R_L across which the output voltage V_o of the evolved network is measured.*

Fitness function Using SPICE the output voltage $V_{o_i}$of the evolved circuits is evaluated in correspondence of the discrete set $\{T_i\} = \{T_0, T_1, \ldots, T_{20}\} = \{0°\mathrm{C}, 5°\mathrm{C}, 10°\mathrm{C}, 15°\mathrm{C}, \ldots, 100°\mathrm{C}\}$ of 21 equispaced values of temperature in the range of interest. The difference $(V_{o_i} - V^*_{o_i})$ between the output voltage obtained at the temperature T_i and the desired output

voltage $V^*_{o_i} = \gamma\, T_i$ is calculated. The fitness function f is defined as follows

$$f = -\sum_i (V_{o_i} - V^*_{o_i})^2$$

This means that the values of fitness are negative when there is a discrepancy between any of the computed output voltage values V_{o_i} and the desired output voltage value $V^*_{o_i}$. A null value of fitness signals the attainment of an output voltage that complies exactly with the requirements.

Device set The device set for the current experiment contains an NPN and a PNP bipolar junction transistor (the details of the SPICE transistor models used can be obtained from the SPICE deck relative to this experiment shown in Table D.2 on page 271 of Appendix D).

Initial population Using the same symbols used for the parameters of the previous experiment

☑	number of chromosomes in the initial genome	$n_{ic} = 1$
	number of devices in the initial genome	$n_{id} = 5$
	number of copies of each I/O port in the initial genome	$n_{ip} = 2$
☑	length of sequences for terminals in the initial genome	$l_{it} = 20$
	length of spacer sequences in the initial genome	$l_{is} = 20$

The only difference in the evolutionary parameters relatively to the previous experiments is the number of devices and I/O ports used in the generation of the initial population. This choice was dictated by presence of two elements in the device set in place of the unique element of the previous experiment. Increasing the number of devices inserted in the genome of the individuals of the initial population guarantees a higher probability of presence of both kinds of devices in the initial genome. Interestingly enough, as Figure 4.30 reveals, the increase in the number of devices randomly generated and inserted in the genomes of the initial population resulted in the initial elimination of most of these devices, and their later substitution with devices generated by the evolutionary process. This observation indicates that – as could be expected – randomly generated devices have a small probability of cooperating to produce the required functionality, and that evolution

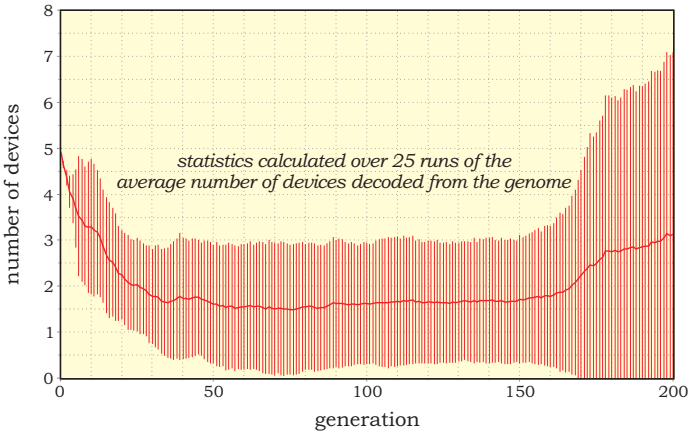


Figure 4.30: *A detail relative to the first 200 generations of the statistics of the population average number of devices decoded from the genome in the runs reported in Figure 4.31. The figure shows the mean value and the standard deviation bars calculated over the 25 runs shown in Figure 4.31. Note that although the number of devices assigned to each individual of the initial population is 5, this number decreases initially, to increase again at the end of the range of generations considered. This suggests an initial elimination of most of the randomly generated initial devices and their substitution with devices newly generated by the genetic operators during the evolutionary process.*

proceeds instead complexifying incrementally elementary circuits composed of one or two devices.

All the remaining evolutionary parameters are those specified in the previous experiment.

Results Figure 4.31 shows the result of 25 repetitions of Experiment 12. The maximum fitness data reveals that 15 of the 25 experiments attained a fitness very close to zero within 8000 generations,[5] corresponding to the evolution of at least one circuit that approximates well the required linearity of the relationship between the output volt-

[5]This anomalous number of generations is due to the premature interruption of the planned 10000 generations due to extraneous reasons.

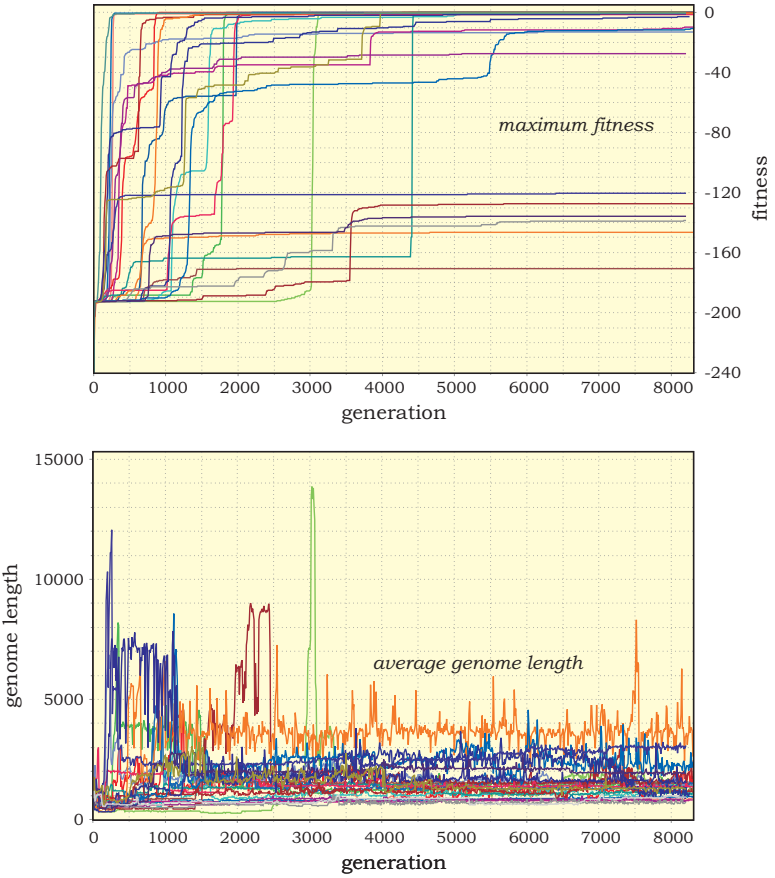


Figure 4.31: *The result of 25 repetitions of Experiment 12 aimed at the evolution of a temperature sensing electronic circuit. A better performance corresponds to higher values of fitness, with a maximum value of zero corresponding to the exact realization of the required circuit function. The inspection of the curves of the maximum fitness (top) reveal that after about 8000 generations 15 of the 25 runs have produced at least one circuit realizing a fitness very close to zero, with several of the remaining runs showing the signs of an ongoing fitness progress towards zero. The curves of the average genome length (bottom) show a behavior similar to that observed in Figure 4.23 for the evolution of the voltage reference circuit.*

age and the circuit temperature. To these results apply the comments made in the context of the previous experiment for the results shown in Figure 4.23.

Figure 4.32 shows an example of evolved temperature sensing circuit from one of the successful runs. The performance of this circuit is illustrated in Figure 4.33. Figure 4.34 shows the best circuit produced after 25 generations using a population size of 640,000 in the experiment of evolution of a temperature sensing circuit with genetic programming (GP) described in Chapter 49 of (Koza et al., 1999), and Figure 4.35 shows the schematics of the corresponding evolved circuit. The comparison of the circuit of Figure 4.32 with that of Figure 4.35 confirms the difference in the composition of the circuits evolved with the two approaches, with a greater abundance of resistors in the circuit evolved with the approach proposed in this thesis.

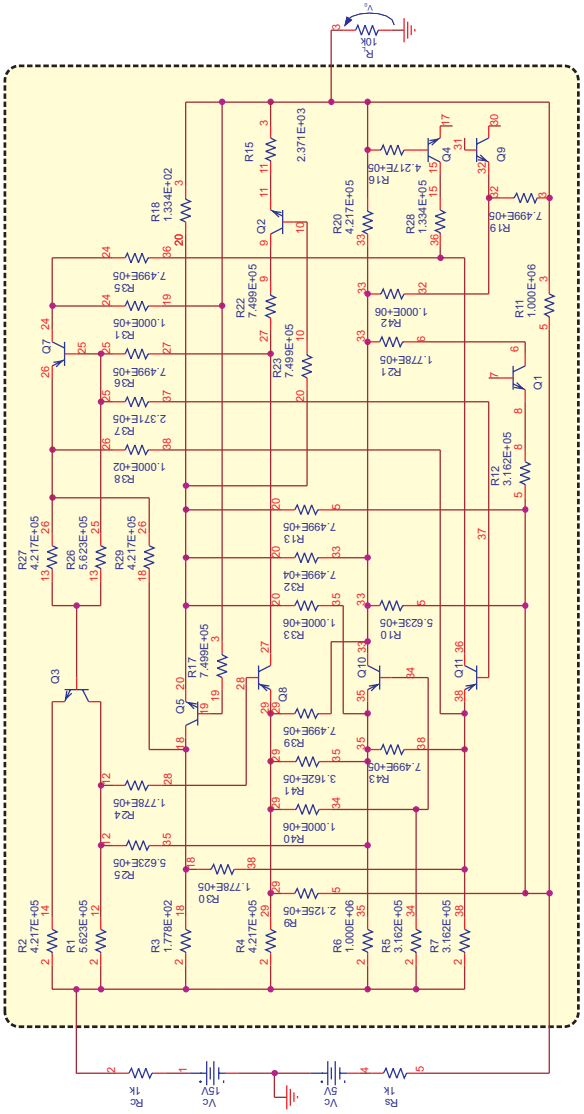


Figure 4.32: *An example of temperature sensing circuit evolved in the context of Experiment 12. The devices of the preassigned external circuit are drawn outside of the dotted box. The composition of this circuit is similar to that of the voltage reference circuit shown in Figure 4.25, showing a few transistors decoded from the genome, connected by a larger number of resistors. The SPICE input file corresponding to this circuit is shown in Table D.2 of Appendix D.*

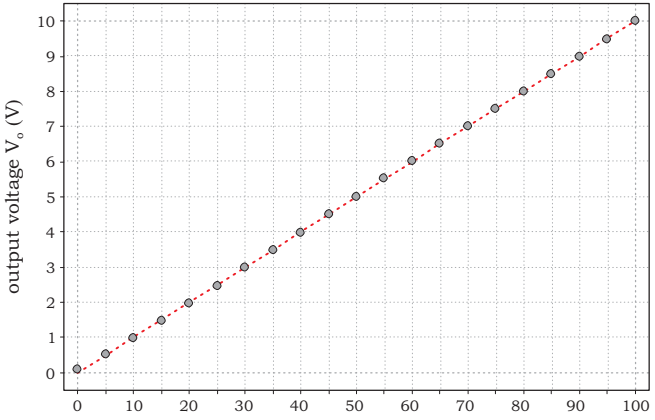


Figure 4.33: *The graph of the output voltage V_o of the evolved temperature sensing circuit shown in Figure 4.32 plotted as a function of the circuit temperature T. The markers correspond to the discrete values of circuit temperature used to evaluate the circuit fitness. The circuit displays an excellent performance, as witnessed by the small distance of the markers from the dotted line that represents the ideal linear relationship between T and V_o.*

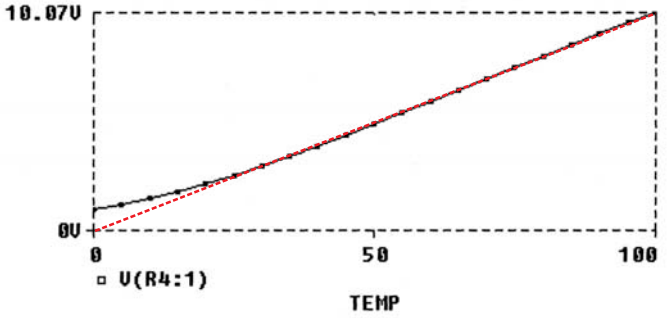


Figure 4.34: *Figure 39.3c (adapted) from (Koza et al., 1999) shows the output voltage of the temperature sensing circuit shown in Figure 4.35 plotted as a function of the circuit temperature T. The circuit is evaluated at the same values of input voltage and circuit temperature used to plot the curves of Figure 4.33. The markers correspond to the values of temperature at which the circuit was simulated to evaluate its fitness, and the dotted line represents the ideal linear relationship between T and V_o.*

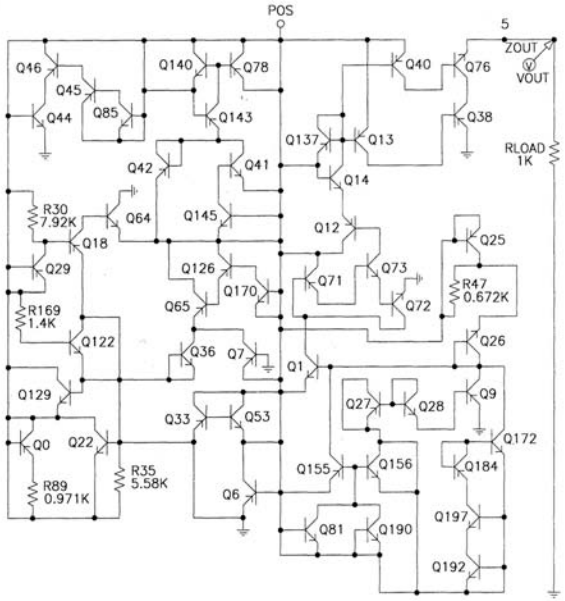


Figure 4.35: *Figure 49.5 from (Koza et al., 1999) shows a temperature sensing circuit evolved with genetic programming (GP). The prevalence of the number transistors over that of the non-null connecting resistors is even more apparent here than in the evolved voltage reference circuit shown in Figure 4.28. Correspondingly, the evolved temperature sensing circuit evolved with the sequence matching technique shown in Figure 4.32 confirms the prevalence of the number resistors over that of the devices decoded from the genome in circuits evolved with the sequence matching technique, which was already observed in the evolved voltage reference circuit drawn in Figure 4.25. Note that contrary to the circuit of Figure 4.32, this circuit is not connected to a fixed voltage source negative with respect to ground, although such a source was available to the evolving circuit in this experiment.*

4.3.3 Experiment 13: Evolution of a Gaussian function generator

The third experiment of network evolution is aimed at the synthesis of a Gaussian function generator electronic circuit. Figure 4.36 shows the devices of the predefined external circuit: it is composed of a fixed voltage source V_p, and a variable voltage source V_c connected in series to the resistor R_c, and by a "load" voltage source V_L. The goal of the evolutionary experiment is the synthesis of a circuit producing through the load voltage source an output current I_o that is a (non normalized) Gaussian function of the variable input voltage V_c in the range $2\text{V} \leq V_c \leq 3\text{V}$, with a peak value $I_{o_{max}} = 80\text{nA}$ in correspondence of $\overline{V}_c = 2.5\text{V}$, and a "standard deviation" $\sigma = 0.1\text{V}$. In formulas

$$I_o(V_c) = I_{o_{max}} e^{-\frac{(V_c - \overline{V}_c)^2}{2\sigma^2}} = 80 \times 10^{-9}\, e^{-\frac{(V_c - 2.5)^2}{0.02}} \qquad 2\text{V} \leq V_c \leq 3\text{V} \qquad (4.3)$$

Like the previous two, this analog circuit evolution problem is derived from (Koza et al., 1999), where its original ideation is attributed to Adrian Stoica.

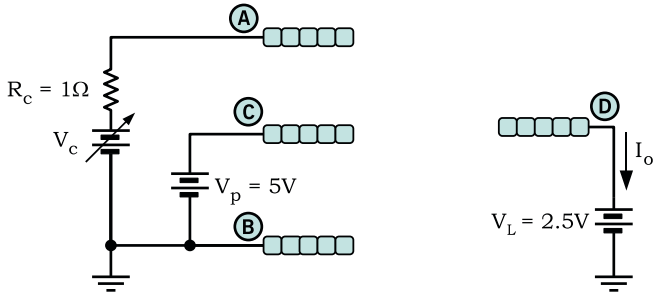


Figure 4.36: *The devices of the external network in the experiment of evolution of a Gaussian function generator circuit. The connection of the evolved network with the devices of the external network is based on the use of I/O ports associated with the external devices. The external network is composed of a variable voltage source V_c connected to a series resistance R_c, a fixed voltage source V_p, and another fixed voltage source V_L through which the output current I_0 of the evolved network is measured.*

Fitness function Using SPICE the output current I_{o_i} of the evolved circuits is evaluated in correspondence of the discrete set $\{V_{c_i}\} = \{V_{c_o}, V_{c_1}, \ldots, V_{c_{100}}\} = \{2\mathrm{V}, 2.01\mathrm{V}, 2.02\mathrm{V}, \ldots, 3\mathrm{V}\}$ of 101 equispaced values of voltage in the range of interest. The difference $(I_{o_i} - I^*_{o_i})$ between the output current I_{o_i} obtained for $V_c = V_{c_i}$ and the desired output voltage $V^*_{o_i} = I_o(V_c)$ given by equation 4.36 is calculated. The fitness function f is defined as follows

$$f = -10^{14} \sum_i (I_{o_i} - I^*_{o_i})^2$$

where the leading multiplicative factor is used to scale the results in a range comparable of that of the previous experiments. The definition of the fitness implies that the values of fitness are negative when there is a discrepancy between any of the computed output values of current I_{o_i} and the desired output current value $I^*_{o_i}$. A null value of fitness signals the attainment of an output voltage that complies exactly with the requirements.

Device set The device set for the current experiment contains two MOSFET transistors: a PMOS and an NMOS. The two models have a fixed channel length of $10\mu\mathrm{m}$ and an evolvable channel width that can vary in the range $[10\mu\mathrm{m}, 200\mu\mathrm{m}]$. The presence of an evolvable parameter in the elements of the device set implies the choice of a technique of genetic representation and decoding of the parameters, that is, the definition of the parameter map (Subsection 3.3.2 on page 72). In this experiment the parameter map is obtained using a fixed sequence of length $l_p = 20$ randomly generated from the genetic alphabet, and a logarithmic quantization of the parameter values, with

- minimum channel width $w_{min} = 10\,\mu\mathrm{m}$
  maximum channel width $w_{max} = 200\,\mu\mathrm{m}$
- alignment score associated with w_{min} $i_{w_{min}} = 20$
- number of channel width values per decade $n_{wd} = 20$

From which follows

- number of channel width values $n_g = 27$
- alignment score associated with w_{max} $i_{max} = 46$
- base of exponential decoder $\alpha \approx 1.122$

The other details of the SPICE transistor models used can be obtained from the SPICE deck relative to this experiment shown in Table D.3 of Appendix D. Note that the bulk terminal of the PMOS transistors is connected by default to the positive terminal of V_p, and the bulk terminal of the NMOS transistors is connected by default to ground.

Initial population Using the same symbols used for the parameters of the previous experiment

✓ number of chromosomes in the initial genome	$n_{ic} = 1$
number of devices in the initial genome	$n_{id} = 10$
number of copies of each I/O port in the initial genome	$n_{ip} = 2$
✓ length of sequences for terminals in the initial genome	$l_{it} = 20$
length of spacer sequences in the initial genome	$l_{is} = 20$

All the other evolutionary parameters are those specified in Experiment 11.

Results Figure 4.37 shows the result of 4 repetitions of Experiment 13. The maximum fitness data reveals that the 10000 generations used in the previous two experiments were not sufficient to synthesize a circuit realizing the required function in an acceptable way. However, all the four runs eventually attained that objective within 30000 generations.

Figure 4.39 shows an example of evolved Gaussian function generator circuit from one of the successful runs. The performance of this circuit is illustrated in Figure 4.40. Figure 4.41 shows the best circuit produced after 36 generations using a population size of 640,000 individuals in the experiment of evolution of a Gaussian function generator with genetic programming (GP) described in Chapter 51 of (Koza et al., 1999), and Figure 4.38 shows the schematics of the corresponding evolved circuit.

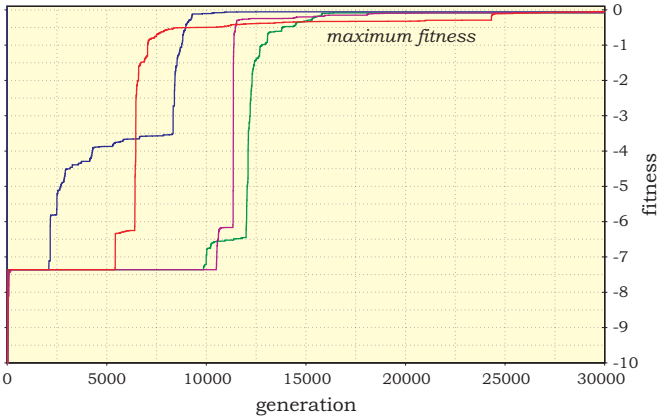


Figure 4.37: *The result of four repetitions of Experiment 13 aimed at the evolution of a Gaussian function generator circuit. A better performance corresponds to higher values of fitness, with a maximum value of zero corresponding to the exact realization of the required circuit function. The trend of the maximum fitness curves and the number of generations increased to 30000 from the value of 10000 used in the two previous examples of network evolution suggest that this problem poses a greater challenge to the evolutionary system. Nonetheless, the curves show that all the runs have eventually produced least one circuit realizing fitness very close to the maximum attainable value of zero.*

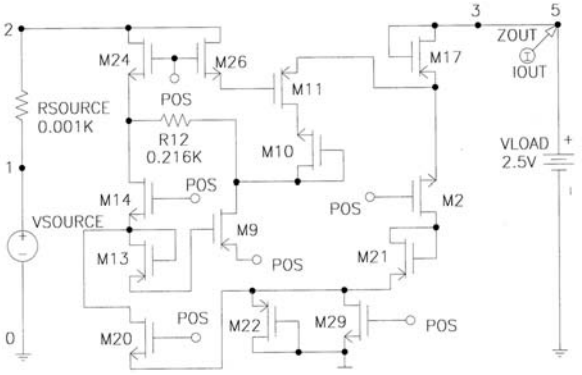


Figure 4.38: *Figure 51.17 from (Koza et al., 1999) shows a Gaussian function generator circuit evolved with genetic programming.*

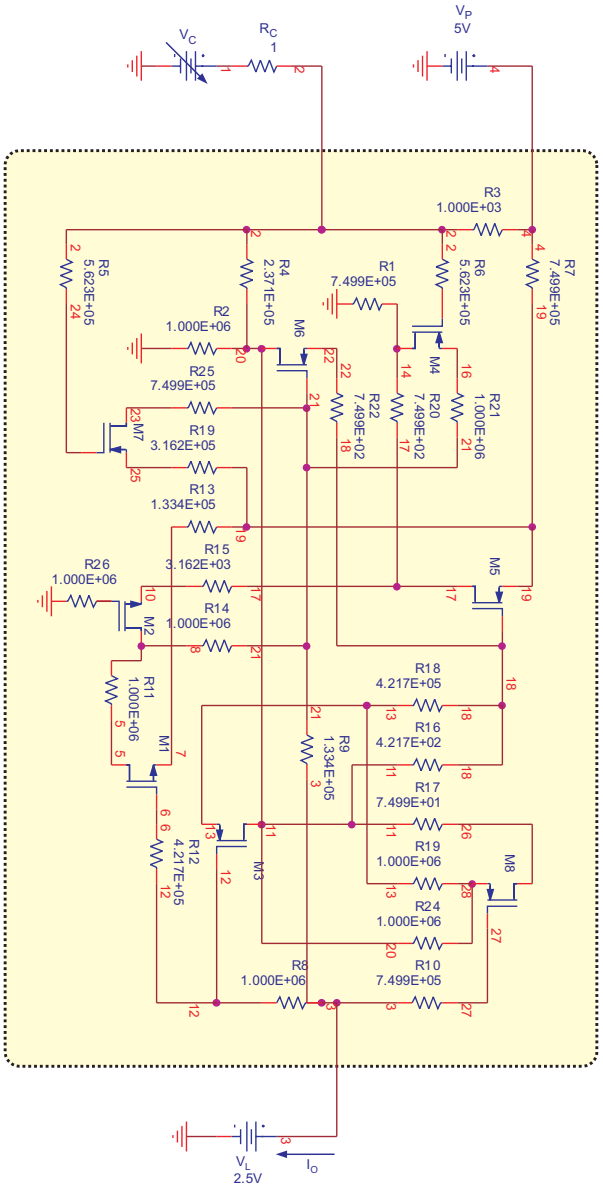


Figure 4.39: *An example of Gaussian function generator circuit evolved in the context of Experiment 13. The devices of the preassigned external circuit are drawn outside of the dotted box. The SPICE input file corresponding to this circuit is shown in Table D.3 of Appendix D.*

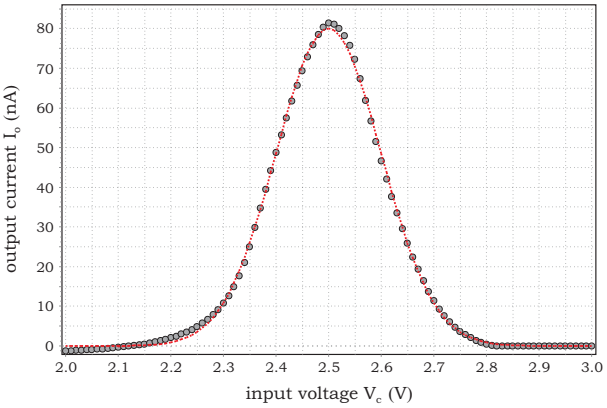


Figure 4.40: *The output current I_0 of the evolved Gaussian function generator circuit shown in Figure 4.39 plotted as a function of the input voltage V_c. The markers are drawn in correspondence of the values of input voltage used to evaluate the circuit fitness. The dotted line represents the ideal Gaussian relationship between V_c and I_o. The circuit realizes a reasonable approximation of the required function, as witnessed by the closeness of the markers to the dotted curve, although some appreciable discrepancy still remains between some part of the curve and the markers.*

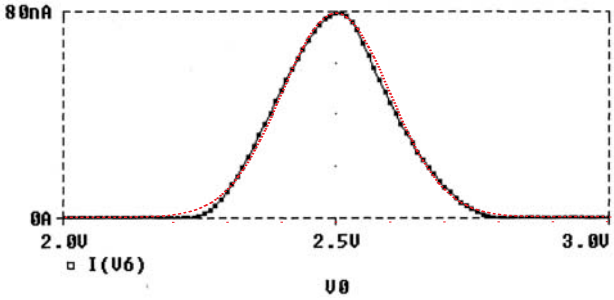


Figure 4.41: *Figure 51.18 (adapted) from (Koza et al., 1999) shows the output current of the evolved Gaussian function generator circuit shown in Figure 4.38 plotted as a function of the input voltage. The circuit is evaluated at the same values of input voltage and circuit temperature used to plot the curves of Figure 4.40. The markers correspond to the values of input voltage at which the circuit was simulated to evaluate its fitness, and the dotted line represents the ideal Gaussian relationship between the input voltage and the output current.*

4.3.4 Experiment 14: Evolution of a XOR function neural network

The goal of this last experiment of network evolution is the synthesis of a neural network realizing the XOR function. Figure 4.42 shows the predefined external devices: two neurons X_0 and X_1 whose output constitutes the input of the evolved network, a bias neuron with constant output, and a neuron Y whose output is the output of the whole network. The Y neuron is a sigmoidal neuron governed by the following equation

$$y = \frac{1}{1 + \exp(-5 \sum_j w_j x_j)} \tag{4.4}$$

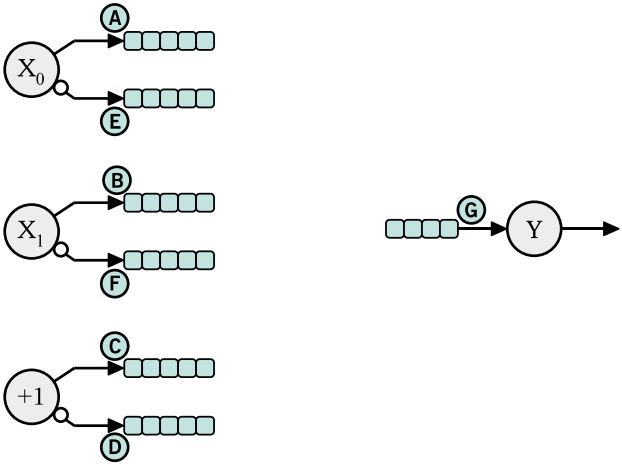


Figure 4.42: *The devices of the external network in the experiment of evolution of a neural network realizing the XOR function. The connection of the evolved network with the devices of the external network is based on the use of I/O ports associated with the external devices. X_0 and X_1 are the two input neurons, the "+1" neuron is the fixed input bias neuron, and Y is the sigmoidal output neuron. In order to simplify the logarithmic quantization of the connection weights, all the input neurons produce also the inverted signal (the inversion is represented graphically by the small circles). In this way the quantization of the interaction strengths can be limited to non-negative values.*

where y is the output signal, the x_js are the input signals, and the w_j are the input connection weights. The evolutionary goal corresponds to the requirement that the function $Y(X_0, X_1)$ realized by the network be the XOR function $Y^*(X_0, X_1)$ defined by $\{Y^*(0,0) = 0, Y^*(0,1) = 1, Y^*(1,0) = 1, Y^*(1,1) = 0\}$. Note that this evolutionary requirement specifies the input-output mapping of the network only in correspondence of binary values of the inputs X_o and X_1 even if the network realizes actually a mapping $\mathbb{R}^2 \to \mathbb{R}$.

Fitness function Like in the case of the Experiment 11 of voltage reference circuit synthesis, to define the fitness function for the XOR evolutionary experiment we define a tolerance interval of width $2\Delta Y = 0.002$ centered on the required output values, and specify that output voltage values falling into that interval will not be penalized in terms of fitness. This choice is due to the fact that sigmoidal neuron models such as that specified by Equation 4.4 realize binary-valued outputs only asymptotically, and in the absence of a tolerance interval this would induce evolution to produce larger and larger networks just to get closer and closer to the asymptotic value even when the required binary function has for all practical purposes already been realized. We thus define the following expression for the error in correspondence of each input pair

$$\varepsilon(X_0, X_1) = \begin{cases} (Y(X_0, X_1) - Y^*(X_0, X_1))^2 & \text{if } |Y(X_0, X_1) - Y^*(X_0, X_1)| \geq \Delta Y \\ 0 & \text{if } |Y(X_0, X_1) - Y^*(X_0, X_1)| < \Delta Y \end{cases}$$

from which we derive the following expression for the fitness f

$$f = -(\varepsilon(0,0) + \varepsilon(1,0) + \varepsilon(0,1) + \varepsilon(1,1))$$

This means that the values of fitness are negative when there is a discrepancy exceeding the tolerance ΔY between the computed output Y and the desired output Y^* for some of the binary input patterns. A null value of fitness signals the attainment of an output that falls within the tolerance interval in correspondence of all the binary input patterns.

Network-specific interaction map For the reasons explained in Appendix B, we use a network-specific interaction map that realizes a logarithmic quantization of the weight values assigned to the connections between outputs and inputs of the neurons. We assign in particular

$\forall$ minimum non null weight value $w_{min} = 0.001$

maximum weight value $w_{max} = 1000$

- alignment score associated with w_{min} $i_{w_{min}} = 1$
- number of weight values per decade $n_d = 6$

From which follows

- number of weight values apart from $w_0 = 0$ $n_g = 37$
- alignment score associated with w_{max} $i_{max} = 37$
- base of exponential decoder $\alpha \approx 1.468$

With this definition of the network-specific interaction map we are defining the genetic representation only for *non negative* weights. Since we can expect our neural network to require in general signed weights in order to realize the required functionalities, we define all the external neurons constituting an input to the evolved network as being provided with both the direct and the inverted output (Figure 4.42).

Device set The devices available to the evolving circuit are two kinds of sigmoid neurons - one excitatory and the other inhibitory - implementing the model specified by Equation 4.4.

All the other evolutionary parameters are those specified for Experiment 11.

Results Figure 4.43 shows the result of 25 repetitions of Experiment 14. The curves of the maximum fitness show that all the runs evolved within 200 generations at least one network realizing the required function with the given tolerance. The runs were continued for several hundreds of generations after the attainment of the required solution in order to illustrate the interesting phenomenon of initial increase in the number of neurons decoded from the genome, followed by the consistent reduction of this number to one or two neurons (the length of the genome shows an analogous trend of initial increase and subsequent reduction). This behavior is interesting as it is known that the XOR problem can be solved with just one hidden neuron (Haykin, 1999), and the genome of the individuals in the initial population contain just one hidden neuron. However, Figure 4.43 shows that the evolution of a functional network appears to be facilitated by the initial expansion of the network to higher levels of complexity in terms of hidden neurons.

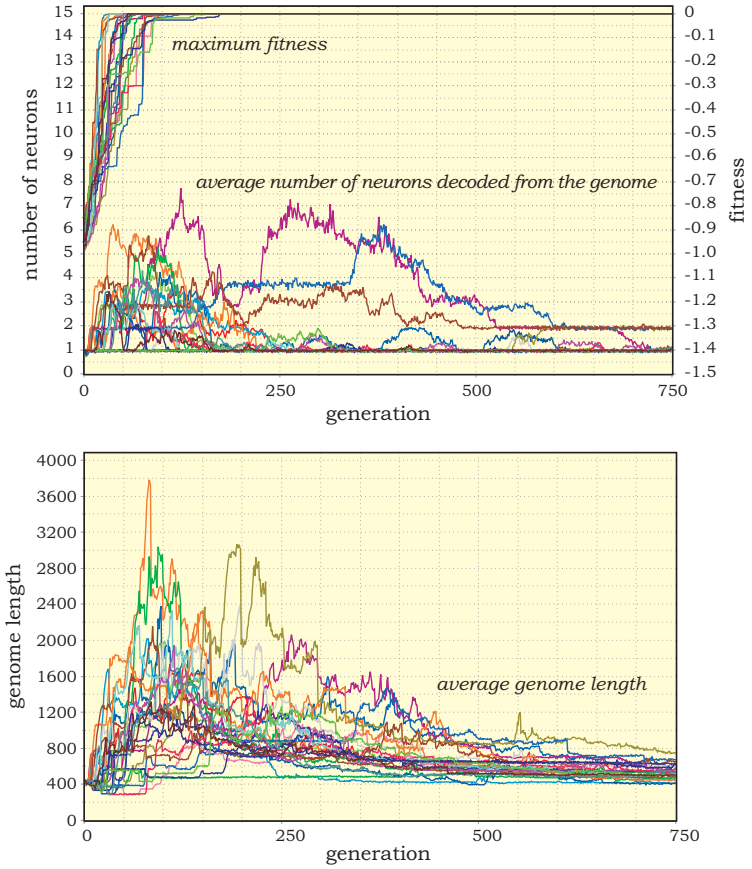


Figure 4.43: *The result of 25 repetitions of Experiment 14 aimed at the synthesis of a neural network realizing the XOR function. Better performance corresponds to higher values of fitness, with a perfect solution having a fitness value of zero. The curves of maximum fitness (top panel, top curves) show that all runs evolve the required function within 200 generations. The curves of the average number of neurons decoded from the genome (top panel, bottom curves) reveal that most runs synthesized initially networks containing several hidden neurons, but later progressed towards networks having one or two hidden neurons on average. The curves of the average genome length (bottom panel) show that the length of the genome follows a similar trend.*

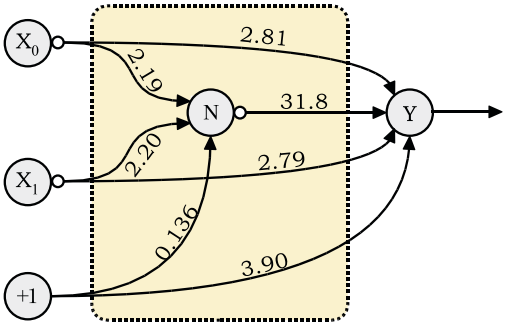


Figure 4.44: *An example of neural network evolved in the context of Experiment 14, aimed at the evolution of network realizing the XOR function. The devices of the preassigned external network are drawn outside of the dotted box. The evolutionary process has generated a genome encoding a single neuron N, realizing the simplest network that is known to be required to solve the XOR problem with the model of neuron used in the experiment (Haykin, 1999). Evolution has also produced the connections between the neuron N, the input neurons X_0, X_1, the bias neuron, and the output neuron Y. Note that evolution used the inverted input signals for X_0 and X_1, and an inhibitory neuron for N. The genome corresponding to this network is shown in Figure 4.46.*

Moreover, this figure reveals that the complexity of the networks can both increase and decrease during the evolution, thanks to the action of the genetic operators that can both create new device descriptors and invalidate existing ones.

Figure 4.44 shows an example of evolved neural network realizing the maximum fitness, and Figure 4.45 shows the plot of the input-output function realized by that network. Figure 4.46 shows the genome from which the network of Figure 4.44 was decoded, and Figure 4.47 shows the genome of the 100 individuals composing the evolved population from which this individual was extracted. Each individual of the population corresponds in Figure 4.47 to a row in the color table, and each letter in the genetic alphabet is represented with a a different color. The genome shown in Figure 4.46 corresponds to the first row of the table. The figure shows that there is a great deal of similarity between the genome of the individual of the population, especially in the initial fragment of the genome. This rapid convergence of the

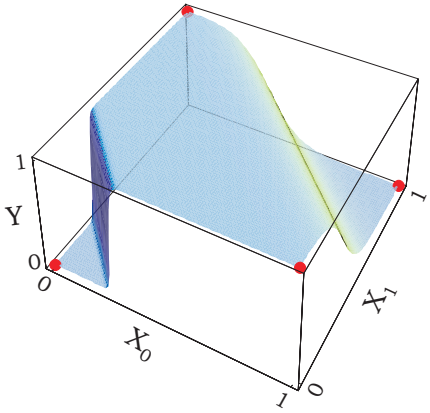


Figure 4.45: *The function $Y = f(X_0, X_1)$ realized by the evolved neural network shown in Figure 4.44. The four dots at the corners of the bounding box correspond to the specification of the XOR function, and reveal that the evolved neural network realizes indeed the required XOR function.*

UTEQABQXLMRGB**NESI**NQOMZADGTPVYDDFPTFQFLCBYLMMJQBTUWGWUJTZ
VUTERMPPMYVYAPGAOBCBHEEJBOTERMOCMTMAYYMMGMRBYB**IOPTGPQYRQ**
TNSVKBZDLNPQKDFPBQLPMSPMIEODQHMCMXLIFQATVGNSTERMYQVIFJAK
EAMQXKYZKCV**IOPTCJTGKZTFPMBWSKQLJAJHUOPTERM**KMIVGLSBTHDUIC
IOPTGLPGCBNHQQALPMBJJTERMCWVQDJLKLPC**IOPTGYJWGUXQGYPMBJMZ**
JLTERMEAZFRNOWHLLNFRIEGNG**IOPTELPMSIXBSQKHGMRSIYFLTERM**BIE
CIYXCQAOXTXOLW**IOPTCUTLGAFPMBEXFSNHUXPNAETERM**LSSRKDRJNMGT
KTAKMRJ**IOPTFHFHLCMAQDSJLMMLMFBROTZNBUETERM**KME**IOPTE**IOSPMY
WQAFBJHHLEOEFFXTERMZLGZBRBEJLOERNF**IOPTFUQFYHLCMVYZMLMZPF**
BEDMEPTERMDADGOMMIUDZFKLUAMF

Figure 4.46: *The genome corresponding to the neural network shown in Figure 4.44. The genome is constituted by a single chromosome. The fragments corresponding to the tokens for the neuron, the I/O ports, and the terminals are underlined. In particular, the **NESI** token corresponds to an inhibitory sigmoidal neuron without evolvable parameters. The genome sequences that are associated with the device terminals are shown in boldface. The remaining characters constitute non-coding genome.*

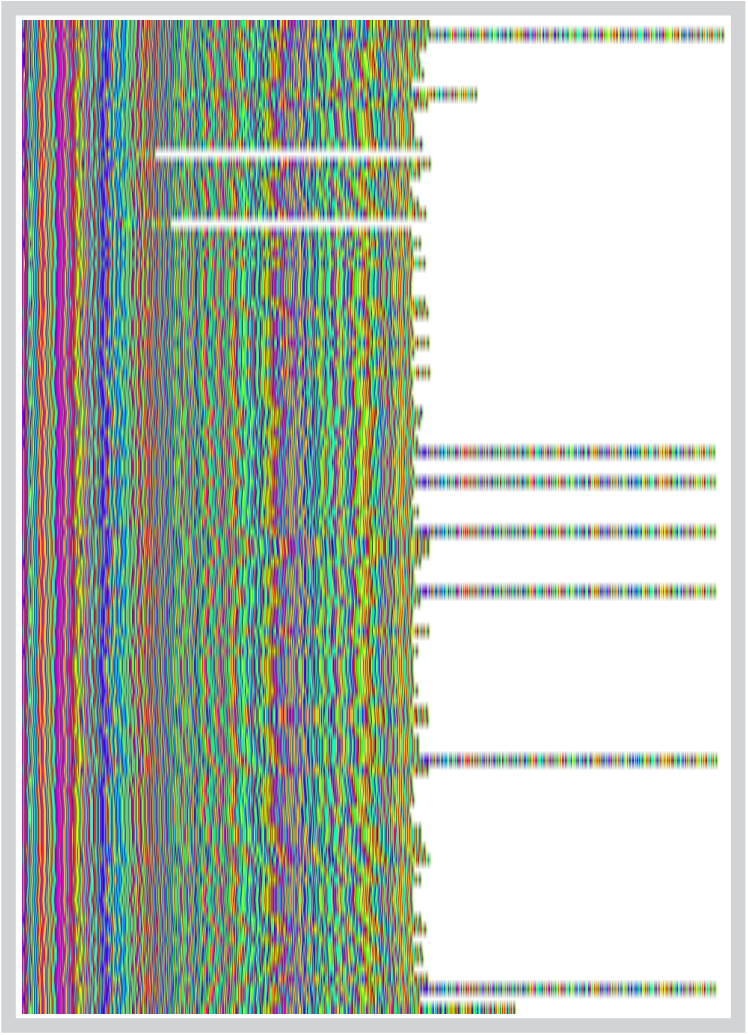


Figure 4.47: *The genome of the 100 individuals composing the evolved population from which the network shown in Figure 4.44 was extracted. The genome of one individual corresponds to a row in this color table, and each letter in the genetic alphabet is represented with a a different color. The genome shown in Figure 4.46 corresponds to the top row of this table.*

evolutionary population to a few kinds of genome or, in other words, this loss of diversity of evolutionary population (see also Appendix A on the issue of population diversity) is not unexpected, as it has been already observed, analyzed and commented in the past (for example (Harvey, 1995; Goldberg, 2002)). Note that Figure 4.46 reveals that the evolutionarily highly conserved initial fragment of genome contains the sequence (the "gene") coding for the hidden neuron of the network shown in Figure 4.44.

4.3.5 Discussion

The examples of evolution of analog networks presented in this section show that the evolutionary system based on sequence matching can indeed synthesise high-performance exemplars of this kind of network. There are certainly many ways in which the few examples shown here could be extended. A first obvious extension in the case of neural network is the use of dynamic neuron models in place of the static model used in Experiment 14, in the context of more realistic problems than the simple XOR function.

Concerning the examples of analog electronic circuits synthesis illustrated above, it must be noted that the requirements represented by the simple fitness functions used do not make the evolved circuits comparable to circuits designed with traditional engineering synthesis approaches. A real-world circuit is typically required to comply with scores of specifications as can be inferred, for example, from the data sheet of any commercial analog integrated circuit. Moreover, the circuit must realize the required specification with the parameters of the devices that are not exactly specified but can vary from specimen to specimen of a given type of device, whereas the SPICE simulations for the evolutionary examples presented in this section were done with fixed sets of device parameters. These problems can be tackled by running more than one circuit simulation for each individual, each testing a different set of specifications, and by assigning to the device parameters during each simulation different values randomly sampled in the whole admissible range. Since the execution of each circuit simulation is very costly in computational terms, especially for the simulations performed in the time domain, this would result in more computationally expensive evolutionary runs. The computational power required to tackle non trivial circuit synthesis problems with this more inclusive approach is just now becoming available, as witnessed by the results

reported in (Koza et al., 2003). Even assuming as available the required computational resources, there remains the problem of using evolutionary algorithms that can deal with the multiple specifications that a single circuit is required to meet. Although many multi-objective evolutionary algorithms exist for this purpose (see, for example, Deb, 2001), they must be tailored to the needs of electronic circuit synthesis. This means in particular that it must be possible to remove the evolutionary pressure on a certain aspect of the circuit behavior once the corresponding specification is met, even if a better performance than the one attained is conceivable for that particular behavior. This corresponds to the fact that a commercial circuit is judged acceptable when its performance meets the required specifications, without any Pareto-ranking (Deb, 2001) being defined for the set of manufactured circuits.

4.4 Summary

The results of the series of experiments described in this chapter demonstrate that the system defined in the previous chapter displays a satisfying evolutionary behavior at various levels. The series of sequence and network matching experiments confirm the suitability of local sequence alignment for the implementation of the sequence interaction map that contributes to compose the device interaction map, both in the abstract evolutionary scenario focused on sequences of Experiments 1 to 7, and in a more network-oriented context of Experiments 8 to 10. Finally, the experiments of network evolution described in the last section show that the evolutionary system can be used to evolve non trivial examples of analog networks with a reasonable computational effort and achieving results that compare favourably with the best results existing in the literature.

Chapter 5

Further issues and conclusions

Overview

Besides summarizing the results and pointing to possible extensions of the work reported in the thesis, this last chapter examines the consequences of the level of abstraction at which the workings of biological systems was implemented in the evolutionary system defined in the thesis. In particular, it considers the consequences of the absence of interaction of the genome with the dynamics of the evolved analog network, which stems from the decision to avoid basing the workings of the evolutionary system on the implementation a low-level dynamics.

5.1 The consequences of abstraction

The results presented in the previous chapter which illustrate the application of the evolutionary system defined in this thesis to the evolution of analog networks confirm the evolutionary potentialities of the system and the validity of the approach that combines von Neumann's insights about the evolutionary growth of complexity and those derived from the observation of the workings of the fundamental levels of organization of biological systems. Since the characteristics of our artificial evolutionary system that were derived from the knowledge of biological systems correspond to their biological counterparts only in an abstract way, it is worth examining the main discrepancies between our system and its biological source of inspiration, trying to estimate the consequences of this abstraction.

In this analysis we must distinguish the many aspects of biological systems that we have not considered in this thesis but which could be easily accommodated in its framework since they comply with the adopted approach, from the aspects where the difference between the biological and the artificial system is substantial. To the first class belong properties like modularity, hierarchical organization, and the existence of a developmental process, which were only briefly mentioned here and there, or for which possible implementation techniques were described but were not required to solve the problems considered in the previous chapter and thus were not exhibited in action. We will still add some further remarks about these aspects below, in context of the discussion about the importance of implementing and constraining a realistic dynamics. To the second class of properties belongs in particular the absence of an interaction between the operation of the analog network and the genome from which it has been decoded. As argued in the next section, this limitation stems from the decision to avoid the implementation of a low-level dynamics.

5.2 Low-level dynamics and constraints

In general, when it comes to combine the idea of evolution with a synthetic approach, we are confronted with the issue of defining the entities and the rules that form the backcloth of our endeavor. We can opt for a very fine granularity, or choose a higher-level description that abstracts many details of the low-level entities. The option of a fine

granularity, corresponding to the implementation of a rich low-level dynamics, is very tempting, since in theory it permits the faithful replication of all the relevant details. This, however, is accompanied by the risk of also implementing many irrelevant details. Before the advent of molecular biology, it was not clear if the workings of biological systems could be understood in terms of the hitherto known laws of physics. Molecular biology has since shown us that biological systems flourish on the harnessing of those laws. Howard Pattee (Pattee, 1995a) has expressed this harnessing in terms of constraints that biological systems impose on the dynamics ensuing from physical laws. In other words, natural selection leads to the formation of structures whose presence influences the dynamics of the surrounding space-time in ways that favor their persistence and, eventually, their self-reproduction and evolution.

Implementing a physically convincing low-level dynamics in our artificial evolutionary systems we could hope to observe the same phenomenon of emergence of dynamic constraints in our evolutionary experiments. As argued by Toffoli (1994), it is not too difficult to obtain a physically convincing dynamics from a very finely grained system implemented, for example, with a cellular automaton. To obtain this result, it is in fact sufficient to endow the rules of the automaton with some basic conservation and symmetry properties. However, a lot of averaging is required to observe anything interesting within such a system. This means that the computational effort required to implement a dynamics allowing the evolution of Pattee's life-like constraints is probably huge. On the other hand, the definition of a low-level dynamics where no massive averaging is required in order to observe interesting evolutionary phenomena appears as a quite challenging problem in itself. This was suggested by the results of a series of exploratory experiments conducted in the context of the research that led to this thesis, similar in spirit to those reported in (Taylor, 2004). Those exploratory experiments based on the use of cellular automata suggested the presence of a difficulty in reconciling the existence of non-trivial dynamic phenomena at the lowest level of definition of the dynamics, and the robustness of the structures that could be evolved to act as constraints for the same dynamics. Probably the same kind of observations led in (Taylor, 2004) to the use of a genome layer generating the constraints which is separate from the cellular automaton layer where the low-level dynamics unfolds. With this choice, however, the problem of the absence of interaction between the genome and the dynamics

which it controls remains unsolved.

The approach adopted in this thesis accepts this absence of interaction but tries at least to define a rich dynamics at the next level, that of the analog network. The particular emphasis that was put on the use of electronic circuit simulators, which implement in an efficient and physically sound way the dynamics of real-world devices, was motivated precisely by the desire to obtain this kind of dynamics at least at this level. As noted in Chapter 2, active electronic devices like transistors can be considered as structures that constrain the physical dynamics just like the structures that evolved in biological systems for the same purpose. Adding to this fact the particular attention that was put in this thesis on the representation of the interactions between the devices (possibly belonging to different compartments of the evolved system) results in the possibility of realizing the evolutionary scenario suggested by Pattee at least from the level of the analog network up to higher levels of organization of the evolved system

The absence of interaction of our genome with the analog network that is decoded from it has of course some consequences. Three of them appear of particular importance in an evolutionary context. The first consequence is the absence of an intrinsic mechanism of activation and inactivation of parts of the genome from the part of the dynamics of the analog network. In biological system it is known that the analog network defined by the genome is in a state of constant flux concerning its structure, and that the genome can be marked for activation and inactivation of parts of it as a consequence of the activity of the signaling and genetic regulatory network, including the possibility of have a long-term memory of this state of activation and inactivation (Lewin, 2004; Marijuán, 1996). This has important consequences in terms of specialization of the cells and in terms of the developmental processes that can be based on that mechanism. Since our artificial system is decoded once an for all, this possibility of a dynamic network structure is not directly accessible. Our artificial system, however, has the possibility of evolving something similar in terms of, for example, memory cells (for example bistable network structures) generating signals that inhibit selectively parts of the network (possibly in response to signals coming from higher levels of organization). At most, an additional inhibitory input must be added to the devices used in the evolutionary experiment to permit their activation and deactivation. Thus this first limitation can be easily overcome with simple modifications of the existing system.

A second consequence of the absence of interaction of the genome with the dynamics of the evolved analog network is the absence of a mechanism of evolution of the "constructor", that is, of the rules that determine the decoding of the genome into the analog network. Although some kind of mechanism could be defined and encoded in the genome to implement some kind of evolvability of this aspect of the evolutionary system, in the absence of an actual low-level dynamics there seem to be no obvious way to define this mechanism in an evolutionary open way, that is, without deciding from the start what kind of changes are permitted to occur to the decoding process. This approach is thus a palliative rather than a cure, and would not endow our artificial evolutionary system with the same flexibility and open-endedness of the constructor possessed by biological systems. There seems to be therefore no solution other than to accept this limitation of our system and focus our efforts – like we did in the previous chapters – on a definition of the decoding system aimed at endowing it from the start with the required evolutionary properties.

The third consequence of the absence of interaction of the genome with the dynamics of the evolved analog network is the absence of a natural way to redefine the kinds of reorganizations to which the genome can be subjected, and the rate at which these reorganization occur. The first aspect, like the redefinition of the constructor discussed in the previous paragraph, has no obvious evolutionarily open solution in the context of our system. Therefore we must also accept it, focusing our efforts on the definition of the operators that we build from the start into our system. The second aspect, that is, the variation of the mutation rates, can instead be solved at least partially by genetically encoding the reorganization rates, possibly making them depend on the activity of the analog network. Since the reorganizability appears more and more to play an central role in the evolution of existing living organisms (Shapiro, 2002), the next section is devoted to the analysis of this important topic.

5.3 Natural genetic engineering

The conventional view of the evolution of the genome of biological organisms sees a genome subject to a small rate of mutation – consisting mostly of substitutions, deletions and insertions of single nucleotides – possibly accompanied by the recombination of pairs of chromosomes.

The mutation process is typically seen as an effect suffered by the organism given the impossibility to correct all the errors that, for various reasons, accumulate in the genome. The current scenario of genome reorganization offered by molecular genetics, however, is much more complex and much more active than that, as witnessed by the following extract

> Evolutionary genomic change occurs largely by a process of Natural Genetic Engineering. Systemic genome organization means that new functions arise by the cut-and-splice rearrangements of genetic modules. Living cells possess mobile genetic elements and other biochemical functions which carry out the underlying DNA rearrangements. Cells regulate the activation of natural genetic engineering functions. Thus cells have a capacity for major genome reorganization in response to evolutionary crisis. Moreover, the fact that natural genetic engineering changes are neither random in nature nor restricted to a single site in the genome means that they can create novel distributed (multilocus) systems and new genome system architectures. (Shapiro, 2002, p. 746)

Note that the statement that the genetic changes are not random refers to the fact that the recombination takes into account the structure of the genome, for example, that corresponding to genes, regulatory regions, or protein domains, instead of operating on the genome seen as a collection of unstructured sequences of nucleotides.

The good news that come to our artificial evolutionary system from the realization of the existence of this phenomenon of natural genetic engineering, is the fact that our system is intrinsically endowed with all the required genetic operators and with the possibility of reorganizing the genome taking into account its structure (which consists in the existence of device descriptors, tokens, sequences associated with terminals and parameters, and so on). Moreover, the process of decoding of the artificial genome produces all the information about this structure, without the need of additional computational effort for its exposure and exploitation. An example of recombination that uses this kind of information is the insertion of new device descriptors in the genome using as sequences associated with the terminals fragments of sequences associated with the terminals of the devices already encoded in the genome (Subsection 3.6.2, page 113).

The fact that the activation of major genome reorganization happens as a response to evolutionary crisis, on the other hand, requires some additional thought since it is not clear what must be considered an evolutionary crisis in our artificial evolutionary context. A possible candidate could be the presence of a phase of stagnation of the evolutionary process, signaled for example by a plateau of the maximum value of fitness attained. This simple approach, however, does not appear to capture the essence of the concept of an evolutionary crisis, if only for the fact that the stagnation of the fitness does not necessarily correspond to a stagnation of the population in the genome space. A more promising approach is based on the concept of viability of the evolved systems (Aubin, 2000; Mattiussi and Floreano, 2003). It defines an evolutionary crisis as a condition where the individuals of the population are not able to maintain a sufficient margin of viability in the prevailing environmental conditions.

An example of this kind of condition taken from the experiments of Chapter 4 would be a change in the range of variation of the input voltage in the experiment of evolution of a voltage reference described in Subsection 4.3.1. To reformulate the original evolutionary problem in terms of viability, we must redefine the goal of the evolutionary process. From the original goal constituted by the attainment of the highest possible value fitness, we turn to a goal constituted by the preservation of the viability represented by the maintenance of the output voltage within a given range. Assuming that the individuals of an evolved population can comply with this viability constraint, the change in the input voltage range would precipitate an evolutionary crisis by taking the output voltage of most individuals near its admissible limits of variation.[1] It is clear, however, that this approach implies a major redefinition of the traditional approach to evolutionary computation (Mattiussi and Floreano, 2003) that cannot be undertaken here and now. We are therefore content with the observation that the kind of genetic representation defined in this thesis is ideally suited to the implementation of a process corresponding to Shapiro's natural genetic engineering. Note, finally, that in addition to the ex-

[1]Note that it is possible to have the activity of the network determine the rate of genetic reorganization by considering, for example, some external components to which the evolved network can connect to localize a value of voltage or current that is interpreted as a modifier of the default rate of application of a given genetic operation. Thus, an evolutionary crisis of the kind described above for the voltage reference circuit would have the possibility of producing consequences in terms of genome reorganization.

istence of environment-dependent variations of the process of genome reorganization described above, it is possible to conceive the existence of heritable variations of the rate and form of the process of genome reorganization, acting either at a local or at a global level (Metzgar and Wills, 2000), and that those phenomena can be also implemented in the context of the evolutionary system defined in this thesis.

5.4 Conclusion

In this thesis we have described a new kind of evolutionary system that was explicitly defined to accommodate the possibility of the growth of complexity of the evolved systems. It applies to a class of systems that we called analog networks and which include many systems of considerable practical interest, such as analog electronic circuits, neural networks, and metabolic, signaling, and genetic regulatory networks, the systematic design and reverse engineering of which constitutes still an open problem. The experiments of analog network evolution reported in this thesis confirm the compliance of the evolutionary system that has been developed, with its stated aims.

Many aspects of the proposed evolutionary systems, such as the compartmentalization and hierarchical organization of the evolved systems, the mechanism of interaction silencing, the application to fields other than electronic circuits and neural networks, and still others, like those considered in this chapter, were described or mentioned but were not required to solve the problems considered, or their exploration could not find place in the research effort reported in this thesis. Most of these topics can be expected to represent, not trivial applications of the principles expounded, but interesting fields of enquiry for further research efforts, exciting and potentially rewarding as that reported in this thesis. Far from considering it a limitation, we see this fact as an additional strength of the present work. In the words of Richard Bellman (1968), "successful research depends, not upon the solution of a succession of isolated individual problems, but upon the forging of connected chain of both problems and solutions. [...] Each problem should lead naturally to further problems; each solution, to further solutions."

Appendix A

New measures of diversity for populations and distances between individuals with highly reorganizable genomes[1]

Overview

This appendix considers the problem of defining a measure of diversity for a population of individuals whose genome can be subjected to major reorganizations during the evolutionary process. First, a measure of diversity for populations of strings of variable length defined on a finite alphabet is introduced. A semi-metric distance between pairs of strings is derived from the measure of diversity for populations. The definitions are based on counting the number of substrings of the strings, considered first separately and then collectively. This approach is related to the concept of linguistic complexity, whose definition we generalize from single strings to populations. Using the substring count approach a new kind of Tanimoto distance between strings is also defined. It is shown how to extend the approach to representations that are not based on strings and, in particular, to the tree-based representations used in the field of genetic programming. The concept of suffix tree of a string is introduced and it is shown how it allows the implementation of the measures of diversity and distances just defined with a computational cost that is linear in both space and time relatively to the length of the strings and the size of the population. The definitions were devised to assess the diversity of populations having genomes of variable length and variable structure during evolutionary computation runs. In particular, these definitions apply to the genome introduced in this thesis for the representation and evolution of analog networks, but applications in quantitative genomics, proteomics, and pattern recognition can be also envisaged.

[1] A version of this Appendix was published as (Mattiussi et al., 2004).

A.1 Introduction

In evolutionary computation (EC) there is often the need to measure the diversity of two or more individuals of a population. This necessity can be dictated by many reasons, for example: the desire to prevent premature convergence of the population; the utility of restarting or stopping an evolutionary algorithm when the population diversity drops below a certain threshold; the requirement of evolving a population of distinct Pareto-optimal solutions in a multi-objective optimization problem; the effort of maintaining a population able to adapt rapidly to a changed environment in the case of dynamic problems (de Jong et al., 2001; Leung et al., 1997; Tomassini et al., 2004; Wineberg and Oppacher, 2000, 2003a,b; and still many others).

The diversity of individuals and populations can be measured either in the genotype or in the phenotype space. When the phenotype or the genotype are constituted by a fixed number p of real parameters, the standard tools of mathematical analysis and those of cluster analysis in the p-dimensional real space R^p can be directly applied for the definition of a measure of diversity (Theodoridis and Koutroumbas, 2003). It is often the case, however, that the structure of the phenotype does not lend itself well to such a straightforward approach; this happens, for example, when the phenotype is a structure - say, a network - with variable topology and number of parameters. In those cases, one is left with the option of either defining a specialized distance between such phenotypic structures, or focus on the genotype space, where the structure of the elements is usually much simpler, for example a sequence of characters. We will assume in the rest of the Appendix that the elements of the genotype space, i.e., the genomes of the individuals, are finite character strings over a finite alphabet. Note that there are contexts, where the genomes have structure more complex than a string, and where the measures defined below and based on the count of strings and substrings cannot be applied directly but must be adapted to the particular genome structure. We will describe below an example of extension of the string-based diversity measure to the case of tree-based genomes used in the field of genetic programming (Keijzer, 1996; Langdon and Poli, 2002).

If the strings that constitute the genomes have fixed length and uniform structure, the definition of a diversity measure for two individuals is typically based on the use of the Hamming distance (although other approaches are possible, see for example (Leung et al., 1997)), that is,

on the count of the number of mismatches between the pair of strings that constitute the genomes of the individuals. With the expression "genomes having uniform structure", we mean that all the individuals have a genome with the same number of subblocks, or genes, with the same phenotypic meaning for the subblocks, and arranged in the same order in the genome. A diversity measure for the whole population can be then obtained from the diversity measure for pairs of individuals by combining all the pairwise distances between individuals (Morrison and De Jong, 2002; Wineberg and Oppacher, 2003a,b).

The case of strings with variable length but still with uniform structure can be treated similarly, provided the Hamming distance is generalized to permit the comparison of strings having different length. A good candidate for this generalization is the so-called *edit distance*, which is based on the use of three elementary operations - insertion, deletion, and substitution of characters - to transform a string into another. A cost is associated with each elementary operation, and the distance is defined as the minimum cost of the sequence of operations that leads from one string to the other. The correspondence established by this minimum cost sequence of operations is called a *global alignment* of the two strings, and algorithms with computational cost proportional to mn exist to perform this task, where m and n are the lengths of the two strings (Gusfield, 1997; Keller and Banzhaf, 1994; O'Reilly, 1997; Sankoff and Kruskal, 1983).

The problem of defining a measure of diversity for a population becomes more complicated if we assume that besides having variable length, the genomes of the individuals may also have different structure. For example, the genome of one individual might have a certain set of genes arranged in a certain order, and another individual might have the same set of genes but ordered differently, or it might even have a different set of genes. In these cases, the methods described above for genome comparison cannot be applied. On the other hand, the genome introduced in this thesis for the representation and evolution of analog networks falls obviously in this class and, more generally, variable structure genomes are particularly interesting, especially in the light of high reorganizability of biological genomes, which has been observed as a response of organisms to a crisis and is coming more and more into focus as one of the key players in the evolutionary potential of organisms (Shapiro, 2002). If, despite its variability, one is aware of the presence of a set of structural motifs (such as, for example, promoter sequences and specific protein coding regions in biological genomes) in

the genome of every individual, the global string alignment approach based on the edit distance can still be applied by localizing and comparing one by one the corresponding motifs in the two genomes, and by assigning a cost to unmatched motifs. Alternatively, one can resort to algorithms that implement *local alignment* in place of global alignment (Gusfield, 1997; Sankoff and Kruskal, 1983). The difference is that the costs that were associated with the elementary operations in the definition of the edit distance, are now interpreted as rewards for matches, and the best matching subsequences - presumably corresponding to functionally significant motifs - are located and evaluated by the algorithm itself within the genomes of pairs of individuals. The computational cost of these approaches, however, becomes rapidly unmanageable, especially in an EC perspective where the population is a highly dynamic entity, whose diversity must be repeatedly calculated during an evolutionary run.

The approach adopted in the present appendix tries to bring together the best of both worlds, by defining a measure of diversity for individuals and populations that applies to genomes with variable length and structure, but does neither assume the knowledge of the genome structure, nor incur the computational cost entailed by the automatic identification of the actual genome motifs. To achieve this result, it looks for *potential* motifs contained in the genomes, and bases its measure of diversity on a combination of their number. This leads to a definition that applies to generic genomes, and which is capable of taking into account at least partially the structure of the genomes, while remaining computationally inexpensive. Moreover – as can be expected from a diversity measure – the definition gives a minimal value of diversity for the case of uniform populations and a maximal value for pairwise maximally distinct individuals, and becomes a (semi-metric) distance when the population reduces to two individuals.

Despite being targeted to highly reorganizable genomes, and, in particular, to the kind of genome used in this thesis, the measures of diversity that are defined below apply also to simpler kinds of genomes, such as those having fixed length and uniform structure. Moreover, the definitions can find applications beyond EC, in any domain that requires the comparison of sequences of symbols - such as genomics, proteomics, chemical structure similarity assessment, and pattern recognition in general - and, even more generally, to domains that require the comparison of generic collections of "individuals" that can each be associated in a meaningful way with a set of features.

A.2 Population diversity

The genomes that we consider are strings of characters. Let us denote with s_{i_j} the string that constitutes the genome of the individual i_j of the population $P = \{i_1, i_2, \ldots, i_n\}$. In the following, we will identify an individual with its genome, and define diversities and distances for individuals in terms of their genome only. We are interested in counting the number of potential motifs contained in each individual genome and in the population genome. Since we do not assume any knowledge of the genome structure, the potential motifs of a string s_i constituting the genome of individual i, are all its substrings, that is, all the strings of characters that appear contiguously in s_i. Let us denote with $\mathcal{S}_i$ the set of substrings of s_i, and with $|\mathcal{S}_i|$ its cardinality. Correspondingly, the potential motifs of a set of individuals, for example the population $P = \{i_1, i_2, \ldots, i_n\}$, are all the substrings that appear in the strings $\{s_{i_1}, s_{i_2}, \ldots, s_{i_n}\}$. We will denote this set of substrings with $\mathcal{S}_{\{i_1,i_2,\ldots,i_n\}}$ and its cardinality with $|\mathcal{S}_{\{i_1,i_2,\ldots,i_n\}}|$. Note that $\mathcal{S}_{\{i_1,i_2,\ldots,i_n\}} = \bigcup_{j=1}^{n} \mathcal{S}_{i_j}$.

Example 1: Consider three individuals i_1, i_2, i_3, whose genomes are the strings $s_{i_1} = aba$, $s_{i_2} = abbc$, $s_{i_3} = babc$. We have:

- $\mathcal{S}_{i_1} = \{a, ab, aba, b, ba\}$, $|\mathcal{S}_{i_1}| = 5$
- $\mathcal{S}_{i_2} = \{a, ab, abb, abbc, b, bb, bbc, bc, c\}$, $|\mathcal{S}_{i_2}| = 9$
- $\mathcal{S}_{i_3} = \{b, ba, bab, babc, a, ab, abc, bc, c\}$, $|\mathcal{S}_{i_3}| = 9$
- $\mathcal{S}_{\{i_1,i_2,i_3\}} = \{a, ab, aba, b, ba, abb, abbc, bb, bbc, bc, c, bab, babc, abc\}$, $|\mathcal{S}_{\{i_1,i_2,i_3\}}| = 14$

We define the measure $D(P)$ of diversity of a population $P = \{i_1, i_2, \ldots, i_n\}$ as follows:

$$D(P) = D(\{i_1, i_2, \ldots, i_n\}) = n \frac{|\mathcal{S}_{\{i_1,i_2,\ldots,i_n\}}|}{\sum_{j=1}^{n} |\mathcal{S}_{i_j}|} \tag{A.1}$$

In words, the diversity of the population is defined as n times the ratio of the total number of substrings in the population genome (that is, considering only once those appearing in the genome of multiple individuals) to the cumulative number of substrings in the genome of the individuals considered separately.

Example 1 *(continued):* For the population constituted by the three individuals i_1, i_2, i_3 defined above, we have:

$$D(\{i_1, i_2, i_3\}) = 3\,\frac{|\mathcal{S}_{\{i_1,i_2,i_3\}}|}{|\mathcal{S}_{i_1}| + |\mathcal{S}_{i_2}| + |\mathcal{S}_{i_3}|} = 3\,\frac{14}{5+9+9} \approx 1.83$$

Let us analyze the properties of this definition by examining some particular cases:

- *Homogeneous population.* The population is constituted by n individuals having the same genome s and, consequently, the same set of substrings $\mathcal{S}$. Therefore, the set of substrings of the population coincides with the set of substrings of each individual: $\mathcal{S}_{\{i_1,i_2,\ldots,i_n\}} = \mathcal{S}_{i_j} = \mathcal{S}$. From the definition of D follows that

$$D(\{i_1, i_2, \ldots, i_n\}) = n\,\frac{|\mathcal{S}|}{\sum_{j=1}^{n} |\mathcal{S}|} = 1 \qquad \text{(A.2)}$$

  an intuitively appealing result, since the population corresponds actually to a collection of clones of a single individual.

Note that the converse of this property is also true, that is, if the measure of diversity of a population is unitary, all the individuals have necessarily the same genome. This can be proved by the following argument. If $D = 1$ then from the definition of D follows that $n\,|\mathcal{S}_{\{i_1,i_2,\ldots,i_n\}}| = n|\bigcup_{j=1}^{n} \mathcal{S}_{i_j}| = \sum_{j=1}^{n} |\mathcal{S}_{i_j}|$. Let us assume that there exists a pair of individuals such that $S_{i_j} \neq S_{i_k}$. This means that there exist at least one substring that belongs to one of these two sets but not to the other. This substring will be counted n times in $n|\bigcup_{j=1}^{n} \mathcal{S}_{i_j}|$, but less than n times in $\sum_{j=1}^{n} |\mathcal{S}_{i_j}|$. Thus the condition $n|\bigcup_{j=1}^{n} \mathcal{S}_{i_j}| = \sum_{j=1}^{n} |\mathcal{S}_{i_j}|$ could not be realized, which contradicts our assumption. This proves that all the individuals in the population must have the same set of substrings in their genome, and, consequently, that they must have the same genome.

- *Population of pairwise maximally distinct genomes.* The population is constituted by individuals whose genomes, considered by pairs, do not have any substring in common, that is, $S_j \cap S_k = \emptyset$ for $j \neq k$. This means that each substring belonging to $\mathcal{S}_{\{i_1,i_2,\ldots,i_n\}}$ belongs only to one of the sets S_j and therefore $|\mathcal{S}_{\{i_1,i_2,\ldots,i_n\}}| = \sum_{j=1}^{n} |\mathcal{S}_{i_j}|$, from which follows that $D(\{i_1, i_2, \ldots, i_n\}) = n$.

As before, the converse is also true, that is, if $D(\{i_1, i_2, \ldots, i_n\}) = n$ the individuals have, pairwise, no substrings in common. This can be proved by the following argument. If $D = n$ then from the definition of D follows that $|\mathcal{S}_{\{i_1,i_2,\ldots,i_n\}}| = |\bigcup_{j=1}^{n} \mathcal{S}_{i_j}| = \sum_{j=1}^{n} |\mathcal{S}_{i_j}|$. Let us assume that there exists a pair of individuals such that $S_{i_j} \cap S_{i_k} \neq \emptyset$. This means that there exist at least one substring that belongs to both sets. This substring will be counted only once in $|\bigcup_{j=1}^{n} \mathcal{S}_{i_j}|$, but at least twice in $\sum_{j=1}^{n} |\mathcal{S}_{i_j}|$. Thus the condition $|\bigcup_{j=1}^{n} \mathcal{S}_{i_j}| = \sum_{j=1}^{n} |\mathcal{S}_{i_j}|$ could not be realized, which contradicts our assumption.

With analogous deductions, it can be proved that the values of diversity obtained in these two cases constitute actually a bound for $D(P)$, that is, that we always have $1 \leq D(P) \leq n$, where n is the size of the population. This fact, along with the interpretation of Equation A.1 in terms of average number of substrings that will be presented shortly, suggests the interpretation of the value of $D(P)$ as the number of *equivalent individuals* of the population. For example, the three individuals of Example 1 above, correspond to about 1.83 equivalent individuals, a population of clones of a single individual corresponds to 1 equivalent individual., and a population of n individuals that have, pairwise, no genetic motifs in common, corresponds to n equivalent individuals.

- *Population with two kinds of genomes.* As a final example, consider a population constituted by a fraction α of its n individuals having the genome s', and the remaining fraction $(1 - \alpha)$ of individuals having a genome s'' that has no substrings in common with s'. We obtain

$$D(P) = \frac{|\mathcal{S}'| + |\mathcal{S}''|}{\alpha|\mathcal{S}'| + (1-\alpha)|\mathcal{S}''|} \tag{A.3}$$

  If $\alpha = 0.5$, that is, if the population is equally divided into individuals of type s' and individuals of type s'',we have $D(P) = 2$ independently from the values of $|\mathcal{S}'|$ and $|\mathcal{S}''|$. The same is approximately true when $|\mathcal{S}'| \approx |\mathcal{S}''|$ and α varies. If, on the other hand, we have $|\mathcal{S}'| \gg |\mathcal{S}''|$ or $|\mathcal{S}'| \ll |\mathcal{S}''|$, we can obtain almost any value of $D(P)$ in the range $(1, n)$.

This last example reveals the limitations of the measure of diversity defined above, and provides further insight into its operation. We can rewrite Equation A.1 as follows

$$D(P) = \frac{|\mathcal{S}_{\{i_1,i_2,\ldots,i_n\}}|}{\frac{1}{n}\sum_{j=1}^{n} |\mathcal{S}_{i_j}|} = \frac{|\mathcal{S}_{\{i_1,i_2,\ldots,i_n\}}|}{\overline{|\mathcal{S}_i|}} \tag{A.4}$$

This shows that the measure of diversity compares the total number of different substrings in the population genome to the average number of substrings $\overline{|\mathcal{S}_i|}$. Therefore, if one or a few individuals possess a number of substrings that greatly exceeds the average number of them in the population, the formula overestimates the diversity of the population, since it implicitly distributes evenly the substrings among the individuals.

A.2.1 Linguistic complexity

The diversity measure defined above is loosely related to the concept of *linguistic complexity* for a string defined on a given alphabet A. The linguistic complexity of a string s is defined as the ratio of the number of substrings of s, to the maximum number of substrings that can be obtained from a string of the same length on the same alphabet (Trifonov, 1990; Troyanskaya et al., 2002). In the spirit of our definition of population diversity given by Equation A.1, we can generalize the concept of linguistic complexity from single strings to populations, as follows

$$LC(P) = LC(\{i_1, i_2, \ldots, i_n\}) = \frac{|\mathcal{S}_{\{i_1,i_2,\ldots,i_n\}}|}{\max\limits_{P'_A \sim P} |\mathcal{S}_{\{i'_1,i'_2,\ldots,i'_n\}}|} \tag{A.5}$$

where $\max_{P'_A \sim P} |\mathcal{S}_{\{i'_1,i'_2,\ldots,i'_n\}}|$ is the maximum number of substrings that can be obtained with a population P'_A built on the same alphabet A of P, and having the same number of individuals and with the same length.

The value of the linguistic complexity $LC(P)$ complements the information constituted by the value of the diversity $D(P)$. $LC(P)$ gives a measure of how well the population realizes the potential of motif existence constituted by the kind and number of its individuals. In other words, it gives an idea of how effectively the population is exploring the genome space, relatively to what can be done with the same number of individuals, with the same genome lengths, and on the same alphabet. Thus, it is a relative measure of diversity, whereas $D(P)$ – which estimates the number of different individuals that the population contains – looks more like an absolute one. For example, a population P of a thousand random binary strings of length two will contain almost certainly all the six possible substrings of length one and two, and will therefore result in a value of $LC(P) = 1$ that testifies the fulfillment of the population potential. On the other hand, the six possible substrings, given the expected value of two and a half substrings for binary

string of length two, will result in $D(P) \approx 2.4$, which suggests that only a handful of different individuals exist in the population but does not inform us about how many could be housed by a population having the same structure.

A.3 Distance between individuals

The diversity of a population is closely connected to the concept of distance between individuals. For example, a measure of diversity for a population can be obtained summing all pairwise distances between its individuals. Moreover, the distance between individuals can be used to define the distance between populations (Wineberg and Oppacher, 2003a). Hence, it is worth considering the possibility of using the substring count approach to define a distance between individuals belonging to populations with genomes of variable length. We will start by trying to derive a distance d between individuals applying our formula for population diversity (Equation A.1) to pairs of individuals $\{i_1, i_2\}$. From the inequalities given above, we know that $D(\{i_1, i_2\})$ satisfies the inequality $1 \leq D(\{i_1, i_2\}) \leq 2$, with the lower bound achieved only for identical individuals. Hence, for the expression

$$d(i_1, i_2) = D(\{i_1, i_2\}) - 1 = 2\,\frac{|\mathcal{S}_{\{i_1,i_2\}}|}{|\mathcal{S}_{i_1}| + |\mathcal{S}_{i_2}|} - 1 \tag{A.6}$$

we have $d(i_1, i_2) \geq 0$, with $d(i_1, i_2) = 0$ if and only if $i_1 = i_2$. Moreover, d is obviously symmetric in its arguments, that is $d(i_1, i_2) = d(i_2, i_1)$ for any pair of individuals. The triangle inequality $d(i_1, i_2) + d(i_2, i_3) \geq d(i_1, i_3)$, however, which would qualify d as a metric and its value as a distance between individuals and between strings, is *not* satisfied. For example, for the individuals i_1, i_2, i_3, with genomes $s_{i_1} = baaaa$, $s_{i_2} = baaaab$, and $s_{i_3} = aaaab$, we have $d(i_1, i_2) = d(i_2, i_3) \approx 0.217$, and $d(i_1, i_3) \approx 0.444$, so that $d(i_1, i_2) + d(i_2, i_3) < d(i_1, i_3)$. This makes of d a *semi-metric distance* in the space of strings. Note that in the following we will not mention explicitly the qualifier "semi-metric" for d. The distance thus defined satisfies the inequality $0 \leq d(i_1, i_2) \leq 1$.

Example 2: Consider the three individuals i_1, i_2, i_3, with genomes $s_{i_1} = abab$, $s_{i_2} = abcb$, and $s_{i_3} = cbab$. We have

$\mathcal{S}_{i_1} = \{a, ab, aba, abab, b, ba, bab\}$, $|\mathcal{S}_{i_1}| = 7$

$\mathcal{S}_{i_2} = \{a, ab, abc, abcb, b, bc, bcb, c, cb\}$, $|\mathcal{S}_{i_2}| = 9$

$\mathcal{S}_{i_3} = \{c, cb, cba, cbab, b, ba, bab, a, ab\}$, $|\mathcal{S}_{i_3}| = 9$

$\mathcal{S}_{\{i_1,i_2\}} = \{a, ab, aba, abab, b, ba, bab, abc, abcb, bc, bcb, c, cb\}$, $|\mathcal{S}_{\{i_1,i_2\}}| = 13$

$\mathcal{S}_{\{i_1,i_3\}} = \{a, ab, aba, abab, b, ba, bab, c, cb, cba, cbab\}$, $|\mathcal{S}_{\{i_1,i_3\}}| = 11$

from which we obtain

$$d(i_1, i_2) = 2\,\frac{|\mathcal{S}_{\{i_1,i_2\}}|}{|\mathcal{S}_{i_1}| + |\mathcal{S}_{i_2}|} - 1 = 2\,\frac{13}{7+9} - 1 = 0.625$$

$$d(i_1, i_3) = 2\,\frac{|\mathcal{S}_{\{i_1,i_3\}}|}{|\mathcal{S}_{i_1}| + |\mathcal{S}_{i_3}|} - 1 = 2\,\frac{11}{7+9} - 1 = 0.375$$

This example shows a characteristic of d that appears at first disturbing. We can consider both the genome of i_2 and that of i_3 as obtained from that of i_1 with a single character substitution, and yet the distance of i_1 from i_2 is greater than that of i_1 from i_3. The reason is that the substitution that leads from the genome of i_1 to that of i_2 is located towards the center of the genome, whereas that leading from i_1 to i_3 is located at one extreme of it. This permits the first substitution to create a bigger set of motifs $\mathcal{S}_{\{i_1,i_2\}}$ relatively to $\mathcal{S}_{\{i_1,i_3\}}$, and this is reflected in the difference of distances. As we observed in the introduction, to define our diversity measure we do not assume any knowledge of the structure of the genomes. Hence, the number of potential motifs produced by a substitution can depend on its position in the genome. If we knew where the genes or the motifs boundaries are located, we could exclude from the count of substrings those crossing those boundaries, and obtain a new definition of diversity and distance still based on the count of substrings but now taking into account our knowledge of the genome structure. In that case, the phenomenon illustrated by Example 2 would not appear. The definition of d, on the other hand, does not suffer the problem of "extraordinary genomes" mentioned in the previous section. If two individuals i_1 and i_2 are such that $|\mathcal{S}_{i_1}| \gg |\mathcal{S}_{i_2}|$, we obtain simply $d(i_1, i_2) \approx 1$, as expected.

A.3.1 Population diversity as sum of pairwise substring distances

We can use d to define a new measure of population diversity D' based on the traditional pairwise comparison of individuals (Wineberg and

Oppacher, 2003a), as follows:

$$D'(P) = D'(\{i_1, i_2, \ldots, i_n\}) = \frac{2}{(n-1)} \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} d(i_j, i_k) \quad , \quad n > 1 \tag{A.7}$$

Note that this definition takes advantage of the fact that $d(i,i) = 0$ and of the symmetry of d, while the multiplying factor keeps the values of D' in the range $[0, n]$.

A.3.2 Tanimoto distance

We can define another distance between individuals based on the counting of substrings, using the Tanimoto measure of similarity between two generic sets X and Y (known also as *Jaccard similarity* (Levandowsky and Winter, 1971)), which is defined as (Theodoridis and Koutroumbas, 2003)

$$\sigma_t(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{A.8}$$

The similarity of two individuals can therefore be defined as

$$\sigma_t(i_1, i_2) = \frac{|\mathcal{S}_{i_1} \cap \mathcal{S}_{i_2}|}{|\mathcal{S}_{i_1} \cup \mathcal{S}_{i_2}|} = \frac{|\mathcal{S}_{i_1} \cap \mathcal{S}_{i_2}|}{|\mathcal{S}_{\{i_1, i_2\}}|} \tag{A.9}$$

from which we can derive the Tanimoto substring distance between two strings

$$d_t(i_1, i_2) = 1 - \sigma_t(i_1, i_2) = 1 - \frac{|\mathcal{S}_{i_1} \cap \mathcal{S}_{i_2}|}{|\mathcal{S}_{\{i_1, i_2\}}|} \tag{A.10}$$

This distance can be substituted to d in Equation A.7 to obtain a Tanimoto diversity measure D'_t for a population of strings

$$D'_t(P) = D'_t(\{i_1, i_2, \ldots, i_n\}) = \frac{2}{(n-1)} \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} d_t(i_j, i_k) \quad , \quad n > 1 \tag{A.11}$$

Example 2 *(continued):* For the population constituted by the three individuals i_1, i_2, i_3, with genomes $s_{i_1} = abab$, $s_{i_2} = abcb$, and $s_{i_3} = cbab$, introduced above, we have

- $\mathcal{S}_{i_1} \cap \mathcal{S}_{i_2} = \{a, ab, b\}$, $|\mathcal{S}_{i_1} \cap \mathcal{S}_{i_2}| = 3$
- $\mathcal{S}_{i_1} \cap \mathcal{S}_{i_3} = \{a, ab, b, ba, bab\}$, $|\mathcal{S}_{i_1} \cap \mathcal{S}_{i_3}| = 5$

from which we obtain

$$d_t(i_1, i_2) = 1 - \frac{|\mathcal{S}_{i_1} \cap \mathcal{S}_{i_2}|}{|\mathcal{S}_{\{i_1,i_2\}}|} = 1 - \frac{3}{13} \approx 0.77$$

$$d_t(i_1, i_3) = 1 - \frac{|\mathcal{S}_{i_1} \cap \mathcal{S}_{i_3}|}{|\mathcal{S}_{\{i_1,i_3\}}|} = 1 - \frac{5}{11} \approx 0.55$$

A.3.3 Generalized distance and diversity

To define the Tanimoto substring distance, we specialized the general definition given by Equation A.13 to the case of strings. We can go in the opposite direction with our distance d between strings defined by Equation A.6, and see it as particular case of the distance between two generic sets X and Y, defined by

$$d(X, Y) = 2\,\frac{|X \cup Y|}{|X| + |Y|} - 1 = \frac{|X \cup Y| - |X \cap Y|}{|X \cup Y| + |X \cap Y|} \tag{A.12}$$

This reveals the similarity of d with the Tanimoto distance d_t, which corresponds to

$$d_t(X, Y) = \frac{|X \cup Y| - |X \cap Y|}{|X \cup Y|} \tag{A.13}$$

Contrary to the case of d, however, d_t satisfies the triangle inequality and is therefore a metric distance (Levandowsky and Winter, 1971; Lipkus, 1999).

In an analogous way, we can interpret Equation A.1 as a particular case of the following measure of diversity for a collection (or multiset (Monro, 1987)) $\{X_1, X_2, \ldots, X_n\}$ of finite and not all empty sets X_j

$$D(\{X_1, X_2, \ldots, X_n\}) = n\,\frac{|\bigcup_{j=1}^{n} X_j|}{\sum_{j=1}^{n} |X_j|} \tag{A.14}$$

This means that D can be used to measure the diversity of a generic population of "individuals" i_j, provided there is a way to associate with each of them a set X_j which is representative of its relevant substructures and features for the application at hand. For example, remaining in the realm of strings, we could deem more meaningful for the assessment of the diversity of a population of them, the use of the set of subsequences (i.e., of characters that do not necessarily appear contiguously) instead of the set of substrings of the individual strings; in

the field of image classification, one could associate with each image a set of its subimages, and so on.

A.3.4 Tree-based representations and genetic programming

As an example of application of the generalized distances and diversity measures defined in the previous subsection to genetic representations not based on strings we can consider the field of genetic programming (GP). In GP the genome of individuals has usually the structure of a tree (Langdon and Poli, 2002). Following an approach proposed by Keijzer (1996) we can associate with each individual i_j the set X_j of all the subtrees of the tree that constitutes its genome. With this choice, Equation A.14 gives a subtree-based measure of diversity for GP populations. Similarly, interpreting X and Y as the set of subtrees of the trees that constitute the genomes of two individuals, Equation A.13 becomes a Tanimoto distance and Equation A.12 becomes a subtree distance between individuals.

A.4 Implementation issues

The definitions of diversity and distance given above are practically useful for EC runtime calculations only if there exist efficient ways to compute the number of substrings of a string and of a collection of strings.

The number of substrings of a string can be calculated efficiently building the so-called *suffix tree* of the string. This is a data structure that represent compactly the substring structure of a string and which is based on a less compact structure called *trie.* The trie associated with a string is a rooted directed tree where the edges are labeled by letters, any path down the tree spells a substring of the string, such that all paths from root to leafs are suffixes of the string and all suffixes of the string are labels of paths from the root (Crochemore and Rytter, 2002) (Figure A.1). Note that paths corresponding to suffixes do not end necessarily in a leaf. Nodes of the trie where paths corresponding to suffixes end are called *essential nodes.* The property of a trie that interests us here is the fact that the number of its edges corresponds to the number of substrings of the string (Troyanskaya et al., 2002).

The trie associated with a string can be compacted by suppressing non-branching non-essential nodes and associating with the edges of

the tree thus obtained the substrings obtained from the chain of original edges (Figure A.1). The resulting structure is called the *suffix tree* of the string (Crochemore et al., 2001; Crochemore and Rytter, 2002; Gusfield, 1997). Since the labels associated with the edges of the suffix tree correspond to substrings of the original string, they can be substituted with pointers to the substring within the string. This allows a further compaction of the suffix tree relatively to the corresponding trie (Figure A.1).

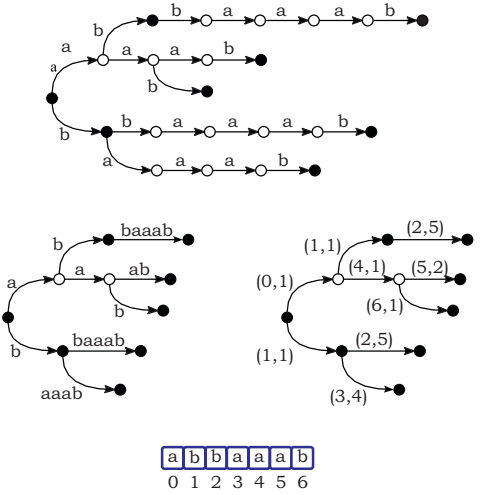


Figure A.1: *The trie and suffix tree of the string* abbaaab*, whose characters are indexed from* 0 *to* 6 *(*bottom*). The trie (*top*) has a single letter associated with each edge and the number of its edges corresponds to the number of substring of the string. Nodes represented in black are* essential nodes *and correspond to suffixes of the string, which can be obtained traversing the tree from the root (which corresponds to the empty suffix) to the node. The trie can be compacted by suppressing non-branching non-essential nodes and associating with the new edges the substrings obtained from the chain of original edges. This gives the suffix tree of the string (*center, left*). An alternative, more compact representation of the suffix tree can be obtained by substituting the substrings associated with each edge with a pair of integers* (p, l) *that gives the position of the start of the substring in the original string, and the length of the substring (*center, right*).*

Several algorithms exist that build the suffix tree of a string with a computational cost that grows linearly with the length of the string, both in terms of computation time and memory occupation. Two popular algorithms are Ukkonen's (Ukkonen, 1995) and McCreight's (McCreight, 1976): a detailed description of these algorithms including the pseudocode can be found in (Crochemore et al., 2001; Crochemore and Rytter, 2002; Gusfield, 1997). These algorithms build the suffix tree by adding successively the characters that correspond to the edges of the trie to which the suffix tree corresponds. Hence, to obtain the number of substrings of a string we must simply count the number of characters added during the construction of its suffix tree. This permits an efficient computation of the number $|\mathcal{S}_{i_j}|$of substrings of the genome of an individual of a population.

The algorithms that build the suffix tree of a single string can be modified to allow the construction of a structure representing the suffixes of a collection of strings which is called the *generalized suffix tree* (Gusfield, 1997, p. 116). The basic idea is to append to each string in the collection an end string marker not belonging to the alphabet from which the strings are formed. To build the generalized suffix tree, one starts by constructing the suffix tree of the first end-marked string using one of the algorithms listed above. Then, the second end-marked string in the collection is matched against the existing suffix tree starting from the root, until a mismatch occurs. At this point the construction of the suffix tree resumes with the first non-matched character of the string. Proceeding in this way for all the strings in the collection results in the construction of the generalized suffix tree. Counting the number of characters added during the construction one obtains the number of substrings of the strings in the collection and, in particular, the number of substrings $|\mathcal{S}_{\{i_1,i_2,\ldots,i_n\}}|$ of the genomes of a population.

A.5 Experimental results and comparisons

We now compare the measures of diversity introduced above with those currently used in EC and in other fields that make use of string comparison. In what follows we will denote with n the size of the population, with l the length of the genomes of the individuals, and with A the alphabet over which the genomes are defined.

A.5.1 Computational cost

We consider the following kinds of diversity measures for populations of n individuals

- *Leung-Gao-Xu diversity*. Leung, Gao, and Xu (Leung et al., 1997) introduced a measure of diversity $D_\lambda(P)$ for populations whose individual genomes are binary strings s_j of fixed length l. $D_\lambda(P)$ is defined as the number of components of the string of integers $\sum_{j=1}^{n} s_j$ (with the sum performed componentwise in $\mathbb{N}$) whose values are not equal to 0 and n. The time computational complexity of the direct implementation of this definition is $O(l \cdot n)$.

- *Moment of inertia diversity, pairwise Hamming diversity*, and *entropic diversity*: We group under a unique heading three diversity measures that are slight variations on the same theme (Morrison and De Jong, 2002; Wineberg and Oppacher, 2003b). In the experiments that will follow we will report the results only for the moment of inertia measure, since the other two require almost the same computation time and have a value that differs from the moment of inertia only by a scaling factor and, possibly, by terms of second order and higher in the character frequencies (Wineberg and Oppacher, 2003b).

  - The *moment of inertia diversity* for populations of binary strings of fixed length l is defined by

$$D_m(P) = \sum_{i=1}^{l} \sum_{j=1}^{n} (s_{ij} - c_i)^2 \tag{A.15}$$

    where s_{ij} is the character in position i of the j-th string, c_i is the i-th coordinate of the centroid

$$c_i = \frac{1}{n} \sum_{j=1}^{n} s_{ij} \tag{A.16}$$

    and the operations are performed in $\mathbb{R}$. The time complexity of the direct implementation of Equation A.15 is $O(l \cdot n)$ (Morrison and De Jong, 2002).

– The *pairwise Hamming diversity* is defined as

$$D_h(P) = \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} d_h(i_j, i_k) \tag{A.17}$$

where $d_h(i_j, i_k)$ is the Hamming distance between the individual genomes. $D_h(P)$ is defined for populations with genomes of fixed length l over an arbitrary finite alphabet. The time complexity of the naive implementation of Equation A.17 is $O(l \cdot n^2)$ but there exist implementations of this measure that reduce the time complexity to $O(l \cdot n)$ (Morrison and De Jong, 2002; Wineberg and Oppacher, 2003b). For the case of binary genomes it can be shown (Morrison and De Jong, 2002) that $D_h(P) = n\, D_m(P)$. Hence, $D_h(P)$ differs from $D_m(P)$ only by a scaling factor and can be implemented with the same complexity. For arbitrary genomes, $D_h(P)$ can be calculated with $O(l \cdot n)$ complexity using the following expression (Wineberg and Oppacher, 2003b)

$$D_h(P) = \frac{n^2}{2l} \sum_{k=1}^{l} \sum_{\alpha \in A} f_k(\alpha)(1 - f_k(\alpha)) \tag{A.18}$$

where $f_k(\alpha)$ is the frequency of the character α at the position k in the population genomes.

– The *entropic diversity* is defined as

$$D_e(P) = -\frac{1}{l} \sum_{k=1}^{l} \sum_{\alpha \in A} f_k(\alpha) \log f_k(\alpha) \tag{A.19}$$

where $f_k(\alpha)$ is the frequency of the character α at the position k in the population genomes. The first term of the Taylor expansion of $D_e(P)$ is proportional to the pairwise hamming distance $D_h(P)$, which is therefore a good approximation. $D_e(P)$ can be implemented with time complexity $O(l \cdot n)$ (Wineberg and Oppacher, 2003b).

✓ *Substring diversity*. It is defined by Equation A.1. Using suffix trees it can be implemented with time complexity $O(l \cdot n)$.

✓ *Pairwise substring diversity* and *pairwise Tanimoto diversity*. They are defined by Equation A.7 and Equation A.11, respectively. Us-

ing suffix trees to compute the number of substrings, and then applying the definitions directly, the time complexity of the implementation is $O(l \cdot n^2)$.

To give an idea of the actual computation time of a typical implementation on a present day personal computer, we have plotted in Figure A.2 the computation time for these measures of diversity as a function of genome length and population size for a randomly generated population of fixed genome length over a binary alphabet. The substring diversity measure is implemented with McCreight suffix tree algorithm (McCreight, 1976). As anticipated, the results for the pairwise Hamming diversity and for the entropic diversity are not shown, since they are almost indistinguishable from the values obtained for the moment of inertia diversity.

The curves of Figure A.2 confirm the predicted time complexities and show that the substring diversity measure $D(P)$, although more computationally expensive than existing diversity measures for fixed length genomes, is sufficiently inexpensive to be usable for runtime diversity measurements on present day computers. Note that the pairwise substring and pairwise Tanimoto curves are almost coincident, and that their $O(l \cdot n^2)$ complexity makes the direct implementation of these measures rapidly impractical for runtime diversity assessment when the population size grows.

A.5.2 Fixed genome length

To give an example of computation of the various kinds of diversities in an actual EC setting, we performed a series of runs of a genetic algorithm on a function optimization problem. The genomes considered in this section have fixed length. Hence, both the conventional non-string-based and the string-based diversity measures can be used. The goal is to show that in this simple setting – the only one where the comparison can be performed – the string-based measures give results comparable to those of the conventional measures.

The function to be optimized was the two-dimensional sine envelope sine wave function (Leung et al., 1997) defined by

$$f(x_1, x_2) = 0.5 - \frac{\sin \sqrt{x_1^2 + x_2^2} - 0.5}{(1 - 0.001(x_1^2 + x_2^2))^2} \tag{A.20}$$

The optimization was performed in the domain $[-100, 100] \times [-100, 100]$,

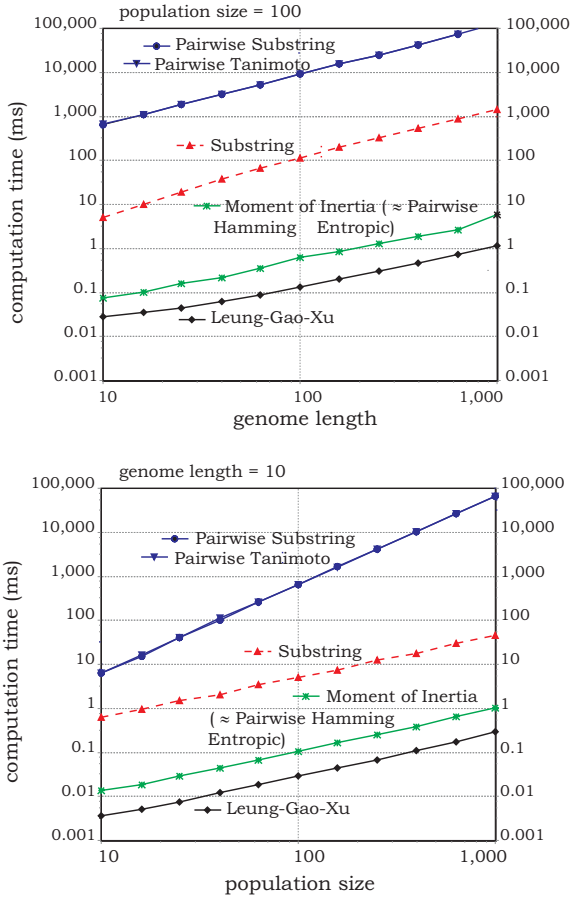


Figure A.2: *Computation time vs. genome length and population size of different measures of diversity for the implementations described in the text and run on a PC with Pentium III microprocessor clocked at 850MHz.*

where the unique global maximum is $f(0,0) = 1$. Each parameter x_i was binary encoded by a string of length 22, so that $l = 44$. The population size was $n = 50$. The algorithm used tournament selection with tournament size $t = 2$, with mutation probability $p_m = 0.01$ and crossover

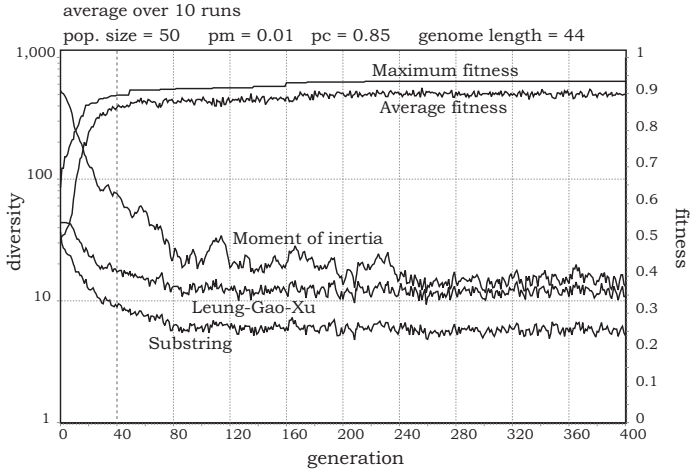


Figure A.3: *The various population diversity measures computed on the population evolved with a genetic algorithm applied to a function optimization experiment.c*

probability $p_c = 0.85$. Figure A.3 reports the results average over ten runs, starting with randomly generated populations.

Figure A.3 does not show the pairwise Hamming diversity, since it differs from the moment of inertia diversity only by a scale factor, nor the entropic diversity, since, after suitable scaling, it is almost indistinguishable from the moment of inertia diversity. The values of the pairwise substring and pairwise Tanimoto diversities are close to those of the Leung-Gao-Xu diversity. To avoid an excessive overlapping of curves, all the substring-based diversities are represented in Figure A.4 for the same set of runs of Figure A.3. In this case, the substring diversities for the genes that represent each parameter were also separately computed as *substring columnwise diversity*. This was done to show that the substring approach can be applied to evaluate the diversity of each genome substructure when these are known. Note that the substring diversity calculated on the whole genome is not the sum of the two substring columnwise diversities, which is instead the case for most kinds of diversity measures currently used, since they

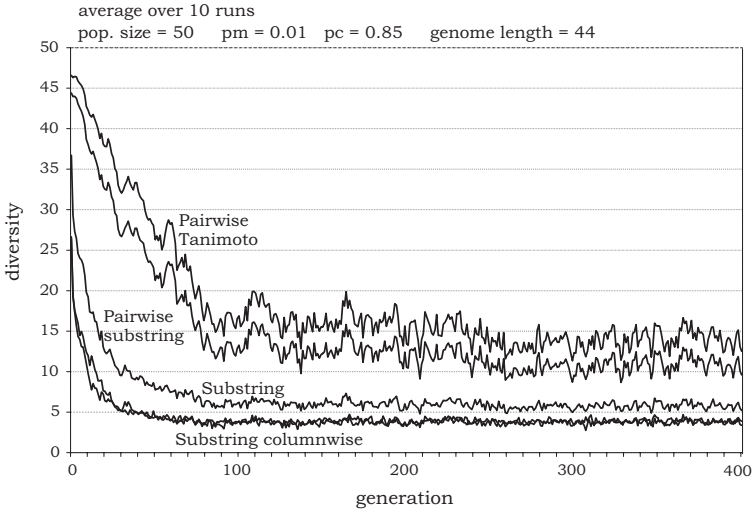


Figure A.4: *The substring-based population diversity measures computed for the evolutionary runs of Figure A.3. The substring columnwise curves refer to the genomic diversity computed for each parameter separately.*

obtain the global diversity summing the contribution of each locus in the genome string.

Figures A.3 and A.4 show that in this conventional setting constituted by populations with fixed genome length, the measures of diversity based on the substring approach behave very much like the familiar diversity measures based on the Hamming distance, or based on the count of the converged bits like the Leung-Gao-Xu measure. In other words, when applied to fixed genome length populations the substring-based measures conform to the behavior expected from a conventional measure of diversity.

A.5.3 Variable genome length

So far we have considered only examples of population diversity calculations made on fixed genome length populations. This was necessary to allow the comparison with conventional diversity measures, which

apply only to the fixed genome length case. However, the measures of diversity introduced here find their justification mainly in the case of populations with variable genome structure, where the approaches based on the Hamming distance cannot be used. To illustrate the applicability of the substring approach to the variable genome length - which is just the first step towards a truly variable structure, to which the substring approach still applies - we modified the encoding of the function optimization experiment described above to allow the variable length encoding of each parameter. More precisely, the parameters x_i were encoded by a binary string whose length was allowed to be different for the two parameters of an individual, and to vary from individual to individual. We started the evolutions with randomly generated populations where each parameter was represented by a gene with length randomly chosen between 2 and 20 bits. To allow the variation of the genome length of individuals during the evolution, the mutation operator was extended to include character insertion and deletion with probabilities $p_i = p_d = 0.001$, in addition to substitution with probability $p_m = 0.01$. Individuals that, following a mutation, were found to have an empty gene were simply removed from the population.

Figure A.5 shows the result of a single evolutionary run of function optimization with variable length genome. Note that the substring-based measures of population diversity introduced above apply unaltered to the case of variable genome length, whereas the traditional population diversity measures cannot be applied to this case.

A.5.4 Highly reorganizable genome

In the previous examples all individuals had a genome with the same structure, that is, the number, meaning, and order of the genes was predefined and at most the length of each gene could vary. We consider now the application of the measure of diversity defined in this appendix to the genome introduced in this thesis for the representation and evolution of analog networks. We consider in particular an example of evolution of an artificial neural network. Figure A.6 reminds briefly the elements involved in the representation and shows an example of the resulting genome. Since, as was abundantly emphasised in the previous chapters, the genome can be subjected to the action of many genetic operators, a given structure can appear in any position in the genome and can be present in several instances as more or less diverging copies. We are therefore in presence of the kind of highly reorga-

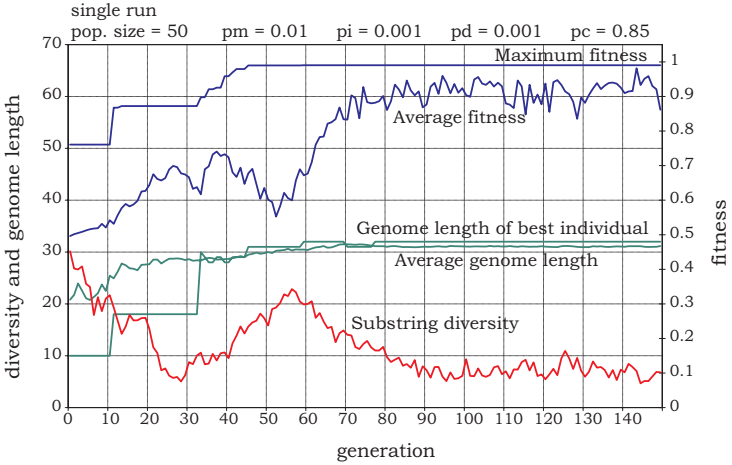


Figure A.5: *The substring diversity measure computed on the population evolved with a genetic algorithm applied to a function optimization experiment with variable length genome. From generations 30 to 60 there is probably an episode of migration of the population from two distinct regions of the genetic space; this is accompanied by a decrease in average fitness and is reflected in an increase of diversity of the population. Note that in this experiment the length of the genome of distinct individuals of the population can be different and, therefore, the Leung-Gao-Xu, pairwise Hamming, moment of inertia, and entropic diversity measures cannot be used.*

nizable genome whose relevance for artificial evolution was discussed in the introduction of this appendix and for which the substring-based diversity and distance measures come to full fruition.

Figure A.7 reports the result of a single evolutionary run aimed at the synthesis of a neural network solving the XOR problem (Haykin, 1999, p. 175). The setup of the system provides two predefined input neurons and an output neuron. The neurons decoded from the genome are thus inserted as hidden neurons (Haykin, 1999). An exact solution is characterized by a fitness value of zero, and is first found at about generation 120. It is known (Haykin, 1999) that the simplest network that can solve the problem has a single (hidden) neuron decoded from the genome. The curve of the average number of decoded neurons

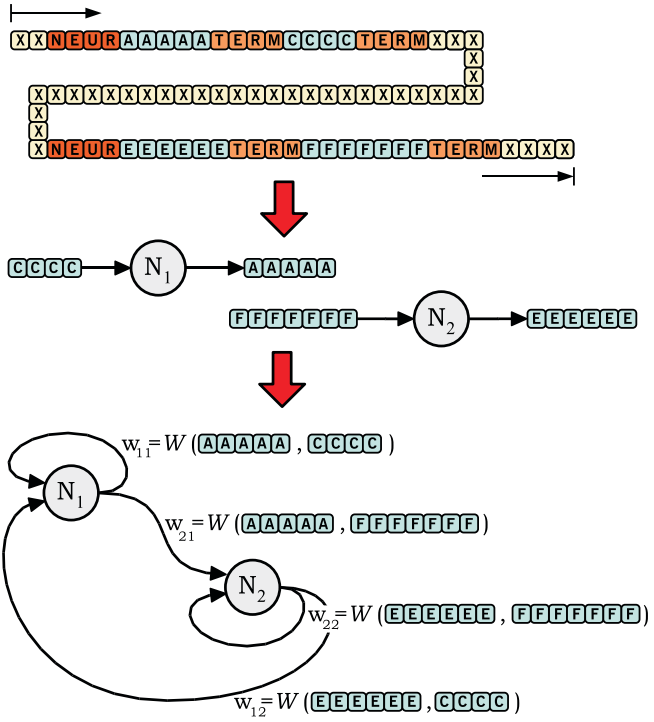


Figure A.6: *A fragment of genome* (top)*, the devices decoded from it* (center)*, and an example of weighted connection established between the devices* (right, bottom)*. A set of predefined tokens ("*NEUR*", "*TERM*", . . .) identifies the regions coding for devices and delimit the strings that will be associated with the terminals by the decoding process. The weight of the connections between the output and the inputs of all neurons is obtained by applying a function f that maps pairs of strings associated with the terminals, to numeric values.*

testifies that the evolutionary process produces networks more complex than this while proceeding towards the exact solution. The same curve shows that these networks are subsequently simplified by the evolutionary process until, at about generation 300, the population is composed almost exclusively by individuals whose genome encodes a

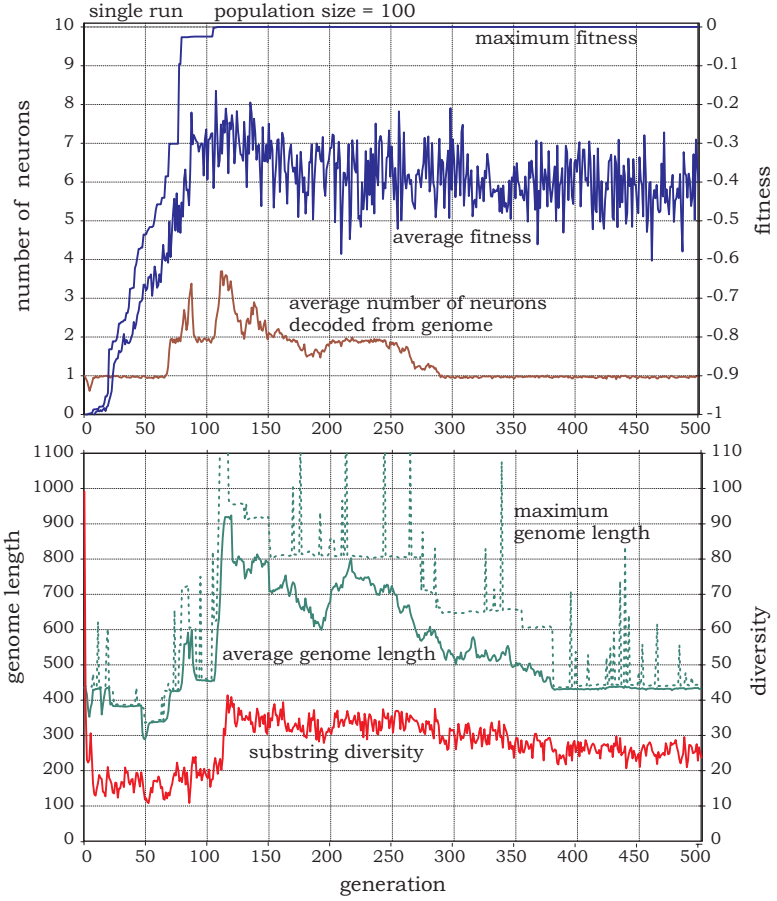


Figure A.7: *An evolutionary run aimed at the synthesis of a neural network for the XOR problem. The meaning of the curves is detailed in the text.*

single neuron. This is due to the presence of a "dead zone" around the required output values, which eliminates any selective pressure among networks that can produce outputs within the dead zone and, in particular, does not reward networks that "associate" many neurons just

to better approximate the exact output values. The curves reporting the average and maximum genome length during the evolution testify the presence of several episodes of duplication and deletion of genome fragments. The duplication episodes in particular appear instrumental to the first attainment of an exact solution. The curve of the population substring diversity shows that the diversity remains low before an exact solution is found, that is, while the selective pressure is high. Successively, the population diversity assumes consistently higher values relatively to the pre-solution phase, irrespective of the value of genome length. This observation supports the hypothesis of the existence of several alternative solutions. Moreover, it indicates that the substring diversity measure is able to capture the essential dynamics of the population diversity. Note that all the non-substring-based diversity measures considered above are defined only for fixed genome lengths and cannot be applied to this kind of variable-length and variable-structure genome.

A.5.5 Nucleotide diversity and substring diversity

In molecular and population genetics the measurement of the polymorphism of a population is based on the *nucleotide diversity* measure Π, defined on a DNA sequence by

$$\Pi = \sum_j \sum_k x_j x_k \pi_{jk} \tag{A.21}$$

where the sum is performed on all the kinds of different sequences, x_j and x_k are the frequencies of the j-th and k-th type of sequences, and π_{jk} is the proportion of different nucleotides between the two types of sequences (Graur and Li, 2000). It follows from this definition that Π corresponds to

$$\Pi = \frac{1}{l\, n^2} \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} d_h(i_j, i_k) \tag{A.22}$$

that is, it is a normalized pairwise Hamming diversity measure (Wineberg and Oppacher, 2003b) which gives the average number of nucleotide differences per site (Graur and Li, 2000).

Figure A.8 shows two examples of calculation of Π for a population of four DNA sequences. Note that the value of Π in the two cases is the same, since the number of nucleotide differences is the same, even if the differences are spread among more individuals in the second case.

Conversely, the value of D is different in the two cases, since D is sensitive to the number of DNA sequences affected by the differences. Hence, the information provided by the substring diversity measure D can complement that conveyed by Π in assessing population polymorphism.

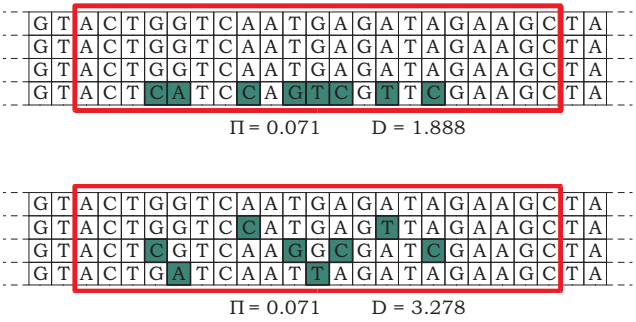


Figure A.8: *The nucleotide diversity measure Π is used in molecular genetics to assess the polymorphism of a population. It measures the average number of nucleotide differences per site in DNA sequences of the population. As such, Π is not affected by the distribution of the differences among the individuals. On the contrary, the substring diversity measure D is sensitive to this difference, and can therefore complement Π in assessing the polymorphism of the population.*

A.6 Conclusion

There is a growing interest in the field of evolutionary computation in the definition of genomes that can be subject to major reorganizations – such as insertions, deletions duplications and transpositions – and still remain decodable. The reason for this interest is the hope to foster in this way the exploration of the search space, the emergence of modularity, the reuse of evolved substructures, and, eventually, the open-endedness of the evolutionary process. At the same time, the effort to improve the performance of evolutionary algorithms results in a tendency towards the integration in the algorithms of a certain degree of control of the population diversity. There follows a need for measures

of diversity that apply to populations with genomes of variable length, and, more generally, to populations with highly reorganizable genomes.

To fulfill this need, we have defined measures of diversity and distances between individuals, that apply to populations and individuals whose genome is constituted by finite strings of variable length on finite alphabets. The definitions are based on the counting of substrings of the population genomes, considered first separately and then collectively. The motivation behind the substring counting approach is the possibility to estimate in this way the potential genomic motifs contained in the genomes. For example, there is the possibility to recognize the similarity of individuals whose genomes are constituted by the same motifs differently arranged. The measures thus obtained do not require any detailed knowledge of the structure of the genome and, therefore, apply to generic genomes. However, information about the genome structure can be taken into account when available by applying the counting procedures to the substructures present in the genomes. The measures introduced are based on properties of the genomes that can be computed in linear space and time, thus making them suitable for runtime application during an evolutionary process. Moreover, these measures and their generalizations can be used for the assessment of the diversity of other kinds of populations, such as tree-based genetic programming populations, biological sequences, and generic collections of sets.

Appendix B

Evolutionary quantization

Overview

This appendix is devoted to the issue of the representation of real-valued parameters in evolutionary algorithms. The issue is formulated as a problem of quantization of real variables. First, a cost of the quantization is defined in general terms, considering the effect of the elementary acts of quantization. Then, actual expressions for the cost are proposed and their relevance to the encoding of real-valued evolutionary parameters is discussed. Some classical results from the communication engineering literature on quantization are reported, and new optimality results are derived for some error functions, probability distributions of the quantized variables, and expressions of quantization cost that are especially relevant to the evolutionary encoding of real variables in general, and to the genetic encoding of the interaction strength of analog networks in particular. The analysis is focused on the characteristics of uniform, logarithmic, and floating-point quantizers in relation to the dynamic range, invariance properties, and arithmetic role of the quantized variables.

B.1 Introduction

The implementation of an algorithm on a digital computer implies the use of a finite representation for the parameters and variables of the algorithm. When the objects to be represented are a range of real numbers, the choice of the representation corresponds to the selection of a finite subset of a continuum of real numbers. Numbers which are not in the selected subset must be approximated by numbers in the subset, and a representation error ensues.

The scenario just described is well known in the field of communication engineering, where it is referred to as the process of *quantization* (Gray and Neuhoff, 1998; Jayant and Noll, 1984). In the communication perspective, a signal source generates a sequence of real numbers x belonging to an interval $\mathcal{X} = [x_{min}, x_{max}]$, and the quantization process transforms each element x of the sequence into a *quantized value* $x_i = q(x)$ belonging to a finite set of quantized values[1] $\mathcal{Q} = \{x_i;\, i \in \mathcal{I}\}$, with $\mathcal{I} = \{1, \ldots, N\}$, $N > 1$, and[2] $x_1 = x_{min}$, $x_N = x_{max}$ (Figure B.1).

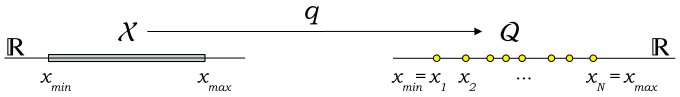


Figure B.1: *In the classical quantization scenario, a real interval $\mathcal{X} = [x_{min}, x_{max}]$ is mapped into a set $\mathcal{Q} = \{x_i\} \subset \mathbb{R}$ of quantized values by a quantizer function $q(x)$.*

The association of x_i with x is based on the partition of the interval $\mathcal{X}$ into a collection of subsets called *cells* $\mathcal{C} = \{C_i;\, i \in \mathcal{I}\}$, with the condition that $x_i = q(x)$ for $x \in C_i$. If we assume N as given, the quantizer

[1] More generally, one can define $\mathcal{Q}$ as a countable set. In the actual cases that interests us here, however, $\mathcal{Q}$ is always finite.

[2] The conditions $x_1 = x_{min}$, and $x_N = x_{max}$ are not strictly necessary for the definition of a quantizer, and correspond to the additional requirement that the boundaries x_{min} and x_{max} of the range of reals that must be quantized are values that must be represented exactly and, therefore, must coincide with two quantized values. This is the case of the quantization of resistance values in the example of analog electronic circuit design described in the text, where the boundary values 1Ω and $10\mathrm{M}\Omega$ of the resistance range are assumed as represented exactly, and the representation error outside this range is of no concern.

optimization task of the communication engineer consists in defining the quantizer – that is, choosing the quantized values x_i and the cells C_i – so as to minimize a suitably defined quantization cost.

In an evolutionary perspective, there is no explicit input variable x but a genetic encoding of real-valued phenotypic parameters. The finite size of the genome limits the values of phenotypic parameters that can be accessed by the evolutionary process to a given finite set $\mathcal{Q} = \{x_i;\, i \in \mathcal{I}\}$. For example, in the case of the evolutionary system for analog networks described in this thesis, given a maximum length for the sequences of characters extracted from the genome and associated with the terminals of the network devices, the sequence interaction map produces a finite set $\mathcal{G}$ composed of sequence interaction values. Without loss of generality we can think of this set as mapped onto the finite sequence of integers $\mathcal{I} = \{1, \ldots, N\}$. The network-specific interaction map associates a value $x_i \in \mathcal{Q}$ with each element $i \in \mathcal{I}$ (Figure B.2).

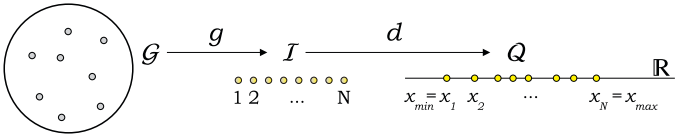


Figure B.2: *In an evolutionary scenario, the quantization is not explicit but corresponds to the implicit constraint imposed on the values of the phenotypic parameters by the use of a mapping of a finite set $\mathcal{G}$ of genetically encoded elements, onto a set $\mathcal{Q} \subset \mathbb{R}$ of quantized values indexed by the integers of a set $\mathcal{I} = \{1, \ldots, N\}$. The map $d : \mathcal{I} \to \mathcal{Q}$ is called the decoder of the quantizer.*

In a more traditional evolutionary computation scenario, the value of a real parameter is encoded into a finite fragment of the genome which can represent N distinct parameter values. We can imagine the N distinct configurations of the fragment of genome coding for the parameter as forming a finite set $\mathcal{G}$. This set is mapped onto the finite sequence of integers $\mathcal{I}$, which is further bijectively mapped onto the set of quantized values $\mathcal{Q}$. Although in these evolutionary scenarios – contrary to the communication engineering case described above – no explicit unquantized variable x and, therefore, no explicit association $x \to x_i$ of a variable x with a quantized value x_i is given, we can still think of the

evolutionary process as "wishing" to employ a real value x and being forced to approximate it with one of the quantized values $x_i \in \mathcal{Q}$. In other words, the evolutionary process can be thought of as a source of real numbers x that must be quantized. This permits us to think in terms of an *evolutionary quantization* problem, and to adopt the quantizer optimization perspective of communication engineering.

Note that the *scalar quantization* scenario just described is clearly a simplification, since the evolutionary process evaluates in reality the fitness of the whole phenotype. Assuming for simplicity that the phenotype can be described by a vector of real parameters, we should thus think in terms of *vector quantization*, that is, in terms of a quantization of the whole parameter space, rather than in terms of a quantization of each real parameter separately. In practice, however, we cannot predict the distribution of the vector of unquantized phenotypic parameters that would be produced by evolution and which is necessary for the determination of the optimal vector quantizer. Moreover, an actual optimization of the quantizer should be based on an estimate of the evolutionary cost of the quantization, or, at least, of its cost in terms of fitness. Apart from trivial cases, an analytic expression of this cost is either not available, or is a highly nonlinear function of the phenotypic parameters. Thus, the pursuit of an actual optimal vector evolutionary quantization appears in fact chimeric.

Another implicit characteristic of the simple quantization scenario described above is that of being *static* (or *nonadaptive*), that is, of being defined at the start of evolution and kept unchanged afterwards. It is however conceivable that a better use of the finite representation resources available could be obtained by allowing the reassignment of the quantization during the evolutionary run. This corresponds to the adoption of a *dynamic* or *adaptive* quantization (Goodman and Gersho, 1974).[3]

To simplify the implementation of the quantizer, a typical and simple solution is to disregard the possibility of defining an adaptive vector quantizer, and to define separately the quantization of each phenotypic parameter before the start of the evolutionary experiment, assuming a quantization cost that is a function of the approximation error of each phenotypic parameter seen as an independent real variable, in the hope that the minimization of this error leads to a minimization of

[3]The *Dynamic Parameter Encoding* technique introduced by Schraudolph and Belew (1992) is an example of this kind of approach, where the range of each real phenotypic parameter is resized during the run.

the evolutionary consequences of the quantization. This approach corresponds to a *static scalar* quantization, and will be implicitly assumed in the rest of this appendix. This assumption appears more acceptable if we consider that it is commonly adopted in many technological fields, where a finite set of standardized components are manufactured and put at the designer's disposal before the start of any actual design process.

B.2 The cost of quantization

Given the number N of quantized values, and the range $[x_{min}, x_{max}]$ of the input value, the *optimal (scalar) quantization problem* consists of choosing the set $\mathcal{Q}$ of quantized values and the set $\mathcal{C}$ of cells in order to minimize the cost of the effects of quantization. To set up an optimization problem we define an error function $\varepsilon(x, x_i)$ that specifies the effect of the single act of quantization, and derive from it an estimate $\mathcal{E}(\mathcal{Q}, \mathcal{C})$ of the cost of the whole quantization strategy, taking into account the probability of the different acts of quantization, and their relevance.

In communication problems, a common choice for $\varepsilon(x, x_i)$ is the squared error $(x - x_i)^2$, with $\mathcal{E}(\mathcal{Q}, \mathcal{C})$ corresponding to the expected mean squared error

$$\mathcal{E}(\mathcal{Q}, \mathcal{C}) = \sum_{i=1}^{N} \int_{C_i} \varepsilon(x, x_i)\, p(x)\, dx = \sum_{i=1}^{N} \int_{C_i} (x - x_i)^2\, p(x)\, dx \qquad \text{(B.1)}$$

where $p(x)$ is the probability density function of the variable x that must be quantized. The use of the squared error as a measure of cost is justified by the fact that, in communicating an analog signal, the term $(x - x_i)$ can be considered noise, and the expected cost represented by Equation B.1 is proportional to the average energy or power of the quantization noise (Gray and Neuhoff, 1998).

From an evolutionary analog network point of view, the mean squared error choice for the cost function does not appear as the only reasonable one. To understand why, it is useful make a short detour in the field of analog electronic design. Analog electronic design can be seen as the art of selecting and connecting a collection of components such as transistors, capacitors, batteries, and still many others, via electric resistors, in order to obtain an electronic circuit realizing a given functionality. When working with discrete components (as opposed to designing analog integrated circuits), the designer has only access to a

finite number of standardized electric resistance values. For example, a common standardized series of resistors is the so-called E12 series, which has twelve resistance values for each decade of resistance value, corresponding to the base value of the decade (a power of 10) multiplied by a coefficient in the set

$$\{1.0,\ 1.2,\ 1.5,\ 1.8,\ 2.2,\ 2.7,\ 3.3,\ 3.9,\ 4.7,\ 5.6,\ 6.8,\ 8.2\} \tag{B.2}$$

The range of actual values goes typically from 1Ω to $10\mathrm{M}\Omega$. This means that, for example, the available resistors include the $1\,\Omega$, $1.2\,\Omega$, and $1.5\,\Omega$ resistance values, but also the $10\,\Omega$, $12\,\Omega$, and $15\,\Omega$ values, and so on, up to $10\mathrm{M}\Omega$.

In designing a circuit, the engineer computes real values of resistance which typically do not correspond to the values of the available standard series. Hence, a further design step consists of substituting the theoretical values with standard resistance values.[4] The example of the quantized values of the E12 series shows that what concerns the electronic designer in this quantization step is neither the absolute error $|x - x_i|$ nor any of its powers such as the squared error $(x - x_i)^2$ or the generic p-th power $|x - x_i|^p$, but, rather, the *relative error* $|x - x_i|/x$. For example, a theoretical value of $12.7\,\Omega$ could be quantized to $12\,\Omega$, and a theoretical value of $127\,\Omega$ could be quantized to $120\,\Omega$, committing a substantially different absolute and squared error, but the same relative error. One of the reasons behind the relevance of the relative error is the fact that a resistor is typically used to convert a value of voltage into a value of current, or vice versa. This conversion implies the multiplication or division of a quantity by the resistance value, and it is well known that for these operations, under rather general conditions for the distribution of the error, it is the relative error of the factors – in the form of squared relative error – that is relevant in the determination of the error of the result (Gillespie, 1983; Sivia, 1996; Taylor, 1996; Silverman et al., 2004; Coutinho et al., 2004). This multiplicative perspective is even more relevant if we consider that to minimize the effect

[4] Usually the designer does not quantize the values of the resistors appearing in the circuits separately, but, rather, identifies subsets of resistors that contribute to the realization of a circuit subfunction and quantizes simultaneously the whole subset. Nonetheless, as explained above, the choice of the static scalar quantization perspective in the optimization of the quantizer is the only realistic one, and is actually implied by the very definition of the sets of standardized resistance values. Note that for reasons of cost and reliability of the resulting circuit, the use of combinations of standard resistance values to obtain a non-standard value is almost never considered in practice.

of external perturbations acting on the whole circuit, the critical performances of a well-designed circuit are typically made to depend on the ratio of resistance values appearing in the circuit, rather than on their absolute value. Note that this does not mean that only the relative error or its powers must be considered relevant in an evolutionary context. We must not forget that when operations of sum and subtraction are performed on real values, it is instead the absolute error of the factors – in the form of squared absolute error – that is relevant in the determination of the error of the result (Sivia, 1996; Taylor, 1996). A later section of this appendix examines these issues – in particular the multiplication and division vs. sum and subtraction aspect – in more depth.

We can expected to find the prominence of the relative error in all cases where operations of multiplication or division are systematically performed on real values. One obvious example is the representation of real numbers used in computer implementations of algorithms implying multiplications and divisions. In this case, if the computer is digital, we have a finite set $\mathcal{B}$ of binary strings that is mapped onto the set $\mathcal{Q}$ of quantized values represented in the computer. In an analog network perspective, another important case where the relative error plays a crucial role is that of the representation of the input weights of the neurons of artificial neural networks, since the weights are multiplied by the amplitude of the incoming signals in the equations defining the behavior of the artificial neurons (Haykin, 1999). We can extend these considerations applying to analog electronic circuit and artificial neural network design and evolution to the general case of analog network design and evolution. We have defined an analog network as a collection of connected devices where the links between devices are characterized by a scalar value of interaction strength. Typically, this scalar value will transform multiplicatively the amplitude of a signal and this means that the relative error is the relevant quantity in representing the interaction strength values.

In evolutionary computation, apart from the questionable preferential use of the squared error as the function $\varepsilon(x, x_i)$, there is another exceptionable aspect of Equation B.1 defining the cost $\mathcal{E}(\mathcal{Q}, \mathcal{C})$ of the quantization strategy, namely, the implicit hypothesis that the cost depends on the average approximation error over the whole range of the x variable. The limit of this assumption lies in the fact that in many design problems the resulting system is as good as the worst of its subsystems. For example, in an analog circuit design problem a badly ap-

proximated resistor value can alter the performance of a subcircuit in such a way that the performance of the whole circuit is compromised, irrespective of the quality of the other parts of the circuit. In this case, as will be shown below, the cost of the approximation does not depend on the average error, but on the maximum error. It is therefore worth to also examine the case of the optimal quantization when the cost depends on the *maximum* (relative or absolute) quantization error. In this case, there is the advantage that the detailed specification of the probability density function is not required for the optimization.

Summing up, in defining the cost of the quantization in view of its optimization in an evolutionary scenario, we must consider the role of the parameters that are being quantized, and, in particular, the kind of mathematical operations in which they are involved, and the functional consequences of the quantization error. In some cases it makes sense to consider the absolute error or its powers as the indicator of the quantization cost; in other circumstances it is better to assign this role to the relative error. Moreover, in some circumstances it makes sense to use as quantization cost the expected error, whereas in other circumstances the adoption of the maximum error as cost is preferable. In the forthcoming sections we will consider the properties of some quantization schemata in relation to these kinds of errors. Since the communication engineering literature has dealt abundantly with the quadratic error case (see for example the review paper by Gray and Neuhoff (1998), and references therein), and since the quantization scheme that is best suited for the absolute error case is already extensively used in evolutionary computation and needs no further advocacy, we will evaluate the performance and optimality of the different quantization schemata focusing on their behavior with respect to the relative quantization error.

Besides the quantization suited to the genetic encoding technique for analog networks based on the device interaction map, we will also consider quantizations used in association with more conventional genetic representations of real parameters defined by mappings which associate univocally a real value to a binary string of fixed length extracted from the genome. In particular, we will compute the number of bits required to achieve a given relative error performance, since this value has an effect on the dimension of the genome and, therefore, of the size of the genetic search space. In this perspective, we will analyze also the properties of the floating point representation of real numbers, which is another option for the direct encoding of real parameters as

fixed length binary strings.

B.3 Specifying a quantizer

We assume as given an interval $\mathcal{X} = [x_{min}, x_{max}] \subset \mathbb{R}^+$ of positive real numbers, which we want to quantize into a predefined number $N > 1$ of discrete values.[5] To this end we define a set of quantized values $\mathcal{Q} = \{x_i;\, x_i \in \mathbb{R},\, i \in \mathcal{I}\}$, with $\mathcal{I} = \{1, \ldots, N\}$, $x_{i+1} > x_i$ for all i, $x_1 = x_{min}$, $x_N = x_{max}$, and a quantizer function $q : \mathcal{X} \to \mathcal{Q}$ associating a quantized value $x_i = q(x)$ with each $x \in \mathcal{X}$. It is useful to decompose the quantizer function into two functions: the (lossy) encoder $e : \mathcal{X} \to \mathcal{I}$, which transforms the input value x into the index $i = e(x)$ of the quantized value, and the decoder[6] $d : \mathcal{I} \to \mathcal{Q}$, which is an order-preserving map that transforms the index i into the quantized value $x_i = d(i)$. The quantizer function is defined by the assignment of a collection of sets called *cells* $\mathcal{C} = \{C_i\,;\, C_i \subset \mathcal{X}\,,\, i \in \mathcal{I}\}$ that constitute a partition of $\mathcal{X}$, and by the condition that $x_i = q(x)$ for $x \in C_i$. The definition of the quantizer corresponds thus to the assignment of the set $\mathcal{Q}$ of quantized values and of the set $\mathcal{C}$ of cells.

The choice of the quantizer can be formulated as an optimization problem by defining a cost $\mathcal{E}(\mathcal{Q}, \mathcal{C})$ of the quantization. The cost function $\mathcal{E}(\mathcal{Q}, \mathcal{C})$ is defined in terms of an error function $\varepsilon(x, x_i)$, which we assume to be non negative, and to vanish only for $x = x_i$. We will examine in particular the properties of quantizers using one of the following two types of *error* function

- absolute error

$$\varepsilon_a(x, x_i) = |x - x_i| \tag{B.3}$$

[5] We assume that the interval $\mathcal{X}$ to be quantized is composed only of non negative real numbers, that is, we assume $x_{min} > 0$. This ensures that relative quantization error is always defined (on the other hand, while dealing with the absolute error, this limitation can be relaxed). The quantization of intervals including non positive real numbers is done by considering separately the positive and the negative subintervals (possibly representing separately the null value). The quantization of an interval $[-x_{max}, -x_{min}]$ such that $[x_{min}, x_{max}] \subset \mathbb{R}^+$ is obtained quantizing $[x_{min}, x_{max}]$ and then reinserting the negative sign.

[6] Encoder and decoder are terms derived from the typical role of the quantizer in a communication system. In the simplest scenario, the input variable x is quantized and then encoded into a codeword which is transmitted through a channel. The received codeword is then decoded and the corresponding quantized value is generated as output.

- relative error

$$\varepsilon_r(x, x_i) = \frac{|x - x_i|}{x} \tag{B.4}$$

Concerning the *cost* of the quantization, we will consider the following four possibilities

- expected absolute error

$$\mathcal{E}_{ea}(\mathcal{Q}, \mathcal{C}) = \sum_{i=1}^{N} \int_{C_i} \varepsilon_a(x, x_i)\, p(x)\, dx \tag{B.5}$$

- expected relative error

$$\mathcal{E}_{er}(\mathcal{Q}, \mathcal{C}) = \sum_{i=1}^{N} \int_{C_i} \varepsilon_r(x, x_i)\, p(x)\, dx \tag{B.6}$$

- maximum absolute error

$$\mathcal{E}_{ma}(\mathcal{Q}, \mathcal{C}) = \max_{[x_{min}, x_{max}]} \varepsilon_a(x, x_i) \tag{B.7}$$

- maximum relative error

$$\mathcal{E}_{mr}(\mathcal{Q}, \mathcal{C}) = \max_{[x_{min}, x_{max}]} \varepsilon_r(x, x_i) \tag{B.8}$$

where $p(x)$is the probability density function of x.

B.4 Optimizing for expected error

Using the expected error as quantization cost, the generic expression for the cost is

$$\mathcal{E}_e(\mathcal{Q}, \mathcal{C}) = \sum_{j=1}^{N} \int_{C_j} \varepsilon(x, x_j)\, p(x)\, dx \tag{B.9}$$

and we can formulate the following two necessary optimality conditions (Max, 1960; Lloyd, 1982)

1. If we assume the cells $C_j \in \mathcal{C}$ as preassigned and fixed, the necessary conditions for a minimum relatively to the position of the

quantized values, are

$$\frac{\partial \mathcal{E}_e(\mathcal{Q},\mathcal{C})}{\partial x_i} = 0 \qquad i = 2,\ldots,N-1 \tag{B.10}$$

where the range of the index is determined by the fact that we assume x_1, and x_N to be fixed at the boundaries of the quantized interval $\mathcal{X}$. Applying the condition expressed by Equation B.10 to Equation B.9 gives

$$\sum_{j=1}^{N} \frac{\partial \left(\int_{C_j} \varepsilon(x,x_j)\, p(x)\, dx \right)}{\partial x_i} = \\ = \frac{\partial \left(\int_{C_i} \varepsilon(x,x_i)\, p(x)\, dx \right)}{\partial x_i} = 0 \qquad i = 2,\ldots,N-1 \tag{B.11}$$

Differentiating under the integral sign[7] we obtain

$$\int_{C_i} \frac{\partial \varepsilon(x,x_i)}{\partial x_i}\, p(x)\, dx = 0 \qquad i = 2,\ldots,N-1 \tag{B.12}$$

2. If we consider the quantized values $x_i \in \mathcal{Q}$ as preassigned and fixed, we can minimize $\mathcal{E}_e(\mathcal{Q},\mathcal{C})$ by assigning to each cell C_i all the values $x \in \mathcal{X}$ which can actually appear as the input of the quantizer (i.e., for which $p(x) \neq 0$) and for which $\varepsilon(x,x_j)$ is minimized by the choice $x_j = x_i$. This leads to the condition

$$C_i \supset \{x \in \mathcal{X} \,:\, \varepsilon(x,x_i) < \varepsilon(x,x_j) \ \text{ for all } j \neq i\,,\ p(x) \neq 0\} \\ i = 1,\ldots,N \tag{B.13}$$

Note that for a generic error function $\varepsilon(x,x_i)$ and probability density $p(x)$ this condition does not uniquely determine the cells. However, with some simple additional conditions on both $\varepsilon(x,x_i)$ and $p(x)$ which will be satisfied in the cases considered below, there is no ambiguity in partitioning $\mathcal{X}$ into a collection of cells C_I according to Equation B.13.

For generic error functions $\varepsilon(x,x_j)$ and probability density functions $p(x)$, these conditions are necessary but not sufficient to ensure the presence of a minimum of the cost and, therefore, of an optimal quan-

[7] A sufficient condition for the differentiability under the integral sign is that $\varepsilon(x,x_j)$ be continuous and with continuous derivative with respect to x_i, although the differentiability may subsist with weaker conditions (Courant and John, 1989, p. 74).

tizer. Trushkin (1982) stated a series of sufficient conditions for the error function and probability density functions. His proofs are based on the observation that assuming the validity of a series of hypothesis corresponding to those listed at the start of the previous section, Equation B.9 has at least a minimum in the interior of the domain. Hence, if the unicity of the solution to Equations B.12 and Equations B.13 can be proved, the solution corresponds necessarily to the optimal quantizer, which is unique. Whereas Trushkin (1982) formulates general conditions for $\varepsilon(x, x_j)$ and $p(x)$ that are then proved to ensure the unicity of the above mentioned solution, we will be content to consider in the following sections some particular cases of $p(x)$ which, when combined with one of the two instances of $\varepsilon(x, x_j)$ considered above, lead to a manifestly unique solution of Equations B.12 and Equations B.13 and, therefore, ensure the optimality of the quantizer.

B.4.1 Absolute error

Substituting $\varepsilon_a(x, x_i) = |x - x_i|$ for $\varepsilon(x, x_i)$ in Equation B.13 we obtain

$$C_i \supset \{x \in \mathcal{X} : |x - x_i| < |x - x_j| \text{ for all } j \neq i\,,\, p(x) \neq 0\}$$
$$i = 1, \ldots, N \quad \text{(B.14)}$$

This means that, apart from the points where $p(x) = 0$, whose assignment to a particular cell is in any case immaterial, the generic cell C_i is constituted by the points of $\mathcal{X}$ that are closer to x_i than to any other quantized value x_j. Points that are equidistant from two quantized values x_i and x_j can be assigned equally well to C_i or to C_j. This means that the cells can be assumed to correspond to subintervals $C_i = (t_{i-1}, t_i]$ determined by a set $\mathcal{T} = \{t_0, \ldots, t_N\}$ of *thresholds* – we will call *subinterval quantizer* this kind of quantizer – that bisect the quantization interval $[x_i, x_{i+1}]$, so that

$$t_i = \frac{x_i + x_{i+1}}{2} \qquad i = 1, \ldots, N-1 \quad \text{(B.15)}$$

with the exception of $t_0 = x_1$ and $t_N = x_N$. Using this result in Equation B.12, with the current expression $\varepsilon_a(x, x_i) = |x - x_i|$ for the error function $\varepsilon(x, x_i)$, we obtain

$$\int_{\frac{x_{i-1}+x_i}{2}}^{\frac{x_i+x_{i+1}}{2}} \frac{\partial\,|x - x_i|}{\partial x_i}\, p(x)\, dx = 0 \qquad i = 2, \ldots, N-1 \quad \text{(B.16)}$$

which corresponds to

$$\int_{\frac{x_{i-1}+x_i}{2}}^{x_i} p(x)\,dx = \int_{x_i}^{\frac{x_i+x_{i+1}}{2}} p(x)\,dx \qquad i = 2,\ldots,N-1 \tag{B.17}$$

For a generic probability density function the system of equations in the unknowns x_i and t_i constituted by Equation B.15 and Equation B.17 cannot be solved analytically. Lloyd (1982) introduced two iterative algorithms for the determination of the quantizers in the general case.

Uniform distribution of probability In the particular case where $p(x)$ is the *uniform probability density function*

$$p(x) = \begin{cases} k & x \in \mathcal{X} \\ 0 & x \notin \mathcal{X} \end{cases} \tag{B.18}$$

where k is the normalizing constant, Equation B.17 becomes simply

$$x_{i+1} - x_i = x_i - x_{i-1} \qquad i = 2,\ldots,N-1 \tag{B.19}$$

and this shows that the unique optimal quantizer in this case is the *uniform quantizer* satisfying the conditions (Kassam, 1978)

$$x_{i+1} - x_i = \frac{x_{max} - x_{min}}{N-1} = \Delta \qquad i = 1,\ldots,N-1 \tag{B.20}$$

$$x_1 = x_{min}\,,\; x_N = x_{max}$$

$$t_i = \frac{x_i + x_{i+1}}{2} \qquad i = 1,\ldots,N-1 \tag{B.21}$$

which distributes uniformly the quantized values in the interval $[x_{min}, x_{max}]$ while complying with the conditions $x_1 = x_{min}$, $x_N = x_{max}$. The corresponding decoder $d_u(\cdot)$ is described by

$$\begin{aligned} x_i = d_u(i) &= x_{min} + (i-1)\cdot\Delta = \\ &= x_{min} + (i-1)\cdot\frac{x_{max} - x_{min}}{N-1} \qquad i = 1,\ldots,N \end{aligned} \tag{B.22}$$

An example of this kind of decoding function is illustrated in Figure B.3.

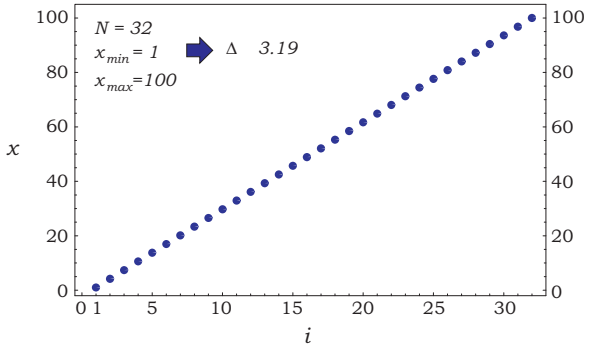


Figure B.3: *The graph of the linear decoder corresponding to the uniform quantization of the interval* $[x_{min}, x_{max}] = [1, 100]$ *using* $N = 32$ *quantized values with the conditions* $x_1 = x_{min}$, $x_N = x_{max}$.

B.4.2 Relative error

Substituting $\varepsilon_r(x, x_i) = \frac{|x-x_i|}{x}$ for $\varepsilon(x, x_i)$ in Equation B.13 we obtain

$$C_i \supset \{x \in \mathcal{X} : \frac{|x - x_i|}{x} < \frac{|x - x_j|}{x} \text{ for all } j \neq i\,,\, p(x) \neq 0\} \qquad i = 1, \ldots, N \tag{B.23}$$

Since we assume $x > 0$, this condition is equivalent to Equation B.14 and leads once again to the result that the quantizer can be assumed to be a subinterval quantizer whose cells $C_i = (t_{i-1}, t_i]$ are determined by thresholds that bisect the intervals $[x_i, x_{i+1}]$

$$t_i = \frac{x_i + x_{i+1}}{2} \qquad i = 1, \ldots, N-1 \tag{B.24}$$

apart from the cases $t_0 = x_1$ and $t_N = x_N$. Equation B.12 now becomes

$$\int_{\frac{x_{i-1}+x_i}{2}}^{\frac{x_i+x_{i+1}}{2}} \frac{\partial(\frac{|x-x_i|}{x})}{\partial x_i}\, p(x)\, dx = 0 \qquad i = 2, \ldots, N-1 \tag{B.25}$$

and corresponds to

$$\int_{\frac{x_{i-1}+x_i}{2}}^{x_i} \frac{1}{x}\, p(x)\, dx = \int_{x_i}^{\frac{x_i+x_{i+1}}{2}} \frac{1}{x}\, p(x)\, dx \qquad i = 2, \ldots, N-1 \tag{B.26}$$

Once again, for a generic probability density function the system of equations constituted by Equation B.24 and Equation B.26 cannot be solved analytically and iterative methods such as those proposed by Lloyd (1982) must be applied to determine the optimal quantizer.

Reciprocal distribution of probability In the particular case where $p(x)$ is the *reciprocal probability density function*

$$p(x) = \begin{cases} k/x & x \in \mathcal{X} \\ 0 & x \notin \mathcal{X} \end{cases} \tag{B.27}$$

where k is the normalizing constant, Equation B.26 reduces to

$$\frac{x_{i+1}}{x_i} = \frac{x_i}{x_{i-1}} \qquad i = 2, \ldots, N-1$$

and this shows that the unique optimal quantizer in this case is the *logarithmic quantizer* satisfying the conditions

$$\frac{x_{i+1}}{x_i} = \beta = \left(\frac{x_{max}}{x_{min}}\right)^{\frac{1}{N-1}} \qquad i = 1, \ldots, N-1 \tag{B.28}$$

$$x_1 = x_{min}\,,\; x_N = x_{max}$$

$$t_i = \frac{x_i + x_{i+1}}{2} \qquad i = 1, \ldots, N-1 \tag{B.29}$$

where β is the *base* of the quantizer. Note that the quantizer is named after the encoder $e : \mathcal{X} \to \mathcal{I}$, which is based on a logarithmic function, whereas the decoder $d_l : \mathcal{I} \to \mathcal{Q}$ is an exponential function $d_l(\cdot)$ that can be written as follows

$$x_i = d_l(i) = x_{min} \cdot \beta^{i-1} = x_{min} \cdot \left(\frac{x_{max}}{x_{min}}\right)^{\frac{i-1}{N-1}} \qquad i = 1, \ldots, N \tag{B.30}$$

An example of this kind of decoder function is shown in Figure B.4.

Remark In communication engineering a composite quantizer that is called logarithmic is obtained by first subjecting the input signal x to the action of a device known as compressor that – disregarding scaling factors – gives as output a signal with amplitude $y = \log_\beta(x+1)$, and then quantizing uniformly y (Panter and Dite, 1951; Smith, 1957; Jayant and Noll, 1984). It is easily verified that the thresholds of the composite quantizer do not follow the distribution prescribed by Equa-

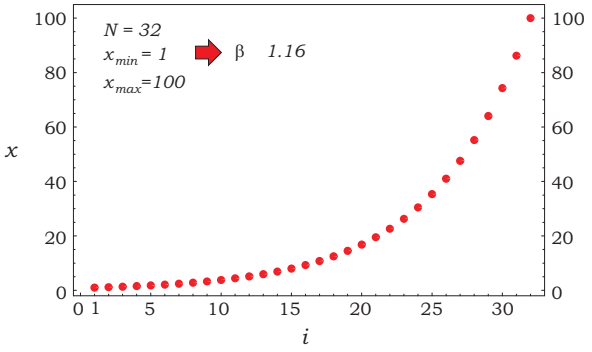


Figure B.4: *The graph of the exponential decoder corresponding to the logarithmic quantization of the interval* $[x_{min}, x_{max}] = [1, 100]$ *using* $N = 32$ *quantized values with the conditions* $x_1 = x_{min}$, $x_N = x_{max}$.

tion B.29 and the distribution of the quantized values does not conform to Equation B.28. Therefore, the composite quantizer differs from what we have called logarithmic quantizer. It is interesting to note, however, that for $x \ll 1$ the compressing transfer function is approximately linear, and for $x \gg 1$ it approximates $y = \log_\beta x$ (Figure B.5). Consequently, the composite quantizer appears as approximating the distribution of the quantization values of a uniform quantizer for small x, and the distribution of quantization values of a logarithmic quantizer for large x, with a smooth transition between the two.

B.5 Optimizing for maximum error

Using the maximum error as quantization cost, the expression for the cost is

$$\mathcal{E}_m(\mathcal{Q}, \mathcal{C}) = \max_{\substack{x \in \mathcal{X} \\ p(x) \neq 0}} \varepsilon(x, q(x)) \tag{B.31}$$

Since $q(x) = x_i$ when $x \in C_i$, and the cells C_i partition $\mathcal{X}$ we can rewrite Equation B.31 as

$$\mathcal{E}_m(\mathcal{Q}, \mathcal{C}) = \max_{x_i \in \mathcal{Q}} \max_{\substack{x \in C_i \\ p(x) \neq 0}} \varepsilon(x, x_i) \tag{B.32}$$

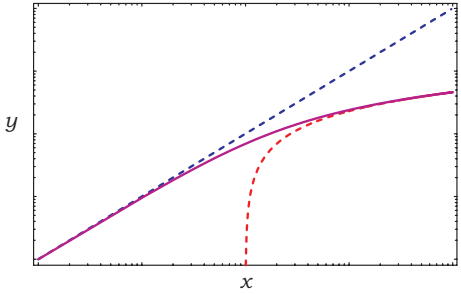


Figure B.5: *A composite quantizer used in communication engineering is based on the use of a compressing function $y = \log_\beta(x+1)$ (continuous curve) followed by a uniform quantizer. The compressing function approximates a linear function (top dashed curve) for small x and a logarithmic function (bottom dashed curve) for large x. Hence, the composite quantizer approximates the distribution of the quantization values of a uniform quantizer for small x, and the distribution of quantization values of a logarithmic quantizer for large x.*

B.5.1 Absolute error

Substituting $\varepsilon_a(x, x_i) = |x - x_i|$ as $\varepsilon(x, x_i)$ in Equation B.32 we obtain

$$\mathcal{E}_{ma}(\mathcal{Q}, \mathcal{C}) = \max_{x_i \in \mathcal{Q}} \max_{\substack{x \in C_i \\ p(x) \neq 0}} |x - x_i| \tag{B.33}$$

This means that the cost of the quantization corresponds to the maximum distance determined by the quantizer between a value $x \in \mathcal{X}$ and the value $x_i = q(x)$ into which it is transformed. Assuming as given and predefined the set $\mathcal{Q}$ of quantized values x_i, a policy that minimizes this distance for *all* the points $x \in \mathcal{X}$ is the by now familiar expression

$$C_i \supset \{x \in \mathcal{X} \,:\, |x - x_i| < |x - x_j| \;\text{ for all } j \neq i\,,\, p(x) \neq 0\} \qquad i = 1, \ldots, N \tag{B.34}$$

This assignment realizes thus a minimum of $\mathcal{E}_{ma}(\mathcal{Q}, \mathcal{C})$ for the given $\mathcal{Q}$, although other quantizers achieving the same cost can in general exist for the same $\mathcal{Q}$. Following the arguments that led to Equation B.15 we can infer that for each set of quantized values $\mathcal{Q}$ there exists a subinterval quantizer that minimizes $\mathcal{E}_{ma}(\mathcal{Q}, \mathcal{C})$ and whose cells correspond to subintervals $C_i = (t_{i-1}, t_i]$ that are determined by a set $\mathcal{T} = \{t_0, \ldots, t_N\}$ of

thresholds that bisect the quantization interval $[x_i, x_{i+1}]$, according to

$$t_i = \frac{x_i + x_{i+1}}{2} \qquad i = 1, \ldots, N-1 \tag{B.35}$$

apart from $t_0 = x_1$ and $t_N = x_N$. Let us call this quantizer the subinterval quantizer associated with $\mathcal{Q}$. The interesting consequence of the minimality of the cost of the subinterval quantizer for a given $\mathcal{Q}$ is that if there exists a quantizer achieving a cost $\mathcal{E}_{ma}(\mathcal{Q}, \mathcal{C})$ for $\mathcal{X}$, there certainly exists a subinterval quantizer for $\mathcal{X}$ which has equivalent or better performance. We can use this fact to prove that the uniform quantizer on $\mathcal{X}$ realizes the unique global optimum. To this end it is sufficient to consider that for a generic subinterval quantizer, assuming $p(t_i) \neq 0$ for all i, the cost corresponds to

$$\mathcal{E}_{ma}(\mathcal{Q}, \mathcal{C}) = \max_{i=1,\ldots,N-1} \frac{x_{i+1} - x_i}{2} = \max_{i=1,\ldots,N-1} \frac{\Delta_i}{2} \tag{B.36}$$

(where $\Delta_i = x_{i+1} - x_i$). The cost corresponds thus to the half-width of the largest of the subintervals determined by the quantized values. In particular, for the uniform quantizer we have

$$\Delta_i = \Delta = \frac{x_{max} - x_{min}}{(N-1)} \qquad i = 1, \ldots, N-1 \tag{B.37}$$

and

$$\mathcal{E}_{ma}(\mathcal{Q}, \mathcal{C}) = \frac{\Delta}{2} = \frac{x_{max} - x_{min}}{2\,(N-1)} \tag{B.38}$$

Should there exist a quantizer achieving a smaller cost than that given by Equation B.38, there would also exist a subinterval quantizer with that cost, or smaller. From Equation B.36 and Equation B.38, this can be expressed as

$$\max_{i=1,\ldots,N-1} \frac{\Delta_i}{2} < \frac{\Delta}{2} = \frac{x_{max} - x_{min}}{2(N-1)} \tag{B.39}$$

where the left-hand side of the inequality refers to the hypothetical subinterval quantizer that outperforms the uniform quantizer, to which the right-hand side refers. This leads to

$$\Delta_i < \frac{x_{max} - x_{min}}{(N-1)} \qquad i = 1, \ldots, N-1 \tag{B.40}$$

and finally to

$$\sum_{i=1}^{N-1} \Delta_i < x_{max} - x_{min} \tag{B.41}$$

In words, all the $N-1$ subintervals determined by the quantized values of the hypothetical subinterval quantizer that outperforms the uniform quantizer must have a length which is less than $\frac{|x_{max}-x_{min}|}{N-1}$. Therefore, these subintervals cannot possibly span the interval $\mathcal{X}=[x_{min}, x_{max}]$, which they are supposed to partition. This contradicts the hypothesis of the existence of a quantizer achieving a smaller cost than the uniform one. Note that in this case – contrary to the case where the cost is the estimated instead of the maximum error – the optimality of the uniform quantizer is independent from the probability density function $p(x)$.

B.5.2 Relative error

Substituting $\varepsilon_r(x, x_i) = \frac{|x-x_i|}{x}$ as $\varepsilon(x, x_i)$ in Equation B.32 we obtain

$$\mathcal{E}_{mr}(\mathcal{Q}, \mathcal{C}) = \max_{x_i \in \mathcal{Q}} \max_{\substack{x \in C_i \\ p(x)\neq 0}} \frac{|x - x_i|}{x} \tag{B.42}$$

Using arguments similar to those just used in the absolute error case, we can show that the subinterval quantizer defined by the bisecting thresholds of Equation B.35 realizes the (not necessarily unique) minimum cost $\mathcal{E}_{mr}(\mathcal{Q}, \mathcal{C})$ for a given set of quantized values $\mathcal{Q}$. For a generic subinterval quantizer, assuming $p(t_i) \neq 0$ for all i, it can be easily verified that the cost corresponds to

$$\mathcal{E}_{mr}(\mathcal{Q}, \mathcal{C}) = \max_{i=1,\ldots,N-1} \frac{\frac{x_{i+1}}{x_i} - 1}{\frac{x_{i+1}}{x_i} + 1} = \max_{i=1,\ldots,N-1} \frac{\beta_i - 1}{\beta_i + 1} \tag{B.43}$$

with $\beta_i = \frac{x_{i+1}}{x_i} > 1$. For a logarithmic quantizer (Equation B.28) we have in particular

$$\beta_i = \beta = \left(\frac{x_{max}}{x_{min}}\right)^{\frac{1}{N-1}} \qquad i = 1, \ldots, N-1 \tag{B.44}$$

and

$$\mathcal{E}_{mr}(\mathcal{Q}, \mathcal{C}) = \frac{\beta - 1}{\beta + 1} \tag{B.45}$$

We can repeat, *mutatis mutandis*, the arguments used to prove the optimality of the uniform quantizer in the case of the absolute error to prove that the logarithmic quantizer is the optimal quantizer for the maximum relative error. Should there exist a quantizer achieving a

smaller cost than that given by Equation B.45, there would also exist a subinterval quantizer with that cost, or smaller. This can be expressed as

$$\max_{i=1,\ldots,N-1} \frac{\beta_i - 1}{\beta_i + 1} < \frac{\beta - 1}{\beta + 1} \tag{B.46}$$

where the left hand side of the inequality refers to the hypothetical subinterval quantizer that outperforms the uniform quantizer, to which the right hand side refers. This leads to

$$\frac{\beta_i - 1}{\beta_i + 1} < \frac{\beta - 1}{\beta + 1} \qquad i = 1, \ldots, N-1 \tag{B.47}$$

Since the $\frac{\beta-1}{\beta+1}$ is a strictly increasing function of β for $\beta > 1$, this gives

$$\beta_i < \beta \qquad i = 1, \ldots, N-1 \tag{B.48}$$

and thus

$$\prod_{i=1}^{N-1} \beta_i < \beta^{N-1} = \frac{x_{max}}{x_{min}} \tag{B.49}$$

However, by definition we have $x_{max} = x_N = \beta_{N-1} x_{N-1} = \beta_{N-2}\beta_{N-1} x_{N-2} = \cdots = \prod_{i=1}^{N-1} \beta_i x_1 = \prod_{i=1}^{N-1} \beta_i x_{min}$, which contradicts Equation B.49 and the hypothesis of the existence of a quantizer achieving a smaller cost than the logarithmic one.

B.6 Uniform quantizer

It is instructive to examine the behavior of the absolute and relative error functions or the two optimal quantizers that we have determined above: the uniform and the logarithmic quantizer. Figure B.6 shows the graph of the absolute error function $\varepsilon_a(x)$ and of the relative error function $\varepsilon_r(x)$ for a uniform quantizer. Both error functions have a series of local maxima, one within each subinterval defined by two adjacent quantized values. The value of these local maxima for the absolute error does not depend on the index i of the interval $[x_i, x_{i+1}]$, and is given by Equation B.38. The value of the local maxima of the relative error depends instead on the index i, and is given by the following expression

$$\max_{[x_i, x_{i+1}]} \varepsilon_r(x) = \frac{1}{2\,i + 1 + 2\,(N-1) \,/\, \left(\frac{x_{max}}{x_{min}} - 1\right)} \qquad i = 1, \ldots, N-1 \tag{B.50}$$

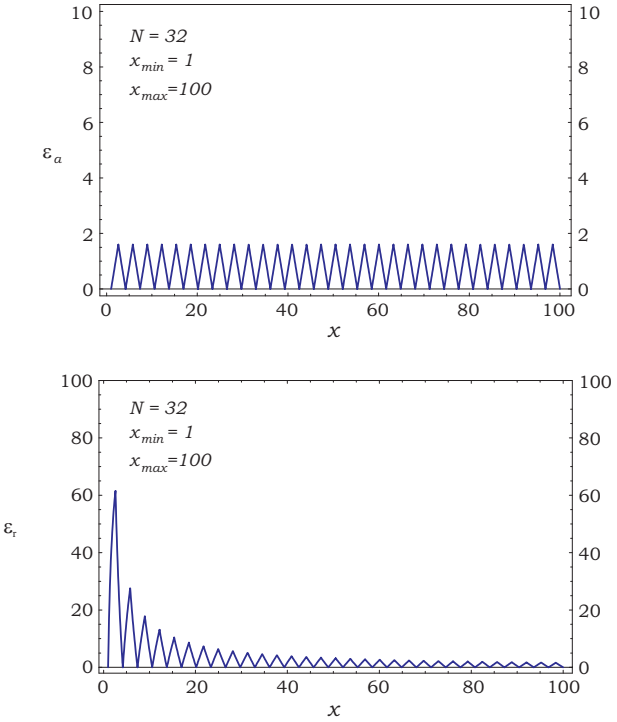


Figure B.6: *The graph of the absolute and relative quantization error functions $\varepsilon_a(x)$ and $\varepsilon_r(x)$ of the uniform quantizer with $N = 32$ quantized values distributed in the interval $[x_{min}, x_{max}] = [1, 100]$. Both error functions show a series of local maxima, one within each subinterval defined by two adjacent quantized values.*

with a maximum value of

$$\mathcal{E}_{mr}(\mathcal{Q}, \mathcal{C}) = \max_{[x_{min}, x_{max}]} \varepsilon_r(x) = \frac{1}{1 + 2\,(N-1)\,/\left(\frac{x_{max}}{x_{min}} - 1\right)} \tag{B.51}$$

For $N > 1$ Equation B.51 is a strictly increasing function of what in engineering parlance is called the *dynamic range* x_{max}/x_{min} of the quantized interval. Figure B.7 shows the graph of the function defined by Equation B.51 as a function of the dynamic range for various values

of N. This figure reveals that the maximum relative quantization error of the uniform quantizer increases rapidly with x_{max}/x_{min}, and for small values of N becomes unacceptable beyond moderate values of the dynamic range.

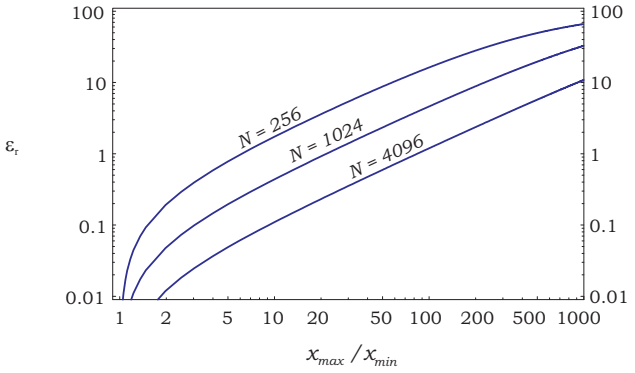


Figure B.7: *The dependence of the maximum relative quantization error of the uniform quantizer as a function of the dynamic range x_{max}/x_{min} of the quantized interval for various values of the number N of quantized values.*

In the perspective of conventional genetic representation of real parameters using binary strings, it is interesting to consider the number of bits n necessary to keep the maximum relative quantization error below a given limit $\overline{\varepsilon}_r$.[8] Considering that with n bits we can represent $N = 2^n$ quantized values, we obtain

$$n(\frac{x_{max}}{x_{min}}, \overline{\varepsilon}_r) = \left\lceil \log_2 \left[\frac{1}{2} \left(\frac{x_{max}}{x_{min}} - 1 \right) \left(\frac{1}{\overline{\varepsilon}_r} - 1 \right) + 1 \right] \right\rceil \qquad \text{(B.52)}$$

where $\lceil \cdot \rceil$ is the function that gives the smallest integer greater than or equal to its argument. The graphs of the function defined by Equation B.52 for some values of error and interval ranges are shown in Figure B.8.

Figure B.7 and Figure B.8 suggest that if we are concerned with the value of the relative quantization error, the uniform quantizer is

[8] This is related to what in communication engineering is called the *rate* of the quantizer.

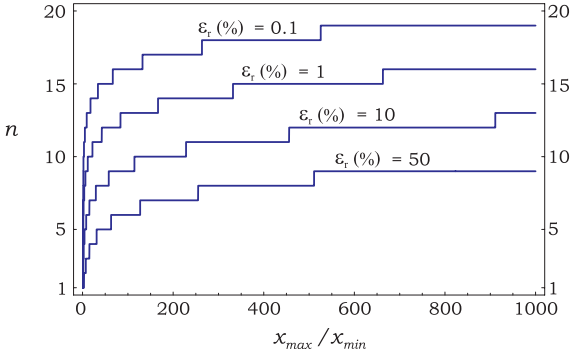


Figure B.8: *The number of bits of the binary representation of a real parameter needed to keep the relative quantization error of the uniform quantizer below a certain limit, plotted as a function of the dynamic range x_{max}/x_{min}.*

adequate only if we need to represent intervals characterized by small values of the dynamic range x_{max}/x_{min}. If we need instead to represent intervals with large values of that ratio, the behavior of the error near x_{min} forces the adoption of large values of n to prevent the quantization cost from becoming unacceptably large.

B.7 Logarithmic quantizer

Figure B.9 shows the graphs of the absolute error function $\varepsilon_a(x)$ and of the relative error function $\varepsilon_r(x)$ for the logarithmic and the uniform quantizer. The graphs of the error functions show a series of local maxima, one within each subinterval defined by two adjacent quantized values. The value of the local maxima of the absolute error – which was independent from i for the uniform quantizer – does depend on the index i of the interval $[x_i, x_{i+1}]$ in the case of the logarithmic quantizer. This is due to the fact that the distance Δ_i between pairs of adjacent quantized values is not constant across the interval $[x_{min}, x_{max}]$, being equal to

$$\Delta_i = x_{min} \cdot (\beta - 1) \cdot \beta^{i-1} \qquad i = 1, \ldots, N-1 \tag{B.53}$$

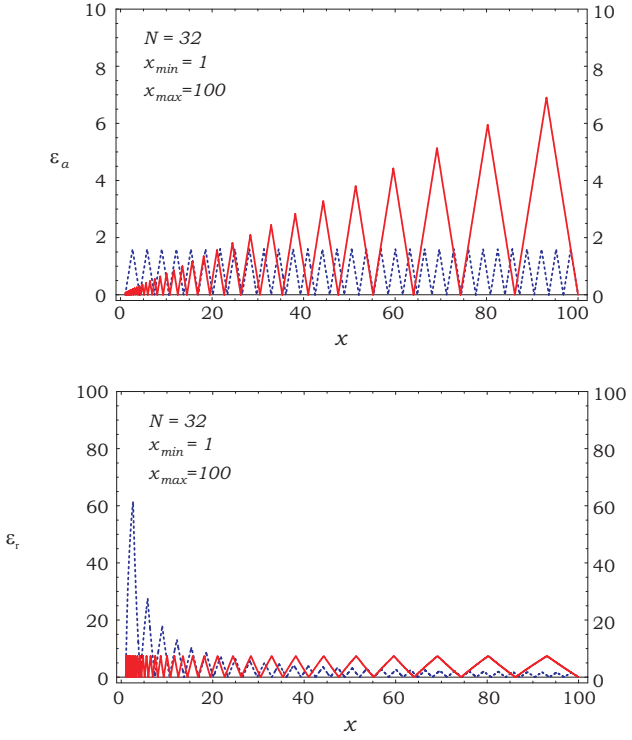


Figure B.9: *The graph of the absolute and relative quantization error functions* $\varepsilon_a(x)$ *and* $\varepsilon_r(x)$ *of the logarithmic quantizer (continuous line) compared with the corresponding graphs of the uniform quantizer (dashed line) with* $N = 32$ *quantized values distributed in the interval* $[x_{min}, x_{max}] = [1, 100]$.

with β defined by Equation B.44. The value of the local maxima of the absolute error corresponds to $\frac{\Delta_i}{2}$, with a global maximum of

$$\mathcal{E}_{ma}(\mathcal{Q}, \mathcal{C}) = \max_{[x_{min}, x_{max}]} \varepsilon_a(x) = \frac{1}{2} x_{min} \cdot (\beta - 1) \cdot \beta^{N-2} \tag{B.54}$$

Contrary to the case of the uniform quantizer, the value of the local maxima for the relative error of the logarithmic quantizer does not depend on the index i of the interval $[x_i, x_{i+1}]$, and is given by Equa-

tion B.45, which can be rewritten as

$$\mathcal{E}_{mr}(\mathcal{Q},\mathcal{C}) = \max_{[x_{min},x_{max}]} \varepsilon_r(x) = \frac{\left(\frac{x_{max}}{x_{min}}\right)^{\frac{1}{N-1}} - 1}{\left(\frac{x_{max}}{x_{min}}\right)^{\frac{1}{N-1}} + 1} \tag{B.55}$$

For $N > 1$, Equation B.55 is a strictly increasing function of the dynamic range x_{max}/x_{min} of the quantized interval. Figure B.10 shows the graphs of the maximum relative error of the logarithmic and uniform quantizer as a function of the dynamic range, for various values of N. This figure reveals that the magnitude of the maximum relative quantization error of the logarithmic quantizer – which is optimal for this cost criterion – increases slowly with x_{max}/x_{min} with respect to the uniform quantizer.

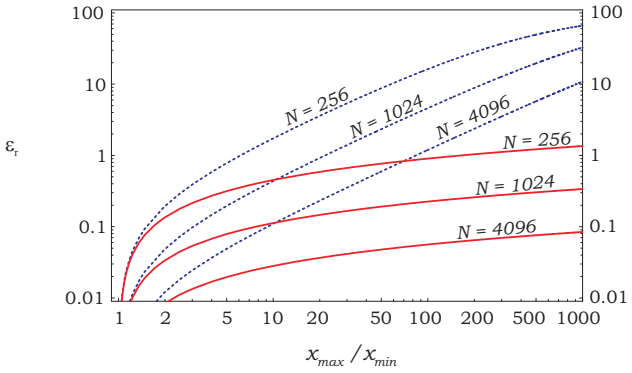


Figure B.10: *The maximum relative quantization error of the logarithmic quantizer (continuous line) and of the uniform quantizer (dashed line) as a function of x_{max}/x_{min} for various values of the number N of quantized values. The magnitude of the maximum relative quantization error of the logarithmic quantizer remains acceptable for much larger values of the dynamic range x_{max}/x_{min} with respect to the uniform quantizer.*

Like in the case of the uniform quantizer, we can determine the number of bits n of a binary string encoding the quantized values, required to keep the maximum relative quantization error below a given

limit $\overline{\varepsilon}_r$, which is given by the following expression

$$n(\frac{x_{max}}{x_{min}}, \overline{\varepsilon}_r) = \left\lceil \log_2 \left(1 + \frac{\log\left(\frac{x_{max}}{x_{min}}\right)}{\log\left(\frac{1+\overline{\varepsilon}_r}{1-\overline{\varepsilon}_r}\right)} \right) \right\rceil \tag{B.56}$$

The graphs of the function defined by Equation B.56 for some values of error and interval ranges are shown in Figure B.11. The comparison of Figure B.11 with Figure B.8 shows that the number of bits required to achieve a given error is much smaller in the case of the logarithmic quantizer.

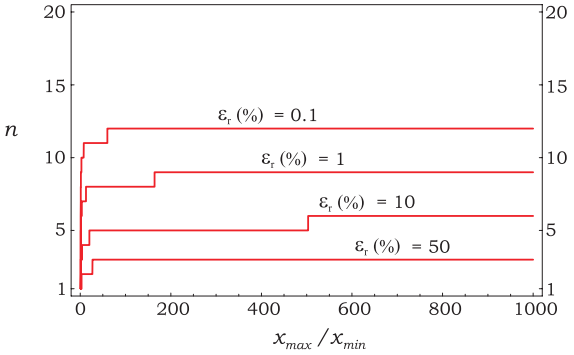


Figure B.11: *The number of bits of the binary representation of a real parameter needed to keep the relative quantization error of the logarithmic quantizer below a certain limit, plotted as a function of the dynamic range x_{max}/x_{min}.*

Remark I Equation B.55 for the maximum relative error could be rewritten to accommodate the case $x_{min} = 0$. The result, however, is $\mathcal{E}_{mr}(\mathcal{Q}, \mathcal{C}) = 1$, independent of x_{max}. This property does not depend on the kind of encoding used, but on the definition of the relative error and the fact that when $x_{min} = 0$ some values of x are approximated by $x_0 = 0$. Those values obviously suffer of a 100% relative representation error. This means that if we assume that the representation is acceptable for our purposes only if the relative error is below a given limit $L < 1$, when we select $x_{min} = 0$ we are actually representing, at best, a set $\mathcal{X} = \{0\} \cup [x'_{min}, x_{max}]$ with $x'_{min} = x'_{min}(L) > 0$, and not the interval

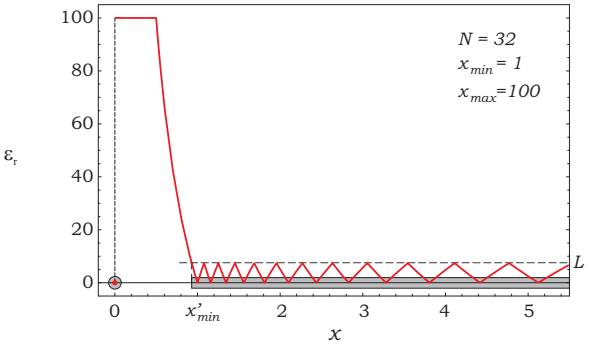


Figure B.12: *Due to its definition, the relative representation error equals* 100% *for the values* $x > 0$ *that are quantized to* $x_0 = 0$*. This means that even when the set of quantized values includes* $x_0 = 0$*, the set* $\mathcal{X}$ *(represented by the shaded region in the figure) that is quantized with a relative error smaller than a predefined value* $L < 100\%$ *has the form* $\{0\} \cup [x'_{min}, x_{max}]$ *with* $x'_{min} > 0$*.*

$[0, x_{max}]$ since there always is at least an interval $(0, x'_{min})$ where the approximation error is above the limit (Figure B.12). For the same reason, when we want to represent an interval of the form $[-x_{max_-}, x_{max_+}]$, with $x_{max_-} > 0$ and $x_{max_+} > 0$, the actual set that we can represent faithfully has the form $\mathcal{X} = [-x_{max_-}, -x'_{min_-}] \cup \{0\} \cup [x'_{min_+}, x_{max_+}]$.

Remark II So far we have considered only codes which distribute N points x_i, $i \in \{1, \ldots, N\}$ across an interval $[x_{min}, x_{max}]$, $x_{min} > 0$, while satisfying the constraints $x_1 = x_{min}$ and $x_N = x_{max}$. These constraints guarantee that the representation error vanishes at the boundary of the interval. However, it is clear that for any given maximum error L achieved by the code in $[x_{min}, x_{max}]$, there are two additional intervals – let us call them $[x'_{min}(L), x_{min}]$ and $[x_{max}, x'_{max}(L)]$ – where the error stays below that limit. Therefore, if the goal of the code is only to keep the error below a certain limit using a minimum amount of resources, it is preferable to allow the points x_1 and x_N to migrate to the interior of the interval $[x_{min}, x_{max}]$ and let the thresholds t_0 and t_N coincide with the boundaries of the interval (Figure B.13). The optimality results derived above for the case $x_1 = x_{min}$ and $x_N = x_{max}$ can be easily extended to this case.

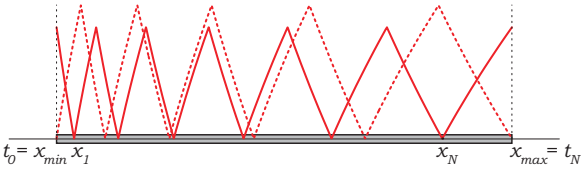


Figure B.13: *Dropping the constraints $x_1 = x_{min}$ and $x_N = x_{max}$ the quantized values x_1 and x_N of the logarithmic quantizer migrate to the interior of the interval $[x_{min}, x_{max}]$, thus producing a smaller maximum relative error with respect to the constrained case.*

Information entropy The definition of a quantizer $(\mathcal{Q}, \mathcal{C})$ entails the association with each quantized value $x_i \in \mathcal{Q}$ of a value of probability $P_i = \int_{C_i} p(x)\,dx$, where $p(x)$ is the probability density function of the quantized variable x, and $C_i \in \mathcal{C}$ is the quantization cell associated with x_i. The information entropy of the quantized variable is defined by $H = -\sum_i P_i \log P_i$, and is maximized when all P_i are equal (Shannon, 1948; Jaynes, 2003). It is easily verified that the uniform quantizer satisfies this maximization condition for a uniform probability distribution, since in this case we have $P_i = \int_{x_i - \frac{\Delta}{2}}^{x_i + \frac{\Delta}{2}} k\,dx = k\,\Delta$, which does not depend on i. Correspondingly, the logarithmic quantizer satisfies the information entropy maximization condition in presence of a reciprocal probability distribution, since we have $P_i = \int_{x_i(\frac{\beta+1}{2\beta})}^{x_i(\frac{\beta+1}{2})} \frac{k}{x}\,dx = k\,\ln\beta$.

B.8 Floating-point quantizer

The floating-point (FP) quantization is the most common way to represent real numbers in present-day general purpose digital computers. It is obtained mixing a logarithmic quantizer and a uniform quantizer. To obtain an FP quantizer we can image to define first a logarithmic quantizer with base $\beta > 1$ and N_e quantization values. According to Equation B.30 the corresponding decoder is described by

$$x_{i_e} = d_l(i_e) = x_{min} \cdot \beta^{\,i_e - 1} \qquad i_e = 1, \ldots, N_e \tag{B.57}$$

Now we define a uniform quantizer with $N_m + 1$ quantization values for each interval $[x_{i_e},\, \beta\, x_{i_e}]$, whose decoder corresponds to

$$x_{i_m} = d_u(i_m) = x_{i_e} + (i_m - 1) \cdot \frac{\beta\, x_{i_e} - x_{i_e}}{N_m} = \\ = x_{i_e} \left(1 + (i_m - 1) \cdot \frac{\beta - 1}{N_m}\right) \qquad i_m = 1, \ldots, N_m \quad \text{(B.58)}$$

The combined action of the logarithmic and uniform quantizer generates the FP quantizer, whose decoder takes as argument the pair of indexes (i_e, i_m) and corresponds to

$$x_{i_e,i_m} = d_{FP}(i_e, i_m) = x_{min} \cdot \beta^{i_e - 1} \left(1 + (i_m - 1) \cdot \frac{\beta - 1}{N_m}\right) \\ i_e = 1, \ldots, N_e \quad , \quad i_m = 1, \ldots, N_m \quad \text{(B.59)}$$

Note that although we define a uniform quantizer with $N_m + 1$ values for each interval $[x_{i_e},\, \beta\, x_{i_e}]$, the $(N_m + 1)$-th value is indexed by the pair $(i_e + 1, 1)$ and not by the pair $(i_e, N_m + 1)$. Therefore, we need to use only N_m values of the index i_m, and the total number of quantized values corresponds to $N = N_e \cdot N_m$. An example of FP decoder is shown in Figure B.14. In this case, given N_e, N_m, x_{min}, and x_{max}, the value of β is obtained solving the equation

$$x_{max} = x_{min} \left(1 + (\beta - 1) \frac{N_m - 1}{N_m}\right) \beta^{N_e - 1} \quad \text{(B.60)}$$

although, typically, the value of β is preassigned rather than calculated.

Figure B.15 shows the graph of the relative error $\varepsilon_r(x)$ function for an FP quantizer. The distance Δ_i between pairs of adjacent quantized values is not constant across the interval $[x_{min}, x_{max}]$, but is obviously constant within each uniformly quantized subinterval $[x_{i_e}, x_{i_e+1}]$, with

$$\Delta_{i_e} = x_{min} \cdot \beta^{i_e - 1} \cdot \frac{(\beta - 1)}{N_m} \qquad i_e = 1, \ldots, N_e - 1 \quad \text{(B.61)}$$

The value of the local maxima of the absolute error corresponds to $\frac{\Delta_{i_e}}{2}$, with a global maximum of

$$\mathcal{E}_{ma}(\mathcal{Q}, \mathcal{C}) = \max_{[x_{min}, x_{max}]} \varepsilon_a(x) = x_{min} \cdot \beta^{N_e - 2} \cdot \frac{(\beta - 1)}{2\, N_m} \quad \text{(B.62)}$$

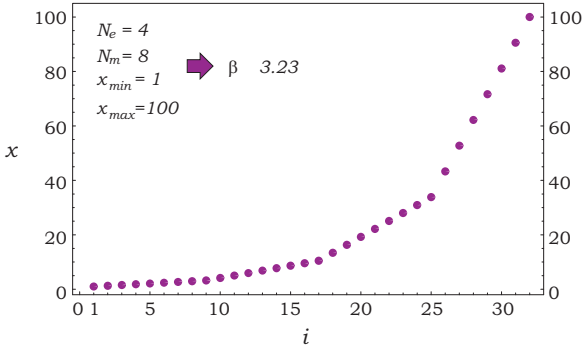


Figure B.14: *The graph of the floating-point (FP) decoder corresponding to the FP quantization of the interval* $[x_{min}, x_{max}] = [1, 100]$ *using* $N = 32$ *quantized values generated by* $N_e = 4$ *exponent elements and* $N_m = 8$ *mantissa elements. Here the value of the base* β *is determined as a consequence of the values of the other parameters of the FP quantization, even if in practice the value of* β *is preassigned and the value of some other parameter is computed instead.*

The global maximum of the relative error is equal to

$$\mathcal{E}_{mr}(\mathcal{Q}, \mathcal{C}) = \max_{[x_{min}, x_{max}]} \varepsilon_r(x) = \frac{1}{1 + \frac{2\,N_m}{\beta - 1}} \tag{B.63}$$

The graph of the relative error for the FP quantizer shown in Figure B.15 is similar to the corresponding graph for the logarithmic decoder shown in Figure B.9. Given the greater complexity of the FP decoder relatively to the logarithmic decoder, there seems to be no reason to opt for the latter. The reason for defining and using an FP quantizer resides in the difficulty of implementing operations of addition and subtraction with a fully logarithmic representation. On the other hand, multiplication, division, and root extraction are greatly simplified, and this has prompted many efforts aimed at the development of efficient algorithms with the corresponding circuit implementations for logarithmic arithmetic (Anuta et al., 1996; Arnold et al., 1990, 1992, 1998; Barlow and Bareiss, 1985; Coleman, 1995; Coleman and Kadlec, 2000; Coleman et al., 2000; Swartzlander and Alexopoulos, 1975; Volkov and Pakshin, 1992; Yu and Lewis, 1991).

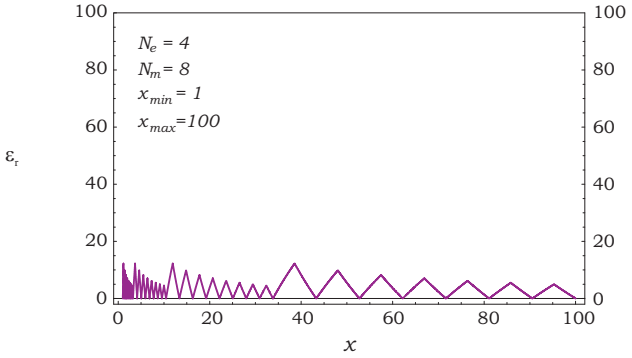


Figure B.15: *The graph of the relative quantization error function $\varepsilon_r(x)$ of the FP quantizer with $N_e = 4$ and $N_m = 8$ in the interval $[x_{min}, x_{max}] = [1, 100]$. The action of the linear part of the quantizer makes the local maxima vary within each subinterval, whereas that of the logarithmic part of the quantizer makes the sequence of maxima replicate in the different subintervals.*

Choosing the base β of the FP quantizer properly, it is possible to obtain a representation that does not seriously compromise the error performances of the logarithmic quantizer while allowing an easy implementation of all the arithmetic operations. On a binary computer the choice is typically $\beta = 2$. This corresponds to uniformly quantize intervals with a dynamic range $x_{max}/x_{min} = 2$. As Figure B.16 shows, with this small dynamic range, the performance of the uniform quantizer is comparable to that of the logarithmic quantizer. For example, the IEEE single precision format, has $\beta = 2$, $N_e = 2^{n_e} = 2^8$, $N_m = 2^{n_m} = 2^{23}$, and $x_{min} = 2^{-t} = 2^{-127}$, which corresponds to $x_{min} \simeq 5.877 \cdot 10^{-39}$, and gives $x_{max} \simeq 6.806 \cdot 10^{38}$, $\mathcal{E}_{mr}(\mathcal{Q},\mathcal{C})_{FP} \simeq 5.960 \cdot 10^{-8}$. For comparison, a logarithmic code using the same total number of quantized values on the same interval would have $\mathcal{E}_{mr}(\mathcal{Q},\mathcal{C})_{log} \simeq 4.131 \cdot 10^{-8}$, which corresponds to a ratio $\mathcal{E}_{mr}(\mathcal{Q},\mathcal{C})_{FP}/\mathcal{E}_{mr}(\mathcal{Q},\mathcal{C})_{log} \simeq 1.443$.[9] In view of this not dramatic difference in error performance, and considering the advantages in arithmetic manipulation, the adoption of FP as the standard

[9] In general, $\lim_{N_m \to \infty}(\mathcal{E}_{mr}(\mathcal{Q},\mathcal{C})_{FP}/\mathcal{E}_{mr}(\mathcal{Q},\mathcal{C})_{log}) = (\beta - 1)/\ln(\beta)$, where β is the base of the FP quantizer. The result given in the text for the case of the IEEE single precision FP format (for which $\beta = 2$) corresponds to an approximation of the limit value $1/\ln(2)$.

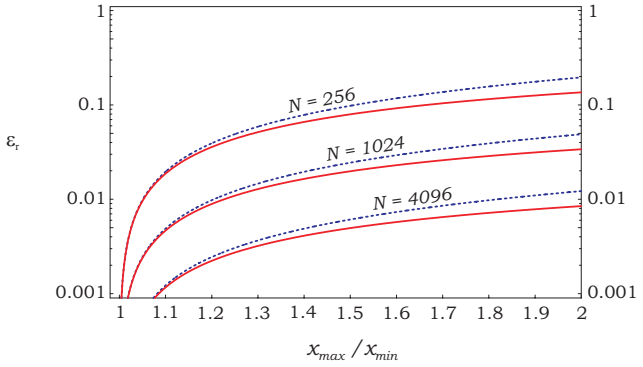


Figure B.16: *The detail of the maximum relative quantization error of the logarithmic quantizer (continuous line) and of the uniform quantizer (dashed line) as a function of x_{max}/x_{min} for various values of the number N of quantized values. Comparing these graphs with those of Figure B.10 shows that in this limited interval of dynamic ranges x_{max}/x_{min}, the performance of the uniform quantizer is comparable to that of the (optimal) logarithmic quantizer.*

representation of real values on digital computers is understandable. If, however, as is often the case in evolutionary computation, there is the need to efficiently represent many real parameters spanning a large dynamic range, with the possibility to convert them to FP prior to their algebraic manipulation, the logarithmic quantization is a good alternative, and becomes a necessity if a custom genetic encoding is required.

B.9 Discussion

The uniform quantizer is commonly assumed as the default quantizer in evolutionary experiments, and is indeed the optimal quantizer if the cost criterion is the maximum absolute error, or the expected absolute error in presence of a uniform probability density function. However, the analysis of the previous sections has shown that if the cost criterion is the maximum relative error, or the expected relative error in presence of a reciprocal probability density function, the optimal choice is the logarithmic quantizer. The pervasiveness of the floating-point representation of real numbers, and the example of the standard series of electric resistors mentioned before testify that the logarithmic and

quasi-logarithmic quantization is of great practical importance. Figure B.17 shows that the distribution of the multipliers of the E12 standard series (Equation B.2) approximates well that of the values of a logarithmic quantizer with $\beta = 10^{\frac{1}{12}} \approx 1.21$.

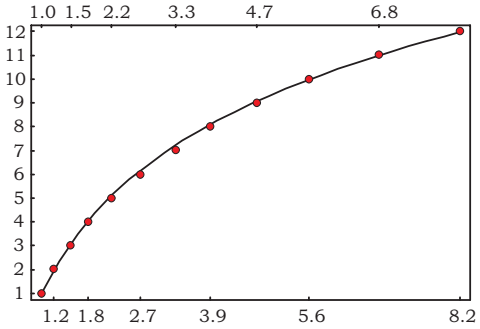


Figure B.17: *The distribution of the resistance values of the E12 standardized resistor series within each decade of resistance value is shown here to follow closely the graph of the logarithmic function corresponding to the encoder of a logarithmic quantizer with* $\beta = 10^{\frac{1}{12}} \approx 1.21$.

Disregarding the special case of *ad-hoc* quantizers that are neither uniform nor logarithmic, we can consider the criteria that can determine the choice between these two latter types of quantizer in setting up an evolutionary experiment. In some cases – like that of the resistors in electronic design – the choice is suggested by the evidence of the existing design practice, and the optimality criterion can be used in the reverse direction to reveal the kind of error that is presumably relevant to the problem at hand (in the case of electrical resistance values in analog electronic circuits, the relative error) and, possibly, the probability distribution of the values that are subject to quantization (for electrical resistance values, a reciprocal distribution). In most circumstances, however, this kind of empirical evidence is not available. Hence, we must proceed by asking first what kind of error appears as most relevant with respect to the quantity that is quantized as a consequence of its genetic encoding. We have already argued in Section B.2 that the preeminence of the relative error is typically a characteristic of quantities that appear as multiplying or dividing factors, whereas

quantities appearing in sums and subtractions are best associated with an absolute error criterion of cost.

When the expected error rather than the maximum absolute or relative error can be assumed as the quantization cost criterion (see comments in Section B.2 on the assessment of this issue), it is necessary to verify also if the probability density function can be assumed as uniform or reciprocal. Once again the relevance of multiplication or summation operations can guide the choice. If no other information about the probability distribution of a variable is known except that is must be translation-invariant, then invariance considerations lead to the choice of a uniform probability density function. If, on the other hand, the only information available is that of scale-invariance, invariance considerations lead to the choice of the reciprocal probability density function (Jeffreys, 1961; Jaynes, 2003). Note that this must not be interpreted as saying that our information about the invariance properties determines the actual frequency of the quantity values as uniform or reciprocal, but that these assignments of the probability density function are the best guesses that we can make, based on the information available.

Another useful information for the estimation of the probability distribution of a variable is the knowledge that it is the result of the repeated summation and subtraction of many factors, since in this case we know that, considering each term as a random variable, our best guess for the probability distribution of the result is a Gaussian with a large variance, which approximates a uniform distribution. If the variable is instead the result of repeated multiplications and divisions, it is the probability of the logarithm of the results that approximates a uniform distribution, and, therefore, the probability of the result tends to a reciprocal distribution (Pietronero et al., 2001). For example, it is an observed fact that the probability density distribution of the first digit of numbers from many kinds of numerical tables is not uniform but tends to follow closely a reciprocal density distribution (Raimi, 1976; Pietronero et al., 2001). This fact – known as *Benford's law* – was attributed by Hamming (1970) to the effect of repeated application of arithmetic operations, especially multiplications and divisions.

Finally, in the context of natural evolution, the optimality of the logarithmic quantization in the presence of conditions that can be considered as plausible for signal encoding in biological organisms – especially in the case of sensory and actuation signals with high dynamic range – can motivate the presence of this kind of encoding in some stage

of biological signal processing thought of as “optimized” by evolution.

B.10 Conclusion

We have formulated and analyzed the problem of the representation of real parameters in evolutionary algorithms as a problem of quantization, and have derived some conditions for the optimality of the uniform and logarithmic quantizer. We have given some criteria for the assessment of the suitability of either of these quantizers. These criteria can assist the experimenter in the choice of a genetic encoding for real parameters that realizes a quantization which is adapted to the characteristics of the represented parameters. The expected result is a reduction of the effect of the quantization on the outcome of the evolutionary process. In particular, the multiple connections that have been revealed between the relevance of the multiplication for a variable, the pertinence of the relative error, and the optimality of the logarithmic quantization, justify its extensive use within this thesis in the context of the genetic representation of interaction strengths of analog networks. Moreover, the signaled connections suggest the examination of the possibility of adopting a genetic encoding realizing a logarithmic quantization in some cases – we think in particular to the encoding of the neuron weights in artificial neural networks – where a genetic encoding implementing a uniform quantization is typically adopted out of habit rather than with deliberation.

Appendix C

Curing SPICE

Overview

This appendix describes the code modifications that were required in order to use SPICE as a simulation engine for the evolutionary runs described in Chapter 4. In particular, it details how the problem of the numerous memory leaks existing in the original SPICE code was solved.

C.1 SPICE

SPICE (Simulation Program with Integrated Circuit Emphasis) is an analog electronic circuit simulator developed and released in the public domain by a group of researchers at the University of California at Berkeley. The beginnings of the program date back to the early 1970s. In the original batch simulation scenario, the user prepares a description of the circuit that must be simulated and of the kinds of analysis required. This description is constituted by a series of lines of text complying with a suitably defined SPICE syntax (for the details see the SPICE User's Manual (Quarles et al., 1989), or (Vladimirescu, 1994)). Since originally the description consisted of a deck of punched cards, the input file containing the description is usually called the "SPICE deck". To use SPICE the user submits a SPICE deck to the program and obtains an output file with the results of the analysis performed by the program.[1]

C.2 SPICE for evolution and optimization

The first two versions of SPICE – SPICE1 and SPICE2 – were written in FORTRAN and were especially targeted to the batch modality of simulation. At the beginning of the 1980s SPICE was partially ported to the C programming language and a simple interactive interface was added to the simulation environment. This porting effort resulted in the realization of the last release of the program - SPICE3 - from the part of the group of its creators. Since many bugs were found in the first release of SPICE3, a series of patched version followed, up to the version 3f4, which is the last official version released in the public domain. SPICE3f4 is the version used as a starting point for the electronic circuit simulation code used in this thesis.

Like almost all complex software systems, SPICE3f4 is not exempt from bugs and fragments of unpolished code. In particular, the program is plagued by the presence of innumerable operations of memory allocation that are not matched by the corresponding operations of memory release that should normally be invoked once the memory is no longer used. This produces what in the jargon of computer science is called a "memory leak" and corresponds to the presence of a certain amount of no longer accessible physical memory during the course of

[1] Appendix D shows some examples of SPICE decks.

the execution of the program. Inspection of the SPICE source code revealed that most, if not all, of these memory leaks is produced in the initial phase of the program execution, where the conversion the input deck into the internal matrix representation of the circuit is performed. This means that in the typical one-shot simulation scenario the presence of the memory leaks does not pose a serious obstacle to the use of SPICE3, since the program reads the input deck only once and then, after the execution of the required analyses, terminates releasing all the program memory (including the leaked parts) to the operating system.

In the context of an evolutionary system or of an optimization process, however, the presence of this kind of memory leak is not acceptable, since a large number of circuits must be simulated without terminating the program. Given that the simulation of each circuit requires the processing of an input deck, this would result in a progressive and potentially unbounded growth of the amount of leaked memory, leading eventually to the abnormal termination of the program. This means that a preliminary step for the use of SPICE3 as circuit simulation engine in our evolutionary system is the elimination of the memory leaks.

The inspection of the SPICE3f4 source code and some preliminary attempt at tracing and correcting the leaks one by one revealed the intricacy of this low-level approach and suggested the devisement of a higher level remedy. The solution eventually adopted is based on the supervisory control of the SPICE code by means of some lines of C++ code that, during a SPICE run, intercept all the memory management calls and build a table of pointers to the allocated memory. In this way, at the end of each circuit simulation the table can be used by the supervisory code to release the memory left unallocated by SPICE. Figure C.1 shows the memory management statistics generated by tracing the operations executed on the allocation table as a consequence of the intercepted memory management operations invoked by SPICE while simulating a circuit. The figure reveals that most of the memory explicitly allocated by SPICE during the simulation was never released by the original code, a phenomenon that was consistently observed in the simulation executed with the memory management tracing active. The use of the supervisory code solved completely the memory leak problem. Moreover, the increase in computational cost imposed on each simulation run by the presence of the additional supervisory code was found to be typically negligible when compared even only with the operation of translation of the input deck into the internal representation of the circuit.

```
==================================================
STDOUT
--------------------------------------------------
Number of "malloc" calls :  1012
Number of "calloc" calls :  0
Number of "realloc" calls:  127
Number of "free" calls :  214
Number of memory leaks :  798
Max size of set of allocated pointers:  804
Max size of allocated memory:  41578 bytes
Size of originally leaked memory:  41346 bytes
Time spent managing the set of pointers (s):  < 0.01
Duration of the SPICE run (s):  0.06
==================================================
STDERR
--------------------------------------------------
==================================================
CP_OUT
--------------------------------------------------
==================================================
CP_ERR
--------------------------------------------------
==================================================
RAWFILE
--------------------------------------------------
Output values:
sweep voltage:  2.00e+00 2.01e+00 ...
vl branch current:  -1.25e-09 -1.19e-09 ...
==================================================
```

Figure C.1: *Memory management statistics of the SPICE run for the deck of the Gaussian function generator circuit shown in Table D.3 on page 272. The lines boldface reveal that the original SPICE code did not release most of the memory allocated during the simulation of this circuit. If not cured, this behavior would result in an unbounded growth of the memory occupation during an evolutionary run using the original SPICE code as simulation engine. This memory leak phenomenon was observed in the simulation of all the circuits whose allocation statistics was analyzed. The presence of lines referring to* `STDERR`*,* `CP_OUT`*,* `CP_ERR`*, and* `RAWFILE` *streams along with the* `STDOUT` *stream used to collect the statistics, signals the fact that all the I/O and file streams used by the original C code of SPICE were redirected to C++ standard template library string streams. The resolution of the timers used to generate the timing data is 0.01 s.*

Another modification that is required to efficiently use SPICE as a simulation engine in an evolutionary and optimization context is the redirection to the physical memory of all the input and output activities of the program targeted at files resident on disk. This was obtained by intercepting all the original I/O calls referring to disk files (and also to the standard I/O streams) and redirecting them to C++ standard template library string streams resident in the physical memory.

Appendix D

SPICE decks

Overview

This appendix contains the SPICE input files ("SPICE decks") of the evolved circuits whose schematic is shown in Chapter 4.

```
* Voltage reference problem
*
.dc Vc 4 6 0.1
.save dc V(3)
.temp 0 25 50 75 100
*
*
.options
+ABSTOL=10.0E-09
+VNTOL=100.E-06
+RELTOL=1.00E-04
+GMIN=1.00E-10
+ITL1=500
+ITL2=200
+ITL4=50
+ITL6=0
+METHOD=TRAP
+MAXORD=2
*
.MODEL BC846B NPN
+IS=1.822e-14 NF=0.9932
+ISE=2.894e-16 NE=1.4
+BF=324.4 IKF=0.109 VAF=82
+NR=0.9931 ISC=9.982e-12
+NC=1.763 BR=8.29 IKR=0.09
+VAR=17.9 RB=10 IRB=5e-06
+RBM=5 RE=0.649 RC=0.7014
+XTB=0 EG=1.11 XTI=3
+CJE=1.244e-11 VJE=0.7579
+MJE=0.3656 TF=4.908e-10
+XTF=9.51 VTF=2.927
+ITF=0.3131 PTF=0
+CJC=3.347e-12 VJC=0.5463
+MJC=0.391 XCJC=0.6193
+TR=9e-08 CJS=0 VJS=0.75
+MJS=0.333 FC=0.979
*
*
Vc 1 0 10
Rc 1 2 1k
Rl 0 3 10k
*
Q1  4  5  6 BC846B
Q2  4  5  6 BC846B
Q3 10 11 12 BC846B
Q4 13 14 15 BC846B
Q5 16 17 18 BC846B
Q6 19 11 12 BC846B
Q7 22 23 24 BC846B
*
R1   0 16 2.371E+05
R2   0 17 1.778E+02
R3   0 24 7.499E+05
R4   2  3 1.909E+03
R5   2  5 3.750E+05
R6   2  6 5.623E+05
R7   2 15 2.699E+05
R8   2 18 7.366E+04
R9   3  6 5.623E+05
R10  3 14 9.955E+02
R11  3 15 5.623E+05
R12  3 18 1.724E+02
R13  4 14 6.817E+02
R14  4 15 3.065E+04
R15  4 16 1.000E+06
R16  4 22 5.620E+01
R17  5 11 9.529E+03
R18  5 23 4.217E+05
R19  6 12 1.302E+03
R20  6 15 2.812E+05
R21  6 19 4.217E+05
R22  6 23 7.499E+05
R23  6 24 4.800E+02
R24 11 16 1.000E+06
R25 11 23 2.699E+03
R26 12 14 5.623E+05
R27 12 15 1.000E+06
R28 12 17 7.499E+05
R29 12 22 1.000E+06
R30 12 23 2.812E+05
R31 12 24 1.581E+05
R32 14 15 7.499E+05
R33 14 17 4.217E+05
R34 14 18 1.778E+02
R35 14 22 7.499E+05
R36 15 16 7.499E+04
R37 15 22 1.000E+05
R38 16 22 7.499E+05
R39 16 24 5.623E+05
R40 17 24 2.371E+02
*
.end
```

Table D.1: *SPICE deck of the evolved voltage reference circuit shown in Figure 4.25 on page 163*

```
* Temperature sensor problem
*
.op
.save dc V(3)
.step temp 0 100 5
*
.options
+ABSTOL=10.0E-09 VNTOL=100.E-06
+RELTOL=1.00E-04 GMIN=1.00E-10
+ITL1=500 ITL2=200 ITL4=50
+ITL6=0 METHOD=TRAP MAXORD=2
*
.MODEL BC846B NPN
+IS=1.822e-14 NF=0.9932
+ISE=2.894e-16 NE=1.4 BF=324.4
+IKF=0.109 VAF=82 NR=0.9931
+ISC=9.982e-12 NC=1.763 BR=8.29
+IKR=0.09 VAR=17.9 RB=10
+IRB=5e-06 RBM=5 RE=0.649
+RC=0.7014 XTB=0 EG=1.11
+XTI=3 CJE=1.244e-11 VJE=0.7579
+MJE=0.3656 TF=4.908e-10
+XTF=9.51 VTF=2.927 ITF=0.3131
+PTF=0 CJC=3.347e-12 VJC=0.5463
+MJC=0.391 XCJC=0.6193 TR=9e-08
+CJS=0 VJS=0.75 MJS=0.333
+FC=0.979
*
.MODEL BC856B PNP
+IS=2.014e-14 NF=0.9974
+ISE=6.578e-15 NE=1.45 BF=315.3
+IKF=0.079 VAF=39.15 NR=0.9952
+ISC=1.633e-14 NC=1.15 BR=8.68
+IKR=0.09 VAR=9.5 RB=10
+IRB=5e-06 RBM=5 RE=0.663
+RC=0.718 XTB=0 EG=1.11 XTI=3
+CJE=1.135e-11 VJE=0.7071
+MJE=0.3808 TF=6.546e-10
+XTF=5.387 VTF=6.245 ITF=0.2108
+PTF=0 CJC=6.395e-12 VJC=0.4951
+MJC=0.44 XCJC=0.6288 TR=5.5e-08
+CJS=0 VJS=0.75 MJS=0.333
+FC=0.9059
*
Vs 4 0 -5
Vc 1 0 15
Rl 0 3 10k
Rc 1 2 1k
Rs 4 5 1k
*
Q11 36 37 38 BC856B
Q10 33 34 35 BC856B
Q9  30 31 32 BC856B
Q8  27 28 29 BC856B
Q1   6  7  8 BC846B
Q2   9 10 11 BC846B
Q3  12 13 14 BC846B
Q4  15 16 17 BC846B
Q5  18 19 20 BC846B
Q7  24 25 26 BC856B
*
R1   2 12 5.623E+05
R2   2 14 4.217E+05
R3   2 18 1.778E+02
R4   2 29 4.217E+05
R5   2 34 3.162E+05
R6   2 35 1.000E+06
R7   2 38 3.162E+05
R11  3  5 1.000E+06
R15  3 11 2.371E+03
R16  3 16 4.217E+05
R17  3 19 7.499E+05
R18  3 20 1.334E+02
R19  3 32 7.499E+05
R20  3 33 4.217E+05
R12  5  8 3.162E+05
R13  5 20 7.499E+05
R9   5 29 2.125E+05
R10  5 33 5.623E+05
R21  6 33 1.778E+05
R22  9 27 7.499E+05
R23 10 20 7.499E+05
R24 12 28 1.778E+05
R25 12 35 5.623E+05
R26 13 25 5.623E+05
R27 13 26 4.217E+05
R28 15 36 1.334E+05
R29 18 26 4.217E+05
R30 18 38 1.778E+05
R31 19 24 1.000E+05
R32 20 33 7.499E+04
R33 20 35 1.000E+06
R35 24 36 7.499E+05
R36 25 27 7.499E+05
R37 25 37 2.371E+05
R38 26 38 1.000E+02
R39 29 33 7.499E+05
R40 29 34 1.000E+06
R41 29 35 3.162E+05
R42 32 33 1.000E+06
R43 35 38 7.499E+05
*
.end
```

Table D.2: *SPICE deck of the evolved termperature sensing circuit shown in Figure 4.32 on page 171*

```
* Gaussian function problem
*
*
.dc Vc 2 3 0.01
.save dc I(Vl)
*
*
.options
+ABSTOL=10.0E-09
+VNTOL=100.E-06
+RELTOL=1.00E-04
+GMIN=1.00E-10
+ITL1=500
+ITL2=200
+ITL4=50
+ITL6=0
+METHOD=TRAP
+MAXORD=2
*
.MODEL NMOS_VAR_CH_W NMOS
.MODEL PMOS_VAR_CH_W PMOS
*
*
Vc 1 0 10
Rc 1 2 1
Vp 4 0 5
Vl 3 0 2.5
*
*
M1 5 6 7 0
+NMOS_VAR_CH_W
+L=1.000E-05
+W=1.000E-05
*
M2 8 9 10 0
+NMOS_VAR_CH_W
+L=1.000E-05
+W=1.000E-05
*
M3 11 12 13 4
+PMOS_VAR_CH_W
+L=1.000E-05
+W=1.000E-05
*
M4 14 15 16 4
+PMOS_VAR_CH_W
+L=1.000E-05
+W=1.000E-05
*
M5 17 18 19 4
+PMOS_VAR_CH_W
+L=1.000E-05
+W=1.000E-05
*
M6 20 21 22 4
+PMOS_VAR_CH_W
+L=1.000E-05
+W=1.000E-05
*
M7 23 24 25 4
+PMOS_VAR_CH_W
+L=1.000E-05
+W=1.000E-05
*
M8 26 27 28 4
+PMOS_VAR_CH_W
+L=1.000E-05
+W=1.000E-05
*
R1   0 14 7.499E+05
R2   0 20 1.000E+06
R3   2  4 1.000E+03
R6   2 15 5.623E+05
R4   2 20 2.371E+05
R5   2 24 5.623E+05
R8   3 12 1.000E+06
R9   3 21 1.334E+05
R10  3 27 7.499E+05
R7   4 19 7.499E+05
R11  5  8 1.000E+06
R12  6 12 4.217E+05
R13  7 19 1.334E+05
R14  8 21 1.000E+06
R26  9  0 1.000E+06
R15 10 17 3.162E+03
R16 11 18 4.217E+02
R17 11 26 7.499E+01
R18 13 18 4.217E+05
R19 13 28 1.000E+06
R20 14 17 7.499E+02
R21 16 21 1.000E+06
R22 18 22 7.499E+02
R23 19 25 3.162E+05
R24 20 28 1.000E+06
R25 21 23 7.499E+05
*
.end
```

Table D.3: *SPICE deck of the evolved Gaussian function generator circuit shown in Figure 4.39 on page 178*

Appendix E

Timings

Overview

This appendix gives some information about the actual evolution times for the examples of analog network evolution presented in Chapter 4.

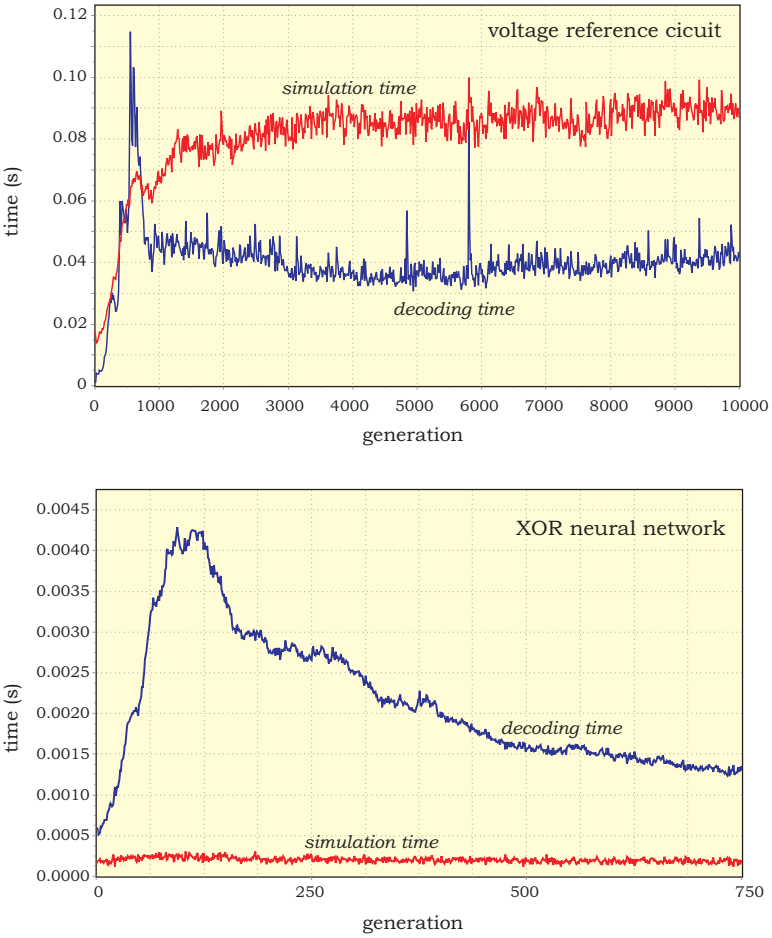


Figure E.1: *The timings of evolution of the voltage reference circuit reported in Figure 4.23 on page 160 (Experiment 11) and of the XOR function neural network reported in Figure 4.43 on page 183 (Experiment 14). The curves refer to the averages calculated over the 25 repetition of the experiment, of the population averages of the time required for the decoding of the genome, and the time required to simulate the decoded analog network. The evolutions were run on a PC with a Pentium 4 microprocessor clocked at 2.4 GHz.*

Bibliography

B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell*. Garland, New York, 4th edition, 2002.

E. Alm and A.P. Arkin. Biological networks. *Current Opinion in Structural Biology*, 13(2):193–202, 2003.

M.A. Anuta, D.W. Lozier, and P.R. Turner. The MasPar MP-1 as a computer arithmetic laborator. *J. Res. Natl. Inst. Stand. Technol.*, 101(2):165–174, 1996.

M.G. Arnold, T.A. Bailey, J.R. Cowles, and J.J. Cupal. Redundant logarithmic arithmetic. *IEEE Trans. Comput.*, 39(8):1077–1086, August 1990.

M.G. Arnold, T.A. Bailey, J.R. Cowles, and M.D. Winkel. Applying features of IEEE 754 to sign/logarithm arithmetic. *IEEE Trans. Comput.*, 41(8): 1040–1050, August 1992.

M.G. Arnold, T.A. Bailey, J.R. Cowles, and M.D. Winkel. Arithmetic co-transformations in the real and complex logarithmic number systems. *IEEE Trans. Comput.*, 47(7):777–7860, 1998.

J.C. Astor and C. Adami. A developmental model for the evolution of artificial neural networks. *Artificial Life*, 6(3):189–218, 2000.

J.-P. Aubin. Elements of viability theory for animat design. In J.-A. Meyer et al., editors, *From Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior*, pages 13–22, Boston, MA, 2000. MIT Press.

W. Banzhaf. Artificial regulatory networks and genetic programming. In R.L. Riolo and B. Worzel, editors, *Genetic Programming Theory and Practice*, pages 43–62. Kluwer, Boston, MA, 2003.

A.-L. Barabási and Z.N. Oltvai. Network biology: Understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, February 2004.

J.L. Barlow and E.H. Bareiss. On roundoff error distribution in floating point and logarithmic arithmetic. *Computing*, 34:325–347, 1985.

G Bateson. *Mind and Nature*. Fontana, London, 1979.

T. Bäck, editor. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, Oxford, 1996.

T. Bäck, D.B. Fogel, and Z. Michalewicz, editors. *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics, Bristol, 2000a.

T. Bäck, D.B. Fogel, and Z. Michalewicz, editors. *Evolutionary Computation 2: Advanced Algorithms and Operators*. Institute of Physics, Bristol, 2000b.

R.D. Beer. On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(4):469–509, 1995.

R.K. Belew. Interposing an ontogenic model between genetic algorithms and neural networks. In S.J. Hanson, J.D. Cowan, and C.L. Giles, editors, *Advances in Neural Information Processing, NIPS5*, pages 99–106. Morgan Kaufmann, San Mateo, CA, 1993.

R. Bellman. *Some vistas of modern mathematics*. University of Kentucky Press, 1968.

P.J. Bentley. Evolving fractal gene regulatory networks for robot control. In W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Zieglre, editors, *Proceedings of the European Conference on Artificial Life, ECAL 2003*, volume 2801 of *LNAI*, pages 753–762, Berlin, 2003. Springer.

P.J. Bentley. Fractal proteins. *Genetic Programming and Evolvable Machines*, 5(1):71–101, 2004.

U.S. Bhalla. Understanding complex signaling networks through models and metaphors. *Progress in Biophysics and Molecular Biology*, 81(1):45–65, 2003.

E.J.W. Boers and I.G. Sprinkhuizen-Kuyper. Combining biological metaphors. In M. Patel, V. Honavar, and K. Balakrishnan, editors, *Advances in the Evolutionary Synthesis of Intelligent Agents*, pages 153–183. MIT Press, Cambridge, MA, 2001.

J.C. Bongard. Evolving modular genetic regulatory networks. In *Proceedings of the IEEE 2002 Congress on Evolutionary Computation, CEC2002*, pages 1872–1877, Piscataway, NJ, 2002. IEEE Press.

J.T. Bonner. *The Evolution of Complexity by Means of Natural Selection*. Princeton University Press, Princeton, NJ, 1988.

J.M. Bower and H. Bolouri, editors. *Computational modeling of genetic and biochemical networks*. MIT Press, Cambridge, MA, 2001.

R.N. Brandon. *Adaptation and Environment*. Princeton University Press, Princeton, NJ, 1990.

D. Bray. Protein molecules as computational elements in living cells. *Nature*, 376:307–312, 27 July 1995.

R. Brooks. The relationship between matter and life. *Nature*, 409:409–411, 2001a.

R. Brooks. Steps towards living machines. In T. Gomi, editor, *Evolutionary Robotics: : From Intelligent Robotics to Artificial Life*, volume 2217 of *LNCS*, pages 72–93, Berlin, 2001b. Springer.

A. Cangelosi, D. Parisi, and S. Nolfi. Cell division and migration in a 'genotype' for neural networks. *Network: Computation in Neural Systems*, 5(4):497–515, November 1994.

J.N. Coleman. Simplification of table structure in logarithmic arithmetic. *Electronics Letters*, 31(22):1095–1096, 1995.

J.N. Coleman and J. Kadlec. Extended precision logarithmic arithmetic. In *Proceedings of the 34th Asilomar IEEE Conference on Signal, Systems and Computers*, pages 124–129, 2000.

J.N. Coleman, Chester E.I., C.I. Softley, and J. Kadlec. Arithmetic on the european logarithmic processor. *IEEE Trans. Comput.*, 49(7):702–715, 2000.

M. Conrad. Information processing in molecular systems. *Curr. Modern Biol.*, 5:1–14, 1972.

M. Conrad. The price of programmability. In R. Herken, editor, *The Universal Turing Machine: A Half Century Survey*. Kammer and Unverzagt Verlag, Hamburg, 1988.

M. Conrad. The geometry of evolution. *Biosystems*, 24:61–81, 1990.

M. Conrad. Molecular and evolutionary computation: the tug of war between context freedom and context sensitivity. *Biosystems*, 52:99–110, 1999.

M.J. Cooke. *Semiconductor Devices*. Prentice Hall, New York, 1990.

R. Courant and F. John. *Introduction to Calculus and Analysis. Volume II*. Springer, New York, 1989.

F.A.B. Coutinho, L.F. Lopez, and E. Massad. Comment on "The distribution of composite measurements: How to be certain of the uncertainties in what we measure," by M. P. Silverman, W. Strange, and T. C. Lipscombe [Am. J. Phys. 72 (8), 1068-1081 (2004)]. *Am. J. Phys.*, 72(12):1530, December 2004.

T.E. Creighton. *Proteins: Structures and Molecular Properties*. W.H. Freeman, New York, 2nd edition, 1993.

M. Crochemore and W. Rytter. *Jewels of Stringology*. World Scientific, Singapore, 2002.

M. Crochemore, C. Hancart, , and T. Lecroq. *Algorithmique du texte*. Vuibert, Paris, 2001.

L.J. Croft, M.J. Lercher, M.J. Gagen, and J.S. Mattick. Is prokaryotic complexity limited by accelerated growth in regulatory overhead? *Genome Biology*, 5(1), 2003.

C. Darwin. *On the origin of species by means of natural selection, or The preservation of favoured races in the struggle for life*. Murray, London, 1859.

E.D. de Jong, R.A. Watson, and J.B. Pollack. Reducing bloat and promoting diversity using multi-objective methods. In L. Spector et al., editors, *GECCO 2001*, pages 11–18, San Francisco, CA, 2001. Morgan Kaufmann.

K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, New York, 2001.

F. Dellaert and R.D. Beer. Toward an evolvable model of development for autonomous agent synthesis. In R. Brooks and P. Maes, editors, *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 246–257, Cambridge, MA, 1994. MIT Press.

F. Dellaert and R.D. Beer. A developmental model for the evolution of complete autonomous agents. In P. Maes, M.J. Mataric, J.-A. Meyer, J. Pollack, and S.W. Wilson, editors, *From Animals to Animats IV*, pages 393–401, Cambridge, MA, 1996. MIT Press.

B. Edmonds. What is complexity? - the philosophy of complexity per se with application to some examples in evolution. In F. Heylighen and D. Aerts, editors, *The Evolution of Complexity*, pages 1–16. Kluwer, Dordrecht, 1999.

P. Eggenberger. Cell interactions as a control tool of developmental processes for evolutionary robotics. In P. Maes, M.J. Mataric, J.-A. Meyer, J. Pollack, and S.W. Wilson, editors, *From Animals to Animats IV*, Cambridge, MA, 1996. MIT Press.

P. Eggenberger. Creation of neural networks based on developmental and evolutionary principles. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, *Proceedings of the International Conference on Artificial Neural Networks, ICANN'97, Lausanne, Switzerland, October 8-10, 1997*, volume 1327 of *LNCS*, pages 337–342, Berlin, 1997a. Springer.

P. Eggenberger. Evolving morphologies of simulated 3d organisms based on differential gene expression. In P. Husbands and I. Harvey, editors, *Proceedings of the 4th European Conference on Artificial Life, ECAL97*, pages 205–213, Cambridge, MA, 1997b. MIT Press.

P. Eggenberger. Combining developmental processes and their physics in an artificial evolutionary system to evolve shapes. In P.J. Bentley S. Kumar, editor, *On Growth, Form and Computers*, pages 302–318. Academic Press, London, 2003.

P. Eggenberger. Comparing direct and developmental encoding schemes in artificial evolution: A case study in evolving lens shapes. In P. Maes, M.J. Mataric, J.-A. Meyer, J. Pollack, and S.W. Wilson, editors, *Proceedings of the Congress on Evolutionary Computation, CEC 2004*. IEEE Press, 2004.

P. Eggenberger, G. Gómez, and R. Pfeifer. Evolving the morphology of a neural network for controlling a foveating retina – and its test on a real robot. In R.K. Standish, M.A. Bedau, and H.A. Abbass, editors, *Artificial Life VIII*, pages 243–251, Cambridge, MA, 2002. MIT Press.

A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer, Berlin, 2003.

D. Floreano and F. Mondada. Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics (Part B)*, 26(3): 396–407, 1996.

D.B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, 2md edition, 2000.

W. Fontana and P. Schuster. Continuity in evolution: On the nature of transitions. *Science*, 280:1451–1455, May 1998a.

W. Fontana and P. Schuster. Shaping space: the possible and the attainable in RNA genotype-phenotype mapping. *J. theor. Biol.*, 194:491–515, 1998b.

J. Gall. *Systemantics: The Underground Text of Systems Lore*. The General Systemantics Press, Ann Arbor, MI, 2nd edition, 1986.

N. Geard and J. Wiles. Structure and dynamics of a gene network model incorporating small RNAs. In R. Sarker, R. Reynolds, H. Abbass, K.-C. Tan, R. McKay, D. Essam, and T. Gedeon, editors, *Proceedings of the 2003 Congress on Evolutionary Computation, CEC2003*, pages 199–206, Piscataway, NJ, 2003. IEEE Press.

D.T. Gillespie. A theorem for physicists in the theory of random variables. *Am. J. Phys.*, 51(6):520–533, June 1983.

D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

D.E. Goldberg. *The Design of Innovation*. Kluwer, Boston, MA, 2002.

D.J. Goodman and A. Gersho. Theory of an adaptive quantizer. *IEEE Transactions on Communications*, 22(8):1037–1045, August 1974.

D. Graur and W.-H. Li. *Fundamentals of Molecular Evolution*. Sinauer Associates, Sunderland, MA, 2nd edition, 2000.

R.M. Gray and D.L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, October 1998.

M. Grene. Hierarchies in biology. *American Scientist*, 75:504–510, September-October 1987.

J.B. Grimbleby. Automatic analogue network synthesis using genetic algorithms. In *Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems, GALESIA 95, Sheffield, 12-14 September 1995*, pages 53–58, London, 1995. IEE.

F. Gruau. *Neural Network Synthesis Using Cellular Encoding And The Genetic Algorithm*. PhD thesis, Ecole Normale Supérieure de Lyon, 1994.

F. Gruau. Automatic definition of modular neural networks. *Adaptive Behaviour*, 3(2):151–183, 1995a.

F. Gruau. Genetic programming of neural networks: Theory and practice. In S. Goonatilake and S. Khebbal, editors, *Intelligent Hybrid Systems*, pages 245–271. Wiley, New York, 1995b.

G. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, Cambridge, 1997.

J. Hallinan and J. Wiles. Evolving genetic regulatory networks using an artificial genome. In Y.-P. Chen, editor, *Proceedings of the 2nd Asia-Pacific Bioinformatics Conference, APBC2004, Dunedin, New Zealand.*, pages 291–296. Australian Computer Society, 2004.

R.W. Hamming. On the distribution of numbers. *Bell Syst. Tech. J.*, 48(8): 1609–1625, October 1970.

F.M. Harold. *The Way of the Cell*. Oxford University Press, Oxford, 2001.

I. Harvey. *The Artificial Evolution of Adaptive Behaviour*. PhD thesis, University of Sussex, 1995.

I. Harvey. Cognition is not computation: evolution is not optimisation. In *Proceedings of ICANN97, 7-10 October 1997, Lausanne, Switzerland*, Berlin, 1997. Springer-Verlag.

I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi. Evolutionary robotics: the Sussex approach. *Robotics and Autonomous Systems*, 20 (2-4):205–224, 1997.

N. Haspel, C.-J. Tsai, H. Wolfson, and R. Nussinov. Reducing the computational complexity of protein folding via fragment folding and assembly. *Protein Science*, 12(6):1177–1187, 2003.

S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, Upper Saddle River, NY, 2nd edition, 1999.

M. Hiratsuka, T. Aoki, and T. Higuchi. Enzyme transistor circuits for reaction-diffusion computing. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 46(2):294–303, February 1999.

R.F. Holt, T.F. Braumandl, and D.J. Mackillop. An index of old-growthness for two BEC variants in the Nelson Forest Region. Technical report, University of British Columbia, Faculty of Forestry, Vancouver, April 1999.

L.D. Hurst, C. Pál, and M.J. Lercher. The evolutionary dynamics of eukaryotic gene order. *Nature Review Genetics*, 5(4):299–310, April 2004.

E. Jablonka and M.J. Lamb. *Epigenetic inheritance and evolution*. Oxford University Press, Oxford, 1995.

N. Jakobi. Harnessing morphogenesis. In P.J. Bentley S. Kumar, editor, *On Growth, Form and Computers*, pages 392–404. Academic Press, London, 2003.

N.S. Jayant and P. Noll. *Digital coding of waveforms: principles and applications to speech and video*. Prentice-Hall, Englewood Cliffs, NJ, 1984.

E.T. Jaynes. Bayesian methods: general background. In H. Justice, editor, *Maximum Entropy and Bayesian Methods in Applied Statistics*, pages 1–25. Cambridge University Press, Cambridge, 1986.

E.T. Jaynes. Probability theory as logic. In P.F. Fougère, editor, *Maximum Entropy and Bayesian Methods*, pages 1–16. Kluwer, Dordrecht, 1990.

E.T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, Cambridge, 2003.

H. Jeffreys. *Theory of Probability*. Oxford University Press, Oxford, 3rd edition, 1961.

S.A. Kassam. Quantization based on the absolute-error criterion. *IEEE Transactions on Communications*, 26(2):267–270, February 1978.

M. Keijzer. Efficiently representing populations in genetic programming. In P. Angeline et al., editors, *Advances in Genetic Programming*, volume 2, chapter 13, pages 259–278. MIT Press, Cambridge, MA, 1996.

R.E. Keller and W. Banzhaf. Explicit maintenance of genetic diversity on genospaces. Unpublished manuscript. Available online at Citeseer, 1994.

P.J. Kennedy and T.R. Osborn. Operon expression and regulation with spiders. In *GECCO 2000*, pages 161–166, San Francisco, CA, 2000. Morgan Kaufmann.

P.J. Kennedy and T.R. Osborn. A model of gene expression and regulation in an artificial cellular organism. *Complex Systems*, 13(1), 2001.

M. Kirschner and J. Gerhart. Evolvability. *Proc. Natl. Acad. Sci. USA*, 95(15): 8420–8427, July 1998.

J. Kitagawa and H. Iba. Identifying metabolic pathways and gene regulation networks with evolutionary algorithms. In G.B. Fogel and D.W. Corne, editors, *Evolutionary Computation in Bioinformatics*, chapter 12, pages 255–278. Morgan Kaufmann, San Francisco, CA, 2002.

H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4(4):461–476, 1990.

K. Kobayashi and M. Ohbayashi. A new indirect encoding method with variable length gene code to optimize neural network structures. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 99, 10-16 July 1999*, pages 4409–4412, Piscataway, NJ, 1999. IEEE Press.

J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992.

J.R. Koza, F.H. Bennet III, D. Andre, and M.A. Keane. *Genetic Programming III*. Morgan Kaufmann, San Francisco, CA, 1999.

J.R. Koza, M.A. Keane, M.J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer, Norwell, MA, 2003.

G. Krauss. *Biochemistry of Signal Transduction and Regulation*. Wiley-VCH, Weinheim, 3rd edition, 2003.

W. Kruiskamp and D. Leenaerts. DARWIN: CMOS opamp synthesis by means of a genetic algorithm. In *Proceedings of the 32nd ACM/IEEE conference on Design automation, June 12-16, 1995, San Francisco, CA*, pages 433–438, New York, 1995. ACM Press.

S. Kumar and P.J. Bentley. Biologically inspired evolutionary development. In A. Tyrrell, P. Haddow, and J. Torresen, editors, *Proceedings of the 5th International Conference on Evolvable Systems: From Biology to Hardware, ICES03*, volume 2606 of *LNCS*, pages 57–68, Berlin, 2003. Springer.

R. Laing. Automaton models of reproduction by self-inspection. *J. theor. Biol.*, 66:437–456, 1977.

W.B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer, Berlin, 2002.

C.G. Langton. Self-reproduction in cellular automata. *Physica D*, 10(1-2): 135–144, January 1984.

J.G. Lawrence and J.R. Roth. Selfish operons: Horizontal transfer may drive the evolution of gene clusters. *Genetics*, 143:1843–1860, 1996.

Y. Leung, Y. Gao, and Z.B. Xu. Degree of population diversity - a perspective on premature convergence in genetic algorithms and its Markov chain analysis. *IEEE Trans. Neural Networks*, 8(5):1165–1176, 1997.

M. Levandowsky and D. Winter. Distance between sets. *Nature*, 234(5): 34–35, 1971.

B. Lewin. *Genes VIII*. Prentice-Hall, Upper Saddle River, NJ, 2004.

R.C. Lewontin. Adaptation. *Scientific American*, 239(9):156–169, 1974.

R.C. Lewontin. Natural history and formalism in evolutionary genetics. In R.S. Singh, C.B. Krimbas, D.B. Paul, and J. Beatty, editors, *Thinking About Evolution*, volume 2, pages 7–20. Cambridge University Press, Cambridge, 2001.

A.H. Lipkus. A proof of the triangle inequality for the Tanimoto distance. *J. of Mathematical Chemistry*, 26:263–265, 1999.

S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982.

J.D. Lohn and S.P. Colombano. A circuit representation technique for automated circuit design. *IEEE Transactions on Evolutionary Computation*, 3 (3):205–219, September 1999.

M.A. Lones. *Enzyme Genetic Programming: Modelling Biological Evolvability in Genetic Programming*. PhD thesis, University of York, 2004.

M.A. Lones and A.M. Tyrrell. Modelling biological evolvability: implicit context and variation filtering in enzyme genetic programming. *BioSystems*, 76(1-3):229–238, August-October 2004a.

M.A. Lones and A.M. Tyrrell. Enzyme genetic programming. In M. Amos, editor, *Cellular Computingt*, pages 19–42. Oxford University Press, Oxford, 2004b.

P.C. Marijuán. Gloom in the society of enzymes: on the nature of biological information. *BioSystems*, 38(2-3):163–171, 1996.

J.S. Mattick. Introns: evolution and function. *Current Opinion in Genetics & Development*, 4(6):823–831, 1994.

J.S. Mattick. Non-coding RNAs: the architects of eukaryotic complexity. *EMBO Reports*, 2(11):986–991, 2001.

J.S. Mattick and M.J. Gagen. The evolution of controlled multitasked gene networks: The role of introns and other noncoding RNAs in the development of complex organisms. *Mol. Biol. Evol.*, 18(9):1611–1630, 2001.

C. Mattiussi and D. Floreano. Elimination and extinction in evolutionary computation. Unpublished manuscript, 2003.

C. Mattiussi and D. Floreano. Evolution of analog networks using local string alignment on highly reorganizable genomes. In R.S. Zebulum et al., editors, *Proceedings of the 2004 NASA/DoD Conference on Evolvable Hardware, 24-26 June 2004, Seattle*, pages 30–37, Los Alamitos, CA, 2004a. IEEE Computer Society.

C. Mattiussi and D. Floreano. Connecting transistors and proteins. In J. Pollack, M. Bedau, P. Husbands, T. Ikegami, and R. A. Watson, editors, *ALife9: Proceedings of the Ninth International Conference on Artificial Life*, pages 9–14, Boston, MA, 2004b. MIT Press.

C. Mattiussi, M. Waibel, and D. Floreano. Measures of diversity for populations and distances between individuals with highly reorganizable genomes. *Evolutionary Computation*, 12(4):495–515, 2004.

J. Max. Quantizing for minimum distortion. *IEEE Transactions on Information Theory*, 6(1):7–12, March 1960.

J. Maynard-Smith. The major transitions in evolution. In G.A. Cowan, D. Pines, and D. Meltzer, editors, *Complexity: Metaphors, Models, and Reality*, pages 457–470. Addison Wesley, Reading, MA, 1994.

J. Maynard-Smith. *Evolutionary Genetics*. Oxford University Press, Oxford, 2nd edition, 1998.

E. Mayr. *What Evolution Is*. Basic Books, New York, 2001.

E.M. McCreight. A space-economical suffix tree construction algorithm. *J. ACM*, 23(1):262–272, 1976.

B. McMullin. *Artificial Knowledge: An Evolutionary Approach*. PhD thesis, University College Dublin, Department of Computer Science, 1992.

B. McMullin. John von Neumann and the evolutionary growth of complexity: Looking backward, looking forward. *Artificial Life*, 6(4):347–361, 2000.

B. McMullin, T. Taylor, and A. von Kamp. Who needs genomes? In *Atlantic Symposium on Computational Biology, Genome Information Systems and Technology (CBGI), March 2001, Duke University, USA*, 2001.

D. McShea. Complexity and evolution: what everybody knows. *Biology and Philosophy*, 6(3):303–324, 1991.

G. Meister and T. Tuschl. Mechanisms of gene silencing by double-stranded rna. *Nature*, 431:343–349, 16 September 2004.

D. Metzgar and C. Wills. Evidence for the adaptive evolution of mutation rates. *Cell*, 101:581–584, 2000.

Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin, 3rd edition, 1996.

J.G. Miller. *Living Systems*. McGraw-Hill, New York, 1978.

G.P. Monro. The concept of multiset. *Zeitschr. f. math. Logik ung Grundlagen d. Math.*, 33:171–178, 1987.

W. Morrison and K.A. De Jong. Measurement of population diversity. In P. Collet et al., editors, *EA 2001*, volume 2310 of *LNCS*, pages 31–41, Berlin, 2002. Springer.

D.W. Mount. *Bioinformatics: Sequence and Genome Analysis*. Cold Spring Harbor Lab. Press, Cold Spring Harbor, New York, 2001.

S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.

S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, Cambridge, MA, 2000.

S. Nolfi and D. Parisi. 'genotypes' for neural networks. In M.A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 431–434. MIT Press, Cambridge, MA, 1995.

S. Ohno. *Evolutionary by Gene Duplication*. Springer, Berlin, 1970.

U.-M. O'Reilly. Using a distance metric on genetic programs to understand genetic operators. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 5, pages 4092– 4097, 1997.

P.F. Panter and W. Dite. Quantization distortion in pulse count modulation with nonuniform spacing of levels. *Proc. IRE,*, 39:44–48, January 1951.

H.H. Pattee. The physical basis and origin of hierarchical control. In H.H. Pattee, editor, *Hierarchy Theory: The Challenge of Complex Systems*, pages 73–108. G. Braziller, New York, 1973.

H.H. Pattee. Symbol-structure complementarity in biological evolution. In E. Jantsch, editor, *The evolutionary vision*, pages 117–128. Westview Press, Boulder, CO, 1981.

H.H. Pattee. Artificial life needs a real epistemology. In F. Moran et al., editors, *Advances in Artificial Life*, pages 23–38. Springer, Berlin, 1995a.

H.H. Pattee. Evolving self-reference: matter, symbols, and semantic closure communication and cognition. *Artificial Intelligence*, 12(1-2):9–27, 1995b.

H.H. Pattee. The physics of symbols: bridging the epistemic cut. *BioSystems*, 60(1-3):5–21, 2001.

L. Pietronero, E. Tosatti, V. Tosatti, and A. Vespignani. Explaining the uneven distribution of numbers in nature: the laws of Benford and Zipf. *Physica A*, 293:297–304, 2001.

A. Pires-daSilva and R.J. Sommer. The evolution of signalling pathways in animal development. *Nature Reviews Genetics*, 4(1):39–49, January 2003.

M. Polanyi. Life's irreducible structure. *Science*, 160:1308–1312, June 1968.

K.R. Popper. *Quantum theory and the schism in physics*. Hutchinson, London, 1982. Postscript to the Logic of scientific discovery, Vol. 3, Edited by W.W.Bartley III.

W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, 2nd edition, 1992.

M. Ptashne and A. Gann. Transcriptional activation by recruitment. *Nature*, 386:569–577, 10 April 1997.

M. Ptashne and A. Gann. Imposing specificity by localization: mechanism and evolvability. *Current Biology*, 8(22):R812–R822, 5 November 1998.

M. Ptashne and A. Gann. *Genes & Signals*. Cold Spring Harbor Lab. Press, Cold Spring Harbor, New York, 2002.

J.C.F. Pujol and R. Poli. Evolving the topology and the weights of neural networks using a dual representation. *Applied Intelligence Journal*, 8(1): 73–84, 1998.

T. Quarles, A.R. Newton, D.O. Pederson, and A. Sangiovanni-Vincentelli. *SPICE3 Version 3f4 User's Manual*. University of California, Berkeley, CA, 1989.

T. Quick, C. L. Nehaniv, K. Dautenhahn, and G. Roberts. Evolving embodied genetic regulatory network-driven control systems. In W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Zieglre, editors, *Proceedings of the European Conference on Artificial Life, ECAL 2003*, volume 2801 of *LNAI*, pages 266–277, Berlin, 2003. Springer.

R.A. Raimi. The first digit problem. *American Mathematical Monthly*, 83: 521–538, 1976.

T. Reil. Dynamics of gene expression in an artificial genome - implications for biological and artificial ontogeny. In D. Floreano, F. Mondada, and J.D. Nicoud, editors, *Proceedings of the 5th European Conference on Artificial Life*, Berlin, 1999. Springer Verlag.

T. Reil. Artificial genomes as models of gene regulation. In P.J. Bentley S. Kumar, editor, *On Growth, Form and Computers*, pages 256–277. Academic Press, London, 2003.

U.L. Rosewich and H.C. Kistler. Role of horizontal gene transfer in the evolution of fungi. *Annu. Rev. Phytopathol.*, 38:25–63, 2000.

F. Rothlauf. *Representations for Genetic and Evolutionary Algorithms*. Physica-Verlag, Heidelberg, 2002.

F. Rothlauf and D.E. Goldberg. Redundant representations in evolutionary computation. *Evolutionary Computation*, 11(4):381–415, 2003.

C. Salzberg and H. Sayama. Complex genetic evolution of artificial self-replicators in cellular automata. *Complexity*, 10(2):33–39, November-December 2004.

C. Salzberg, A. Antony, and H. Sayama. Complex genetic evolution of self-replicating loops. In *ALife9: Proceedings of the Ninth International Conference on Artificial Life*, pages 262–267, Boston, MA, 2004. MIT Press.

D. Sankoff and J.B. Kruskal. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, Reading, MA, 1983.

N.N. Schraudolph and R.K. Belew. Dynamic parameter encoding for genetic algorithms. *Machine Learning*, 9(1):9–21, 1992.

P. Schuster. How does complexity arise in evolution. *Complexity*, 2(1):22–30, 1996.

P. Schuster. The disaster of central control: An impressive example from nature. *Complexity*, 9(4):13–14, 2004.

C.E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27:379–423, 623–656, July, October 1948.

J.A. Shapiro. A 21st century view of evolution. *J. Biol. Phys.*, 28(4):745–764, 2002.

D.B Sherman et al. Genome evolution in yeasts. *Nature*, 430:35–44, 1 July 2004.

M.P. Silverman, W. Strange, and T.C. Lipscombe. The distribution of composite measurements: How to be certain of the uncertainties in what we measure. *Am. J. Phys.*, 72(8):1068–1081, August 2004.

M.L. Simpson, C.D. Cox, and G.S. Sayler. Frequency domain analysis of noise in autoregulated gene circuits. *Proc. Natl. Acad. Sci. USA*, 100(8): 4551–4556, 15 April 2003.

M.L. Simpson, C.D. Cox, G.D. Peterson, and G.S Sayler. Engineering in the biological substrate: Information processing in genetic circuits. *Proceedings of the IEEE*, 92(5):848–863, May 2004a.

M.L. Simpson, G.S. Sayler, J.T. Fleming, J. Sanseverino, and C.D. Cox. The device science of whole cells as components in microscale and nanoscale systems. In M. Amos, editor, *Cellular Computingt*, pages 74–106. Oxford University Press, Oxford, 2004b.

M. Sipper, E. Sanchez, D. Mange, M. Tomassini, A. Pérez-Uribe, and A. Stauffer. A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems. *IEEE Transactions on Evolutionary Computation*, 1(1):83–97, April 1997.

D. S. Sivia. *Data Analysis: A Bayesian Tutorial*. Oxford University Press;, Oxford, 1996.

B. Smith. Instantaneous companding of quantized signals. *Bell Syst. Tech. J.*, 36(3):653–709, May 1957.

T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.

B.M. R. Stadler, P. Stadler, G. Wagner, and W. Fontana. The topology of the possible: Formal spaces underlying patterns of evolutionary change. *J. theor. Biol.*, 213(2):241–274, November 2001.

K.O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

K.O. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21: 63–100, 2004.

E.E. Swartzlander and A.G. Alexopoulos. The sign/logarithm number system. *IEEE Trans. Comput.*, 24(12):1238–1242, 1975.

J.R. Taylor. *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements*. University Science Books, Sausalito, CA, 2nd edition, 1996.

T. Taylor. Creativity in evolution: individuals, interactions, and environments. In P.J. Bentley and D.W. Corne, editors, *Creative evolutionary systems*, pages 79–108. Morgan Kaufmann, San Francisco, CA, 2001.

T. Taylor. Redrawing the boundary between organism and environment. In J. Pollack, M. Bedau, P. Husbands, T. Ikegami, and R. A. Watson, editors, *ALife9: Proceedings of the Ninth International Conference on Artificial Life*, pages 268–273, Boston, MA, 2004. MIT Press.

S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, Sand Diego, CA, 2nd edition, 2003.

T. Toffoli. Occam, Turing, von Neumann, Jaynes: How much can you get for how little? In *Proceedings of ACRI'94*, pages 1–9, Berlin, 1994. Springer.

M. Tomassini, L. Vanneschi, F. Fernández, and G. Galeano. A study of diversity in multipopulation genetic programming. In Pierre Liardet et al., editors, *EA 2003, Artificial Evolution: 6th International Conference, Marseilles, France, October 27-30, 2003*, volume 2936 of *LNCS*, pages 243–255, Berlin, 2004. Springer.

E.N. Trifonov. Making sense of the human genome. In R.H. Sarma and M.H. Sarma, editors, *Structure and Methods*, volume 1, pages 69–77. Adenine Press, New York, 1990.

O.G. Troyanskaya, Y. Arbell, O.and Koren, G.M. Landau, and A. Bolshoy. Sequence complexity of prokariotic genomic sequences: A fast algorithm for calculating linguistic complexity. *Bioinformatics*, 18(5):679–688, 2002.

A.V. Trushkin. Sufficient conditions for uniqueness of a locally optimal quantizer for a class of convex error weighting functions. *IEEE Transactions on Information Theory*, 28(2):187–198, March 1982.

E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14:249–260, 1995.

A. Vladimirescu. *The SPICE Book*. Wiley, New York, 1994.

J. Vohradský. Neural network model of gene expression. *FASEB Journal*, 15:846–854, March 2001.

V.L. Volkov and P.V. Pakshin. The logarithmic number system in control algorithms and information processing. *Soviet Journal of Computer and Systems Sciences*, 30(1):132–138, 1992.

J. von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Urbana, IL, 1966. Edited and completed by A.W. Burks.

G.P. Wagner and L. Altenberg. Complex adaptations and the evolution of evolvability. *Evolution*, 50(3):967–976, June 1996.

J. Watson, J. Wiles, and J. Hanan. Towards more relevant evolutionary models: Integrating an artificial genome with a developmental phenotype. In H.A. Abbass and J. Wiles, editors, *Proceedings The Australian Conference on Artificial Life, ACAL 2003*, pages 288–298, 2003a.

J.D. Watson, T.A. Baker, S.P. Bell, A. Gann, M. Levine, and R. Losick. *Molecular Biology of the Gene*. Benjamin-Cummings, San Francisco, CA, 5th edition, 2003b.

R. Weiss, S. Basu, S. Hooshangi, A. Kalmbach, D. Karig, R. Mehreja, and I. Netravali. Genetic circuit building blocks for cellular computation, communications, and signal processing. *Natural Computing*, 2(1):47–84, 2003.

G.R. Welch. The enzymatic basis of information processing in the living cell. *Biosystems*, 38(2-3):147–153, 1996.

D.J. Whitehead, A. Skusa, and P.J. Kennedy. Evaluating an evolutionary approach for reconstructing gene regulatory networks. In J. Pollack, M. Bedau, P. Husbands, T. Ikegami, and R. A. Watson, editors, *ALife9: Proceedings of the Ninth International Conference on Artificial Life*, pages 427–432. MIT Press, 2004.

D. Whitley, T. Starkweather, and C. Bogart. Genetic algorithms and neural networks: optimizing connections and connectivity. *Parallel Computing*, 14 (3):347–361, August 1990.

M Wineberg and F. Oppacher. Enhancing the GA's ability to cope with dynamic environments. In D. Whitley et al., editors, *GECCO 2000*, pages 3–10, San Francisco, CA, 2000. Morgan Kaufmann.

M. Wineberg and F. Oppacher. Distance between populations. In E. Cantú-Paz et al., editors, *GECCO 2003*, volume 2724 of *LNCS*, pages 1481–1492, Berlin, 2003a. Springer.

M. Wineberg and F. Oppacher. The underlying similarity of diversity measures used in evolutionary computation. In E. Cantú-Paz et al., editors, *GECCO 2003*, volume 2724 of *LNCS*, pages 1493–1504, Berlin, 2003b. Springer.

D.H. Wolpert and W.G. Macready. No Free Lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9): 1423–1447, September 1999.

L.K. Yu and D.M. Lewis. A 30-b integrated logarithmic number system processor. *IEEE J. Solid-State Circuits*, 26(10):1433–1440, 1991.

R.S. Zebulum, M.A. Pacheco, and M. Vellasco. Evolving control metabolisms for a robot. *Evolutionary Computation*, 8(1):93–120, Spring 2000.

R.S. Zebulum, M.A.C Pacheco, and M.M.B.R Vellasco. *Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms*. CRC Press, Boca Raton, FL, 2002.

J. Ziegler and W. Banzhaf. Evolving control metabolisms for a robot. *Artificial Life*, 7:171–190, 2001.

L. Zinchenko, H. Mühlenbein, V. Kureichik, and T. Mahnig. A comparison of different circuit representations for evolutionary analog circuit design. In A.M. Tyrrell, P.C. Haddow, and J. Torresen, editors, *ICES 2003*, volume 2606 of *LNCS*, pages 13–23, Berlin, 2003. Springer.

Index

Curriculum Vitae

Claudio Mattiussi

Personal and professional information

I was born on October 31, 1965, in Elisabethville, Congo. I graduated in 1984 in electronics from Istituto Tecnico Industriale Arturo Malignani, Udine, Italy, with the grade of 60/60. From 1990 to 1992 I worked on and off at ANCIFAP, Trieste, Italy, teaching training courses in electronics. From March 1992 to March 1993 I served my civil service at Associazione La Nostra Famiglia, San Vito al Tagliamento, Pordenone, Italy, which provides rehabilitation and professional training for young disabled. In December 1993 I graduated in electronic engineering with focus on telecommunications from Università degli Studi di Trieste, Italy, with the grade of 110/110 summa cum laude. My thesis project was performed under the supervision of Prof. Enzo Tonti in the field of numerical methods for electromagnetics. From 1994 to June 1995 I developed industrial process control and monitoring software at Studio tecnico Zantoni, Maiano, Udine, Italy. From June 1995 to December 2000 I managed the R&D activities at the Non Ionizing Radiation Laboratory (NIRLAB) of Clampco Sistemi, Romans d'Isonzo, Gorizia, Italy, located in Trieste AREA Science Park, where I was active in the conception and engineering of electronic and electro-optical RF measurement and antenna systems. In January 2001 I joined Dario Floreano's Evolutionary and Adaptive Systems Team (now Laboratory of Intelligent Systems) at the Ecole Polytechnique Fédérale de Lausanne (EPFL) as a research assistant, and in December 2001 I submitted my PhD research proposal in the field of evolutionary computation, which culminates in the present thesis.

Publications

1. C. Mattiussi, M. Waibel, and D. Floreano (2004) Measures of Diversity for Populations and Distances Between Individuals with Highly Reorganizable Genomes. *Evolutionary Computation*, 12 (4), pp. 495-515.

2. C. Mattiussi, and D. Floreano (2004) Connecting Transistors and Proteins. In *ALife9: Proceedings of the Ninth International Conference on Artificial Life*, 12-15 September 2004, Boston (MA), MIT Press, pp. 9-14.

3. C. Mattiussi, and D. Floreano (2004) Evolution of Analog Networks using Local String Alignment on Highly Reorganizable Genomes. In *Proceedings of the 2004 NASA/DoD Conference on Evolvable Hardware*, 24-26 June 2004, Seattle, Zebulum, R.S. et al. (eds.), IEEE Press. pp. 30-37.

4. D. Floreano, J.C. Zufferey, and C. Mattiussi (2003) Evolving Spiking Neurons from Wheels to Wings. In *Dynamic Systems Approach for Embodiment and Sociality*, K. Murase and T. Asakura (eds.), Magill, Advanced Knowledge International, International Series on Advanced Intelligence, Vol 6, pp. 65-70.

5. D. Roggen, D. Floreano, and C. Mattiussi (2003) A Morphogenetic Evolutionary System: Phylogenesis of the POEtic Circuit. In *Proceedings of the Fifth International Conference on Evolvable Systems*, Tyrrell, A. M. and Haddow, P. C. and Torresen, J. (eds.), pp 153-164.

6. D. Floreano, and C. Mattiussi (2002). *Manuale sulle reti neurali*, seconda edizione, Bologna: Il Mulino.

7. C. Mattiussi (2002). Review of: "Digital Biology: The Creation of Life Inside Computers and How It Will Affect Us" by Peter J. Bentley. *Artificial Life* 8: 379-382.

8. C. Mattiussi (2002). A Reference Discretization Strategy for the Numerical Solution of Physical Field Problems, *Advances in Imaging and Electron Physics* 121, pp. 143-279.

9. D. Floreano, and C. Mattiussi (2001). Evolution of Spiking Neural Controllers for Autonomous Vision-based Robots. In *Evolutionary Robotics IV*, T. Gomi (ed.), Berlin: Springer-Verlag.

10. C. Mattiussi (2001). The Geometry of Time-Stepping, *Progress in Electromagnetics Research* 32, pp. 123-149.

11. C. Mattiussi (2000). The Finite Volume, Finite Difference, and Finite Elements Methods as Numerical Methods for Physical Field Problems, *Advances in Imaging and Electron Physics,* 113, pp. 1-146.

12. C. Mattiussi (1998). Edge Elements and Cochain-Based Field Function Approximation. In *Proceedings of the 4th International Workshop on Electric and Magnetic Fields*, Marseille (France), pp. 301-306.

13. C. Mattiussi (1997). An Analysis of Finite Volume, Finite Element, and Finite Difference Methods Using Some Concepts from Algebraic Topology, *J. Comput. Phys.* 133, pp. 289-309.