Stefano Cagnoni · Marco Mirolli
Marco Villani *Editors*

Evolution, Complexity and Artificial Life

Springer

Evolution, Complexity and Artificial Life

Stefano Cagnoni • Marco Mirolli • Marco Villani
Editors

Evolution, Complexity and Artificial Life

Springer

Editors
Stefano Cagnoni
Department of Information Engineering
University of Parma
Parma
Italy

Marco Mirolli
Consiglio Nazionale delle Ricerche
Istituto di Scienze e Tecnologie della Cognizione
Rome
Italy

Marco Villani
Facoltà di Scienze della Comunicazione
University of Modena and Reggio Emilia
Reggio Emilia
Italy

ISBN 978-3-642-37576-7 ISBN 978-3-642-37577-4 (eBook)
DOI 10.1007/978-3-642-37577-4
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013957138

Preface

Traditionally, artificial evolution, complex systems, and artificial life were separate fields, each with its own research community, but we are now seeing increased intertwinement and hybridization among them. It is now hard to imagine any work in one of these areas that does not refer to techniques or theoretical results normally considered to belong to one of the other two.

Evolution and complexity characterize both biological life and artificial life, whenever direct modeling of biological processes is pursued or populations of interacting artificial biologically inspired entities are created, from which complex behaviors can emerge and evolve.

The latter consideration, besides offering a proof of the tight connections existing between these disciplines, also gives an idea of the breadth of the related topics of interest, and of the different study viewpoints, ranging from purely scientific and exploratory approaches aimed at verifying biological theories to technology-focused applied research aimed at solving difficult real-world problems raised by practical and industrial tasks.

As a result of the hybridization between these disciplines, the same is happening to the corresponding research communities, and common conference tracks and workshops are being organized worldwide.

We conceived the idea of editing a book to collect contributions offering a wide panoramic view of the opportunities that cooperation among the three disciplines can produce when we started organizing the fourth edition of WIVACE, the Italian Workshop on Evolutionary Computing and Artificial Life, which was held in Parma in February 2012.

This edited book includes invited chapters from leading scientists in the fields of artificial life, complex systems, and evolutionary computing, aimed at authoritatively introducing readers to some of the main research topics that are not only shared by the three research fields, but that, in some cases, can only exist, thanks to the contribution of all three disciplines.

The book also contains a selection of the best papers presented at WIVACE 2012, thoroughly revised and extended by the authors.

The contributions, ranging from fundamental theoretical issues to state-of-the-art real-world applications, have been organized into five parts, based either on their kind (research-oriented or application-oriented) or topics (biological modeling, mind and society, evolution).

This subdivision denotes how the modeling of biological systems, both anatomical-functional and social, constitutes the wide general field in which evolution, complexity, and artificial life coexist most closely and which is studied most frequently.

This is not surprising, as artificial life and evolution are biologically inspired disciplines per se, while it is hard to imagine something which is more complex than the functioning of the human body and, possibly even more so, human society.

In the following we provide a brief overview of each chapter. On one hand, to allow readers to figure out a general picture of the composite research field induced by the interactions of the three disciplines, and, on the other hand, to let readers quickly locate the chapters which look most interesting to them.

Part I: Research Issues

The first chapter, by Domenico Parisi, discusses the challenges posed by constructing embodied artificial agents (i.e., robots) not as practical applications but as a means to understand human behavior. Since there are many ways to construct an artifact that reproduces a single (human) phenomenon, the author argues that robots constructed as theories of behavior should attempt to reproduce many different phenomena at the same time, as the more phenomena an artifact reproduces, the more likely it is that the artifact actually explains reality rather than being a "toy." Hence, Parisi discusses a number of different phenomena that robots should try to reproduce in order to explain human behavior and that are currently underinvestigated in robotics and artificial life.

The second chapter, by Riccardo Poli and Christopher R. Stephens, deals with the difficult problem of building a theory of evolutionary systems with which we can understand and predict natural evolutionary dynamics. In particular, as a means to develop such a theory, their chapter proposes a technique originally used in statistical physics named coarse graining. The method consists of finding a set of collective variables that may offer a computationally feasible description of a system that has too many degrees of freedom to be analyzed. After describing the general technique, the authors show how to apply it to evolutionary systems in order to describe and understand their complex dynamics.

Part II: Biological Modeling

This part of the book examines the important issue of the organization principles of living systems. In particular, the main focus is on the organization of the relationships among the parts inside living beings: complementary to the reductionist and physical approaches, this kind of strategy searches for the structure of feedbacks that constitute the organizational processes of living beings, and that drive and channel their future changes, i.e., their evolution. In doing so, signal exchange and noise play a fundamental role: what these chapters interestingly show is that these drivers do not counteract each other but rather they integrate in order to allow the emergence of recurrent patterns of activation that, in turn, are an important part of the cells' regulatory processes. Notably these regulating entities are dynamical objects whose monitoring requires us to introduce new concepts and ideas.

The chapter "Models of Gene Regulation: Integrating Modern Knowledge into the Random Boolean Network Framework" introduces the theme of regulatory genetic networks, modeled by means of random Boolean networks—RBN for short—an abstract framework introduced four decades ago and now becoming one of the major models for complex systems due to their interesting dynamical behavior. In recent years interest in this approach has been renewed through important theoretical advances and also, as far as the application to genetics is concerned, by the availability of genome-wide expression data which can be properly described by RBNs. Moreover, the new versions of this framework can now describe complex phenomena like cell differentiation and whole organisms or tissues. This framework is a common feature of the first three chapters of this part of the book. Thus, Christian Darabos, Mario Giacobini, Jason H. Moore, and Marco Tomassini introduce into RBNs abstractions inspired by recent advances in genetics and biology. In particular, they discuss the topological structure of gene relations and the effects of the adjournment strategy on the model results.

Marco Villani and Roberto Serra discuss the stability properties of RBNs, introducing a new measure (attractor sensitivity) that seems particularly relevant for their application to the dynamics of gene regulatory networks. They also review results that show that RBNs can properly account for data on perturbations induced by gene knock-out in real organisms, thus revealing that living beings tend to live in, or close to, critical states. Last but not least, the authors show that adding noise to RBN framework can lead to a nice model of cell differentiation.

Stefano Benedettini, Andrea Roli, Roberto Serra, and Marco Villani show that it is possible to generate (evolve, train) an ensemble of Boolean networks that can accomplish particular requirements, while keeping the other main relevant statistical features of classical RBNs. This ensemble can be designed by means of optimization processes in which metaheuristics can optimize suitable objective functions.

The chapter by Pasquale Stano, Giordano Rampioni, Luisa Damiano, Francesca D'Angelo, Paolo Carrara, Livia Leoni, and Pier Luigi Luisi shares the spirit of these studies, focusing on the chemical communications among cells, seen as autopoietic objects, stressing again the importance of the structure of feedbacks

that constitute the current organizational processes of living beings. The authors moreover introduce the theme biological/chemical information and communication technology (bio/chem-ICT), which aims at extending the well-known field of ICT, up to now classically based on the transmission of electrical or electromagnetic signals, to the biological/chemical world of molecules.

Part III: Mind and Society

This part of the book presents models that deal with organism behavior and sociocultural phenomena. The chapters that make up this part reflect different approaches to artificial life: from biomimetic models (Santucci et al.), to models that try to reveal general principles (Pugliese), to models that are directly related to natural phenomena (Acerbi et al.), to the use of artificial life techniques for applicative purposes (Gigliotta et al.).

Building artificial agents able to autonomously learn new skills and to easily adapt to different and complex environments is an important goal for robotics and machine learning. In their chapter, Vieri G. Santucci, Gianluca Baldassarre, and Marco Mirolli propose that artificial agents with a learning signal that resembles some characteristic of dopaminergic neurons would be an advancement in the development of more autonomous and versatile systems, thanks to the enabling of cumulative learning abilities. To validate this hypothesis they perform experiments with a simulated robotic system that has to learn different skills to obtain rewards, showing that the proposed learning signal is able to drive the cumulative acquisition of different skills in a way that would not be possible otherwise.

The contribution by Francesco Pugliese presents evolutionary robotics models of the development of categorization abilities. Two different experiments are described, one involving mobile robots that perceive the color of the floor through ground sensors, and the other involving a robotic camera that can move on images. In both cases, the environments contain noisy images that must be categorized, and in both cases evolved robots are able to perform the task. More importantly, in both experiments, the robots that during evolution are facilitated by receiving linguistic signals that tell the robot the category of the perceived image achieve better performance than the robots that did not receive any help, even when, after evolution, the facilitating signals are not provided. Hence, the reported simulations suggest that social linguistic input may exert a facilitating role for the individual development of categorization abilities.

It is not only individual behavior that can be explained through computational models, but also social and cultural phenomena. The next contribution, by Alberto Acerbi, Stefano Ghirlanda, and Magnus Enquist, presents simple models of cultural evolution that try to explain cultural dynamics. In particular, the contribution deals with regulatory traits, that is traits that are culturally transmitted but that, in turn, regulate cultural transmission, such as the propensity to copy others or the ability to influence others. The authors study how the evolution of these traits influences

cultural phenomena like the emergence of open or conservative societies or the ups and downs of cultural traits, i.e., fashions, and they conclude that the presence of regulatory traits renders cultural evolution more flexible than genetic evolution, requiring substantially different models.

In the last chapter of this part, Onofrio Gigliotta, Orazio Miglino, Massimiliano Schembri, and Andrea Di Ferdinando show two ways to build serious game systems exploiting the power of artificial life (AL) techniques for educational purposes; in particular, the authors apply agent-based models, neural networks, genetic algorithms, and robotics. In the first approach, neural networks and genetic algorithms are utilized as open tools to guide artificial organisms design, in order to make it possible for users to learn the fundamental principles of autonomous robotics. In the second case AL techniques are used to model game mechanics—e.g., artificial team dynamics and avatar behavior—whereas the user learns psychological leadership theories by governing a team of artificial agents, the followers. The two cases show how AL techniques can boost serious game systems toward a new level of usability in the context of bioinspired evolutionary design processes and in the field of management training.

Part IV: Applications

This part of the book contains five chapters that give readers an idea of the wide range of applications that can be tackled using techniques derived from the disciplines under consideration and from their hybridization. As shown, apart from the immediately perceptible diversity of the application fields, it is interesting to note how the very nature of results belong to different domains, from very concrete and direct industrial applications, among which signal and image processing and analysis techniques are rather popular, to the creation of models which can be used to forecast the actions of drugs and to assess patient life expectations, from software agents developed to live in and patrol virtual environments such as communication networks to physical agents that can interact with biological tissues.

An example of this latter, futuristic kind of application is offered in the chapter by Oleg Semenov, Darko Stefanovic, and Milan N. Stojanovic, where they study the behavior of synthetic nanoscale walkers made with catalytic DNA legs attached to a rigid body, called molecular spiders, which are able to move across a surface propelled by the multivalent chemical interactions of their multiple legs with the surface itself. Molecular spiders may find important use in biomedical applications, such as searching for clinically relevant targets on the surface of a cell. The authors present simulation-based results on the efficiency of concurrent search for multiple targets by multiple molecular spiders, which influence each other's motion through stigmergic processes.

The two following chapters provide examples of the use of evolutionary computation techniques to solve pattern recognition problems of biomedical interest.

Mario Giacobini, Paolo Provero, Leonardo Vanneschi, and Giancarlo Mauri use genetic programming (GP) to analyze the genetic profile of cancer patients to forecast the outcome of the pathology and to tailor therapy individually. The so-called 70-gene signature was analyzed using a number of pattern-recognition techniques, against which the authors' GP-based approach was compared and shown to outperform the others.

Antonia Azzini, Mauro Dragoni, and Andrea G.B. Tettamanzi apply the results of their years-long research into the hybridization between EC and neural nets to a dataset of ECG recordings, acquired using cheap devices and transmitted over low-band connections with a configuration designed to meet the needs of African countries where cardiologists are not available on-site and the quality of communications is often very low. The dataset on which the evolutionary networks have been tested was previously used in a contest, so the results of the hybrid evolutionary neural system could be compared with those obtained by a large number of other approaches, performing comparably to the best of these.

The chapter by Yvonne Bernard, Lukas Klejnowski, David Bluhm, Jörg Hähner, and Christian Müller-Schloer describes a system which also relies on evolutionary computation techniques. Their approach is based on the ideas of a discipline, called organic computing, which studies the development of agents that are able to cooperate and continuously self-adapt to cope with changing environmental conditions. In their chapter the authors develop evolutionary agents that act in the Trusted Desktop Grid, a distributed computing environment, with no central control, which optimizes the sharing of computing resources.

Massively multiplayer online games (MMOGs) are increasingly successful, since they allow players to explore huge virtual worlds and to interact in many different ways, either cooperating or competing. Given the huge and ever-growing number of users, game designers have to apply strong scalable real-time strategies in order to maintain control of the system. Stefano Sebastio, Michele Amoretti, Jose Raul Murga, Marco Picone, and Stefano Cagnoni present a middleware called PATROL, based on a structured peer-to-peer overlay scheme. Among other features, PATROL provides AI-based modules to detect cheating attempts that the decentralized communication infrastructure may favor: in particular, the authors show how honest bots can detect cheating bots in real time, using strategies based on neural networks. The evolutionary agents' community is evaluated in three different situations: coexistence and competition with the best-performing adaptive agents developed in previous studies; behavior in the presence only of evolutionary agents; and, finally, in an environment in which the presence of egoistic agents introduces disturbance into the system.

Part V: Evolution

The last part of the book is dedicated to research issues in evolutionary computation and deals with two hot topics in the field, of which the former is more theory-oriented, while the latter is more technology-related.

Multiobjective optimization consists of the search for a set of so-called non-dominated solutions which provide a sampling of the Pareto front for the problem, i.e., the set of optimal values which can be obtained for each criterion, once the values of the other, independent and usually counteracting, criteria have been set.

Parallel implementation of evolutionary algorithms has always been a popular issue, thanks to their intrinsically parallel structure, but it is presently booming since multicore CPUs and handy development environments have become available, making multicore or general-purpose Graphics Processor Unit (GP-GPU) computing a higher-level task; this contrasts with the effort that was required to develop similar programs for exploiting the low-level features of processors and, especially, of GPUs, using traditional programming languages that did not have specific support for parallel computation.

Hernán Aguirre, Akira Oyama, and Kiyoshi Tanaka describe an evolutionary multiobjective algorithm which tries to bias the convergence of the population of trade-off solutions onto the Pareto front such that they assume a desired statistical distribution. To this end, they propose Adaptive ϵ-Ranking, which iterates a sampling procedure that applies ϵ-dominance with a suitable mapping function. An analysis of the experimental results, made on the functions of the DTLZ family with six objectives, shows that recombination plays a crucial role in finding a set of solutions with the desired distribution.

Finally, Kiyoharu Tagawa proposes a parallel Java implementation of differential evolution for multicore processors, of which two versions are compared: one is demonstrated to be computationally preferable, as it is able to exploit the multiprocessor's parallel capabilities more efficiently, while the other appears to provide more consistent results over different functions and thread configurations. Both versions benefit from a significant speed-up with respect to a sequential implementation, up to 60× when high-dimensional problems are tackled.

Parma, Italy — Stefano Cagnoni
Rome, Italy — Marco Mirolli
Reggio Emilia, Italy — Marco Villani

Acknowledgments

We would like to gratefully acknowledge all the people whose contributions made it possible to realize this project.

In the first place, of course, the authors of the chapters, for contributing original chapters or for making significant efforts to improve and extend their already excellent contributions to WIVACE.

Next, we thank the Fondazione Cariparma (http://www.fondazionecrp.it/) for its substantial financial support for the organization of WIVACE and for the distribution of printed copies of this resulting book to the workshop participants.

Finally, we thank the scientific sponsors: the EU-funded AWARENESS coordination action (http://www.aware-project.eu/), with a special thanks to Jennifer Willies, always at the forefront when help is needed, and to Alessandro Filisetti; the Italian Association for Artificial Intelligence (http://www.aixia.it/) and its President, Paola Mello; the University of Parma (http://www.unipr.it/); the University of Modena and Reggio Emilia (http://www.unimore.it/); and the Institute of Cognitive Sciences and Technologies of the Italian National Research Council (http://www.istc.cnr.it/).

We sincerely hope that the readers will appreciate the results of this effort, and that the publication of this book may further extend the overlap between evolutionary computation, the study of complex systems, and artificial life—fields which we have tried to show are separate but necessary components, or viewpoints, of possibly the same artificial, but less and less virtual, world.

September 2013


UNIVERSITÀ DEGLI STUDI DI PARMA


FONDAZIONE CARIPARMA


CNR
ISTC
ISTITUTO DI SCIENZE E TECNOLOGIE DELLA COGNIZIONE


UNIVERSITÀ DEGLI STUDI
DI MODENA E REGGIO EMILIA


awareness
SELF-AWARENESS IN
AUTONOMIC SYSTEMS

Contents

Part I Research Issues

One Artefact: Many Phenomena 3
Domenico Parisi

Taming the Complexity of Natural and Artificial Evolutionary Dynamics 19
Riccardo Poli and Christopher R. Stephens

Part II Biological Modeling

Models of Gene Regulation: Integrating Modern Knowledge into the Random Boolean Network Framework 43
Christian Darabos, Mario Giacobini, Jason H. Moore, and Marco Tomassini

Attractors Perturbations in Biological Modelling: Avalanches and Cellular Differentiation 59
Marco Villani and Roberto Serra

Automatic Design of Boolean Networks for Modelling Cell Differentiation 77
Stefano Benedettini, Andrea Roli, Roberto Serra, and Marco Villani

Towards the Engineering of Chemical Communication Between Semi-Synthetic and Natural Cells 91
Pasquale Stano, Giordano Rampioni, Luisa Damiano, Francesca D'Angelo, Paolo Carrara, Livia Leoni, and Pier Luigi Luisi

Part III Mind and Society

Cumulative Learning Through Intrinsic Reinforcements 107
Vieri G. Santucci, Gianluca Baldassarre, and Marco Mirolli

Development of Categorisation Abilities in Evolving Embodied Agents: A Study of Internal Representations with External Social Inputs 123
Francesco Pugliese

Regulatory Traits: Cultural Influences on Cultural Evolution 135
Alberto Acerbi, Stefano Ghirlanda, and Magnus Enquist

Building Up Serious Games with an Artificial Life Approach: Two Case Studies 149
Onofrio Gigliotta, Orazio Miglino, Massimiliano Schembri, and Andrea Di Ferdinando

Part IV Applications

The Effects of Multivalency and Kinetics in Nanoscale Search by Molecular Spiders 161
Oleg Semenov, Darko Stefanovic, and Milan N. Stojanovic

Towards the Use of Genetic Programming for the Prediction of Survival in Cancer 177
Marco Giacobini, Paolo Provero, Leonardo Vanneschi, and Giancarlo Mauri

A Neuro-Evolutionary Approach to Electrocardiographic Signal Classification 193
Antonia Azzini, Mauro Dragoni, and Andrea G.B. Tettamanzi

Self-Organisation and Evolution for Trust-Adaptive Grid Computing Agents 209
Yvonne Bernard, Lukas Klejnowski, David Bluhm, Jörg Hähner, and Christian Müller-Schloer

Honest vs Cheating Bots in PATROL-Based Real-Time Strategy MMOGs 225
Stefano Sebastio, Michele Amoretti, Jose Raul Murga, Marco Picone, and Stefano Cagnoni

Part V Evolution

Distribution Search on Evolutionary Many-Objective Optimization: Selection Mappings and Recombination Rate 241
Hernán Aguirre, Akira Oyama, and Kiyoshi Tanaka

Concurrent Implementation Techniques Using Differential Evolution for Multi-Core CPUs: A Comparative Study Using Statistical Tests 261
Kiyoharu Tagawa

Erratum E1

Part I
Research Issues

One Artefact: Many Phenomena

Domenico Parisi

Abstract Robots are a new way of expressing theories of behaviour. If a robot behaves like a real organism, the ideas which have been used to construct the robot capture what lies behind behaviour and therefore explain behaviour. But robots can be "toys" that do not tell us very much of truly interesting about the behaviour of real organisms. Current robots—especially because they are constructed with specific applications in mind—tend to follow the principle "one artefact/one phenomenon" and, if one follows this principle, many different artefacts can be constructed which reproduce the same phenomenon and it is arbitrary to choose among them. To avoid constructing "toy" robots, one should follow the opposite principle "one artefact/many phenomena" and one and the same robot should reproduce a variety of known phenomena concerning behaviour. We illustrate a variety of different phenomena concerning the behaviour of human beings and their societies which robots that can be properly called "human" (and not just "humanoid") should be able to reproduce.

1 Introduction

Imagine you want to understand the behaviour of living organisms by constructing artefacts which behave like living organisms. If you can show that the artefact actually behaves like a living organism, you are entitled to claim that the ideas on the basis of which you have constructed the artefact explain the behaviour of the living organism. This is a new approach to doing science: understanding reality by remaking reality. Traditionally, science explains reality by proposing mathematical theories or theories expressed in words. The new approach formulates scientific

D. Parisi (✉)
Institute of Cognitive Sciences and Technologies, National Research Council, Rome, Italy
e-mail: domenico.parisi@istc.cnr.it

S. Cagnoni et al. (eds.), *Evolution, Complexity and Artificial Life*,
DOI 10.1007/978-3-642-37577-4_1,

theories not by using mathematical or verbal symbols but by incorporating them in artefacts.

The problem with trying to understand the behaviour of living organisms by constructing artefacts that behave like living organisms is that the artefacts can be "toys", something which can be interesting and entertaining but does not tell us very much about reality. The problem is not that the artefacts simplify with respect to reality. The artefact is a theory and all theories simplify with respect to reality and they let us understand reality because they capture the basic mechanisms and processes that underlie the observed phenomena, leaving everything else out. The problem is that the artefact may not make the *appropriate* simplifications and it may not capture what is truly important about the phenomena of interest. So, the answer to the question 'Is the artefact a "toy" or does it really let us understand reality?' is not clear, and this may explain why traditional scientific disciplines do not appear to be much interested in artefacts as theories. (There are also other reasons for this lack of interest: expressing theories as artefacts is a big change in how to do science which will take time to impose itself and scientists are not routinely taught the necessary skills and methodologies for expressing theories as artefacts and for understanding the theories so expressed.)

One way of dealing with the problem of theories/artefacts as "toys" is to apply the principle "*one* artefact/*many* phenomena". Most current theories/artefacts follow the opposite principle: "*one* artefact/*one* phenomenon". One single phenomenon can be reproduced by constructing many different artefacts, and it may be arbitrary to choose among these different artefacts. (In most cases the criterion for choosing is that it is "my" artefact.) On the contrary, if the same artefact reproduces *many* different phenomena, there are fewer possible artefacts that can reproduce all the phenomena and choosing among them becomes easier and less arbitrary.

Artefacts that are intended to reproduce the behaviour of living organisms are called robots. Robotics is flourishing today but current robots are technologies, not science. They are not constructed to express theories of behaviour and as tools to better understand the behaviour of living organisms but with practical applications in mind. Robots are constructed for industrial and military applications, for assistance to ill or old people, for medicine and surgery, for training and entertainment, and for other practical uses. Robots as science and robots as technologies should not be completely separated because robots as technologies can pose new problems to science and robots as science can suggest new practical applications. But the two research fields should not be confused because they have different criteria of success. The criterion of success for robots as technologies is "Does the robot have practical applications and economic value?" while the criterion of success for robots as science is "Does the robot help us to better understand the behaviour of living organisms?" Today, most research money is for robots as technologies and this has the consequence that the potential of robots for developing a new and more powerful science of behaviour is not really exploited. This has implications for our problem of artefacts as "toys". Robots as technologies do only one thing—the practical application for which they have been constructed—and, therefore, they

follow the principle "one artefact/one phenomenon". Robots as science must follow the opposite principle "one artefact/many phenomena".

In this chapter we discuss a number of different behavioural phenomena that our robots should be able to reproduce. Some of these phenomena are found in all sorts of animals but many of them only exist in human beings. Today, one often hears of humanoid robots but humanoid robots are robots which externally resemble human beings, exhibit some elementary human behaviours such as reaching and grasping an object with the hand or walking on two legs, respond appropriately to sounds (words), and move their eyes and face so as to express human-like emotions. But their resemblance to human beings is very limited and, often, superficial and they tell us very little of interest about human beings because they do not follow the principle "one artefact/many phenomena".

Human beings are very complex animals and constructing human robots is a very complex task—and it mainly remains a task for the future. But constructing human robots by following the principle "one artefact/many phenomena" has an interesting implication. Reality is a very large ensemble of different phenomena and science inevitably segments reality into separate parts and entrusts the study of these different parts to separate scientific disciplines. The problem is that all the phenomena of reality are linked together and, often, the phenomena which are studied by one scientific discipline can only be understood and explained by taking into consideration the phenomena studied by another scientific discipline. The problem is not so serious for the sciences that study nature—physics, chemistry, biology—because these sciences share the same empirical methods, the same type of theories, and the same conception of reality as made of physical causes that produce physical effects and as possessing an intrinsic quantitative nature. For the sciences that study human beings the situation is different. These sciences—psychology, linguistics, anthropology, sociology, economics, political science—have very different methods, very different theories and conceptual traditions, and do not have a shared conception of the object of their study—human beings. Human beings are very complex animals but one important reason why we still do not understand them is the existence of separate scientific disciplines for studying them.

Constructing human robots by following the principle "one artefact/many phenomena" implies ignoring the divisions among the different disciplines and sub-disciplines that study human beings. The robots should not have only skills and capacities but also motivations and emotions, they should not have only behaviours but also a mental life, they should have languages, cultures, technologies, societies, economies, and political organizations. This is clearly a difficult task. But by following the principle "one artefact/many phenomena", robotics can become a "lingua franca" that unifies all the different disciplines that study human beings—much as mathematics is a "lingua franca" which unifies all the disciplines that study nature.

2 Human Robots

In this section we briefly describe a number of human phenomena that we should reproduce with robots if we want to be justified in calling the robots "human robots". All these phenomena should be reproduced by using the same robotic platform.

2.1 Robots that Evolve, Develop, Learn, and Have Cultures and Technologies

Most current robots are programmed by us to exhibit the behaviours they exhibit but robots as science cannot be programmed by us because real organisms are not programmed by anyone. If we follow the principle "one artefact/many phenomena", our robots should not only exhibit the same behaviours which are exhibited by real organisms but they should also acquire those behaviours in the same way as real organisms acquire them. Real organisms autonomously acquire their behaviours through a variety of different processes. One is evolution: changes that occur in a succession of generations of individuals which reproduce selectively and with the constant addition of random variations to the inherited genotypes due to recombination and genetic mutations. Another one is development: changes which occur during the life of an individual and which are mainly specified in the inherited genotype. A third one is learning: changes which also occur during the life of the individual but which are mainly due to the particular experiences that the individual happens to make in the environment in which it lives. The last two ones are almost uniquely human: cultural changes and technological changes. Cultural changes are changes in the behaviours shared by a community of individuals that interact together and learn from each other (culture) and which are due to the selective imitation of behaviours and the constant addition of new invented behaviours. Technological changes are due to copying the best existing artefacts and purposefully changing them to make them better. As we have said, culture and technology are almost exclusive human adaptations. Unlike nonhuman animals, human beings live in an artificial environment which they themselves have created with their cultures and technologies and, since they continuously change this environment, the environment continuously changes them.

There is some work on robots that evolve and learn and much less work on robots that develop, while there is almost no work on robots that have cultures and technologies (on robots that have cultures and technologies, see Sect. 2.10 below). But what is crucial is that, in human beings, all the different processes of change occur together. Therefore, we should not construct robots that either evolve or learn or develop or have cultures and technologies but we should construct robots which at the same time evolve, learn, develop, and change their cultures and technologies. Only if we construct robots like these, we will be able to reproduce with our robots how the different processes of change interact together and influence

each other. There is some work on robots that both evolve and learn and on robots that both evolve and develop according to a development program encoded in their genotype, but most of the work is still to be done. Among the problems which should be addressed by adopting the principle "one artefact/many phenomena" are the following. How evolution creates the rewarding value of some stimuli and the punishing value of other stimuli which both play a crucial role in learning? How evolution creates the basis for learning from others and, therefore, the emergence of cultures? Is there an influence of learning on evolution and of culture on evolution?

2.2 Robots in Their Natural Environment and in the Experimental Laboratory

Organisms are what they are because of the particular environment in which they live and to which they are adapted. Therefore, our robots should live, evolve, develop, and learn in an environment. This is not what is done in today's robotics. Even when a robot is not programmed by us but it learns to do what it does, the robot is trained in a sort of experimental laboratory in which the robot learns to respond to the stimuli provided by us and the context in which learning occurs is either absent or controlled by us. The experimental method is a fundamental tool of science but, when it is applied to the study of behaviour, it has many limitations because the natural environment to which an organism is adapted is very different from the experimental laboratory. In the natural environment the stimuli that arrive to the organism's sensors are not decided by the experimenter but are largely determined by the organism's own behaviour and context always exists and it is unpredictable and mostly uncontrollable. Psychologists study behaviour in the experimental laboratory because studying behaviour in the natural environment is very difficult and expensive. This is where robots can be useful. By applying the principle "one artefact/many phenomena", we should study the behaviour of a robot both in the "natural environment" in which the robot lives and in an "experimental laboratory" controlled by us. And, in addition, we can make "natural experiments" which are impossible to do with real animals: we let different populations of robots live and evolve in different environments and look at the consequences that these different environments have for the robots' behaviour and adaptive pattern. Studying the behaviour of robots both in their "natural environment" and in an "experimental laboratory" will allow us to better understand why real animals and real human beings behave as they behave in laboratory experiments and may suggest what to look for in their natural environment.

2.3 Neurorobots

The behaviour of most animals is controlled by a brain. Therefore, the principle "one artefact/many phenomena" requires that we work not just with robots but with

neurorobots, that is, robots whose behaviour is controlled by a "brain": an artificial neural network made up of neuron-like units and synapse-like connections. The neural network that controls the robots' behaviour can be much simplified with respect to the real brain but what is important is that this simplified neural network reproduces interesting behaviours. Progressively, these simplifications should be eliminated and more and more of the actual anatomy and physiology of the brain should be reproduced in the neural network of the robots so that the robot is able to reproduce more behaviours and more sophisticated behaviours. However, the neural network of a neurorobot should not reproduce what neuroscientists know about the brain as an end in itself—as is often done in computational neuroscience—but it should allow us to understand how the different structures and processes of the brain identified by neuroscientists control behaviour. Biological entities can only be understood if we understand their function. We cannot understand the heart or the lungs unless we understand how they control the circulation of blood and air in the body. The function of the brain is to control behaviour, and we cannot understand the brain if we do not understand how its different anatomical structures and physiological processes result in behaviour.

2.4 Robots that Have Both Halves of the Mind

The human mind—and the mind of all animals—has two halves: the cognitive half and the motivational/affective half. Most work on robots is dedicated to the cognitive half of the mind, to capacities, skills, knowledge, thinking, reasoning, and planning. But one cannot really explain the behaviour of nonhuman animals and human beings if one ignores the other half of their mind: their motivations, their emotions, the value of things for them. Two reasons explain why the motivational/affective half of the mind plays a marginal role in robotics. One is that objective and quantitative empirical data on the motivational/affective half of the mind are more difficult to obtain and to reproduce than objective and quantitative data on the cognitive half of the mind. The other reason is linked to the applied nature of current robotics. Robots as practical applications can be *cognitively* autonomous but they cannot be *motivationally* autonomous. Given a task, they should autonomously carry out the task. This is cognitive autonomy. But they cannot have their own motivations and decide autonomously which motivation to try to satisfy with their behaviour at any given time because they should not do things for themselves but for us. Therefore, they cannot have motivational autonomy.

Constructing robots that have both halves of the mind has other implications for the principle "one artefact/many phenomena". Current robots have a body and neurorobots also have a brain. But the body of robots is a body with an external shape and external sensory and motor organs, and their brain only interacts with the external environment. The body of real organisms does not only have an external shape and external sensory and motor organs but it also has *internal* organs and systems and their brain interacts not only with the external environment but also

with these internal organs and systems. Adding an "internal" robotics to the current "external" robotics is an important step forward from the point of view of the principle "one artefact/many phenomena".

Robots with motivational autonomy are robots which can actually have emotions, in opposition to current robots which only *appear* to have emotions because they reproduce emotional expressions in their face. An organism has many different motivations but it cannot satisfy all its motivations at the same time. Therefore, at any given time the organism must "decide" which motivation to try to satisfy with its behaviour, and these motivational decisions are even more important, for the organism's survival and reproduction, than the skills and cognitive capacities that allow the organism to satisfy its motivations. Emotions are states of the organism's brain/body that allow the organism to take better and faster motivational decisions. The principle "one artefact/many phenomena" requires that we construct robots that make motivational decisions and have emotions. And this has implications for the robots' neural network which must be more realistic and must include not only neurotransmission but also neuromodulation.

While the "cognitive" circuits based on neurotransmission mainly respond to sensory input from the external environment (and to sensory input self-generated inside the brain; see Sect. 2.6 below) the neuromodulatory circuits which encode emotional states interact with the rest of the body, with both the internal organs and systems inside the body and the external shape and appearance of the body. This causes the external expression of emotions which plays such a crucial role in social interaction and, given this role, it is possible that the effects of emotional states on the external shape and appearance of the body have been an important adaptive pressure for developing emotional states.

2.5 Robots that Have Language

Language is a crucial component of the adaptive pattern of human beings and there is considerable work on robots that have "language". The language of today's robots, like human language, is made of sounds which have arbitrary meanings (words) but from all other points of view it is more like an animal communication system than human language. Its rarely has grammatical classes of words (verbs, nouns, adjectives, etc.), signals made up of smaller signals (compositionality), "syntactic rules" to put together the meaning of smaller signals and generate the meaning of larger signals, ambiguous words, abstract words, metaphors, and idiomatic expressions. These are all semantic and syntactic limitations of the "language" of current robots. But the "language" of current robots is not like human language especially from a pragmatic point of view. Human beings do all sorts of things with language: they describe, inform, ask, command, pray, suggest, try to convince, etc. Most current robots do only one of two things with their "language": they either name objects and actions or they respond to commands.

But language is important for human beings not only as a communication tool but because it shapes the human mind. Words co-vary with *separate* components of the individual's overall nonlinguistic experiences and these co-variations are incorporated in the brain of an individual so that nonlinguistic experiences are segmented into parts which can be recombined together in novel ways, making human behaviour more articulated and creative. For example, the overall nonlinguistic experience of seeing someone who grasps a glass with his or her hand can be accompanied, in different occasions, by hearing a word that co-varies with the action of grasping but not necessarily with grasping glasses, or with glasses but not necessarily with grasping them, or with the person who is grasping the glass but not necessarily with the person when he or she is grasping the glass. This capacity to segment nonlinguistic experiences is a crucial aspect of human language which should be reproduced with robots if the robots are to be legitimately called human robots and if we want to follow the principle "one artefact/many phenomena".

Other crucial properties of human language that should be reproduced with robots are its innate bases and the role played by learning specific languages. One should evolve a population of robots which are born with a genotype which allow them to learn the specific language spoken in the robots' community, whatever the language. The robots which are not born with that type of genotype should not be able to learn any language but the specific language learned by the robot should segment the robot's nonlinguistic experiences in specific ways (see above).

2.6 Robots that Have a Mental Life

Another crucial component of the human adaptive pattern is that human beings have a mental life. Mental life is the self-generation of sensory inputs by the brain. Sensory inputs normally are caused by events in the external environment or inside the organism's body but outside the brain. But the brain of human beings can also self-generate its own sensory inputs and respond to these self-generated sensory inputs, perhaps by self-generating other sensory inputs. This is remembering, imagining, predicting, planning, thinking, dreaming, and having hallucinations. The self-generated sensory inputs can be nonlinguistic sensory inputs but language plays a central role in the mental life of human beings because human beings talk to themselves. Their brain self-generates linguistic sounds and they respond to these self-generated linguistic sounds as they respond to the linguistic sounds produced by other human beings.

Both nonlinguistic and linguistic mental life should be reproduced in robots if we want to construct human robots by respecting the principle "one artefact/many phenomena". The robot's neural network includes some units whose pattern of activation is very similar to the pattern of activation observed in the neural network's sensory units and is processed in very similar ways but while the pattern of activation of the sensory units is caused by events outside the brain, the pattern of activation of these units is self-generated inside the brain.

One crucial aspect of mental life is the ability to predict. Predicting should be distinguished from anticipating. All animals anticipate future sensory inputs because their brain incorporates in its structure the co-variations between successive sensory inputs. But only human beings predict, where predicting is self-generating in one brain's a sensory input which is like some future sensory input. Predicting is important because it allows the brain to replace missing sensory inputs with predicted sensory inputs and to judge if the predicted effects of a planned action are "good" or "bad" and, on this basis, to decide whether to actually execute the action, and it becomes particularly effective when the predicted sensory inputs are linguistically labeled.

Mental life plays an important role not only with respect to the cognitive half of the human mind but also with respect to its affective half. In all animals sensory inputs often cause "good" or "bad" emotional states because they are associated with either an increase or a decrease in the organism's survival and reproductive chances. Seeing one's mate or one's offspring causes a "good" emotional state because it is associated with the reproduction of one's genes. Seeing a predator causes a "bad" emotional state because it is associated with the risk of dying. Nonhuman animals do not have a mental life and therefore for them "good" and "bad" emotional states are only evoked by actual sensory inputs from the external environment or from inside their body. The human brain self-generates sensory inputs and these self-generated sensory inputs can evoke "good" and "bad" emotional states. This is why, unlike nonhuman animals, human beings can be happy and unhappy, where happiness is the tendency to have many "good" emotional states and unhappiness is the tendency to have many "bad" emotional states because these "good" or "bad" emotional states are evoked by continuously self-generated sensory inputs.

The self-generation of sensory inputs which cause "good" or "bad" emotional states also explains other human phenomena. Many forms of psychopathology are associated with the inability to block the self-generation of sensory inputs that cause "bad" emotional states. (On psychopathological robots, see Sect. 2.7 below.) Religious beliefs and practices have the function both to favour the self-generation of sensory inputs that cause "good" emotional states (imagining a protecting god) and to block the self-generation of sensory inputs that cause "bad" emotional states (imagining a life after death). Meditation can be seen as entirely blocking mental life and the tendency to self-generate all sorts of sensory inputs. Art is the self-generation of "good" sensory inputs by physically acting on the environment by creating paintings, music, and literary works—and even "bad" sensory inputs because practicing art both as author and public can be useful to become more sophisticated in dealing with "bad" sensory inputs in real life. Human robots should have psychopathologies, religious practices and beliefs, meditation, and art, and endowing them with a mental life as the self-generation of sensory inputs is a necessary pre-condition.

2.7 Robots that Are Inter-Individually Different and Have Psychopathologies

An important property of living organisms is that no two individuals are identical and, if we want to follow the principle "one artefact/many phenomena", we should reproduce inter-individual differences with robots. We should construct populations of robots, not individual robots, and we should examine how one robot is different from all other robots. This can be done if the robots acquire their behaviours through biological or cultural evolution (see Sect. 1 above) which both take place in populations of inter-individually different robots. However, genetic and cultural algorithms are often used in robotics as a technique to arrive at the best solution to some pre-defined problem and, therefore, one is interested in the best robot, not in all the robots of the population and in what makes one robot different from other robots. In contrast, in many cases one can better understand behaviour by studying individuals which are not very good at doing what is needed for their survival and reproduction.

Another limitation of current evolutionary robotics is that what one is interested in is the "fitness" of the robots, which is a single measure of a robot's behaviour, while two robots can have the same fitness but they are very different if one examines their behaviour in the controlled conditions of an "experimental laboratory". For example, one robot can be very good at approaching food when it sees the food but the robot does not explore adequately the environment when it does not see any food, while another robot may not be very good at approaching food when it sees the food but it explores many parts of the environment and, in this way, increases the probability to find the food. The robots have different degrees of the two capacities, approaching food and exploring the environment, but they have the same fitness.

Inter-individual differences exist with respect to both the cognitive half of the mind and its motivational/affective half. Two robots can be equally good at approaching food but one robot immediately stops looking for food and flies away when a predator appears, while the other robot is slower to react to the predator or it becomes paralysed when it sees the predator. Inter-individual differences are linked to pathologies since pathologies are conditions which cause an individual to exhibit behaviours which seriously damage the individual's chances to survive and reproduce, or be happy. By applying the principle "one artefact/many phenomena", we should reproduce with robots not only healthy behaviours but also pathological behaviours, and the pathological behaviours should concern both the cognitive half of the mind (neurological illnesses) and the motivational/affective half (psychiatric or psychological illnesses). We should examine not only the behaviour of ill robots but also their brain, we should study what has caused their pathological condition, and we should make predictions on their illness and find how to cure mentally ill robots.

Clearly, inter-individually different robots and pathological robots do not make sense if robots are constructed for practical applications. Industrial production is based on making identical copies of a prototype, inter-individually different robots

are not reliable and their behaviour cannot be predicted, and certainly one does not want robots that have pathologies. This clearly demonstrates that robots as technology and robots as science are very different and that the principle "one artefact/many phenomena" can only be applied to robots as science.

2.8 Life History, Sexual Differences, and Families

Current robots just exist. Human beings do not just exist but they are born, develop, become reproductive, have offspring, age, and die. And at each stage of their life history not only their body changes but their mind and their behaviour also change. Furthermore, human beings are not just human beings but they are male and female human beings, and male and female human beings have different roles in reproduction and may have different minds and different behaviours because of genetic and cultural reasons. If we want to follow the principle "one artefact/many phenomena", we should not construct "robots" but we should construct robots that are born and die, that have a life history, and that are male and female. Some work is being done on infant robots and on how they develop and learn but most current robotics ignore life history and sexual differences.

Human beings, and animals more generally, do not behave in the same way towards other human beings who have the same genes and towards genetically extraneous human beings. Parents feed and care for their offspring, mothers, fathers, and their offspring live together in families, young offspring stay near to their parents to be protected by them and to learn from them. Generally, human beings behave altruistically towards other individuals who have similar genes, where behaving altruistically is behaving in ways that increase the survival/reproductive chances of the other individual and decrease the survival/reproductive chances of the individual who behaves altruistically, while they tend to behave selfishly towards genetically extraneous individuals. These differences in behaviour should be reproduced with robots, and robots should also help us to find out when and in what conditions altruistic behaviours are exhibited towards genetically unrelated individuals.

2.9 Robots Which Obtain What They Need from Other Robots

Many animals are social animals. They interact not only with inanimate objects and members of other animal species but also with members of their same species. Human beings are an eminently social species and, therefore, the principle "one artefact/many phenomena" for human robots requires us to construct robots which have a rich and complex social life. Current robotics ignores most aspects of social behaviour and sociality. Ant or swarm robotics reproduces colonies of insect-like robots which have the same genes and this makes it impossible to study the very

important phenomena linked to sociality among genetically unrelated individuals (see Sect. 2.8 above). And in any case the behaviours exhibited by the robots of ant and swarm robotics are extremely simple. Some robots are constructed to interact with us but these robots are only practical applications and they ignore that sociality is among members of the same species. So, a social robotics is a task for the future.

As we have said, human beings are an especially social species and, in fact, many human phenomena are social phenomena: human societies, cultures, economies, political organizations. But even social life as inter-individual interaction has many complexities. All organisms interact with the environment to obtain from the environment what they need to survive, reproduce, and live well. But the social environment made up of conspecifics is very different from the non-social environment made up of inanimate objects. While to obtain from inanimate objects what they need, organisms act physically on them—they walk of them, grasp them, move them, modify them, construct them, etc.—to obtain what they need from their conspecifics, they must manipulate the motivations of their conspecifics so that their conspecifics behave in ways that satisfy their needs. This is a crucial aspect of social life that we should reproduce with robots. Social life is the capacity to manipulate the motivations of others. So, if we want to reproduce social life with robots, our robots should have motivations and they should be able to know what are the motivations of other robots and how to manipulate the motivations of other robots.

2.10 Robots Which Have Cultures and Technologies

Human beings learn not only by interacting with the inanimate environment but they also learn by interacting with their conspecifics. They imitate their conspecifics and they are taught by their conspecifics. Learning from others is one of the most important advantages of living socially because it is faster and generally more effective than learning by interacting with the non-social environment and because it leads to the emergence of shared behaviours, beliefs, and values which are called the culture of the group and which make the behaviour of others more predictable. Learning from others has many similarities with genetic inheritance. As selective reproduction, genetic recombination, and the constant addition of random mutations to the inherited genotypes lead to biological evolution, in the same way selective imitation and the constant addition of both random noise and purposeful innovation to what is learned from others lead to cultural evolution. But biological evolution and cultural evolution also have many differences. One necessarily inherits one's genotype from one's parents while one can learn from any member of one's community—and the size of the community of individuals from which one can learn is important and can accelerate cultural change. In fact, this may have been a pressure for increasing the size of human societies. Another factor which may have accelerated cultural change is that, as we have already said, what is learned

from others is not only changed by random noise or imperfect learning but also by purposeful innovation.

Another species-specific trait of human beings is that they change the environment in which they live by modifying the environment and by constructing all sorts of artefacts. This is very important to reproduce with human robots because, while nonhuman animals adapt to the environment as they find it, human beings change the environment to make the environment more adapted to them. Artefacts are copied from existing artefacts and the evolution of artefacts (technological evolution) has similarities with both biological evolution and cultural evolution. The artefacts which are taken as models to be copied are selected from the best artefacts—those which allow their users to live better—and they are modified so as to make them better. In technological evolution, purposeful innovation plays an even more important role than in cultural evolution and we can explain—and should reproduce with robots—science as a human trait which has culturally evolved because of its role in technological innovation.

Clearly, if we want to construct human robots by following the principle "one artefact/many phenomena", our robots should have cultures and technologies. A robot learns from a "model" robot by comparing its response to some sensory input to the model's response to the same sensory input and by progressively changing the connection weights of its neural network so that at the end it responds in the same way as its model. (Learning from others represents a pressure on robots to live near to other robots, even if they are not genetically related robots.) Cultural change occurs because robots choose their models based on the success (fitness) of potential models and they add random changes to the behaviour that they copy from their models. A robot copies the artefacts which exist in its community in that its neural network has sensory neurons which encode the properties of an artefact and motor neurons which encode the actions which produce a copy of the artefact. In this case too, the artefacts of the most successful robots are taken as models to be copied and random changes are added so that the copy is not identical to the model. If we allow the robots to learn from the other members of their community and to copy the artefacts existing in their community, the size of the population from which a robot selects its models turns out to be an important variable in the evolution of both cultures and technologies. For example, if the robots limit themselves to copying the artefacts which are used within their family (by their parents), the quality of the artefacts takes more generations to improve compared to a population of robots which choose the best artefacts as models to be copied from the entire pool of artefacts used in their community. This, again, may have been a pressure for increasing the size of human communities.

2.11 Robotic Economies

If we define a "good" as anything which an organism tries with its behaviour to have, the number of things which are goods for human beings is much greater than

the number of things which are goods for nonhuman animals, and in human societies this number tends to increase exponentially. This is linked to the complexity of human economies in which new goods are created by using existing goods, different individuals specialize in the production of different goods, most goods are obtained from others through the exchange of goods, and many individuals coordinate themselves to produce goods that no single individual would be able to produce by working alone. If we want to construct human robots by following the principle "one artefact/many phenomena", clearly our robots should have complex economies. This is a completely new research area but the first steps can already be made.

The first step is to construct robots that have external stores in which they put their goods and to which only one robot, the owner of the goods, has access. Some nonhuman animals store their food for future use but external stores and the ownership of goods is a typical and extremely important human specialty. The possession of external stores allows a robot to survive in difficult seasonal environments and to traverse zones of the environment without food and reach new zones which are rich in food. External stores can be personal stores or family stores, and family stores imply economic inheritance if at birth a robot does inherit not only the genes of its parents but also their goods. As we have said, external stores imply the ownership of goods but, while in ancient human communities the ownership of goods is guaranteed by the owner with its only forces, in modern societies the guarantee of the ownership of goods is entrusted to a central authority—and this has been an important factor in the emergence of states.

Among the most important consequences of possessing external stores for one's goods are the exchange of goods and the emergence of specialization in the production of goods. If the robots need two different types of food to survive, some robots specialize in collecting one type of food and other robots specialize in collecting the other type of food and then all robots can have both types of food by exchanging one type for the other type. Specialization and the exchange of goods increases the robots' fitness in all sorts of environments but it is especially useful if some goods can only be found in one zone of the environment and other goods can only be found in another zone of the environment or if collecting (producing) different types of goods requires different skills.

The exchange of goods leads to the emergence of money as a good which is exchanged in all exchanges and which is only used to obtain goods from others. Money facilitates the exchange of goods because, while one exchanges one good for another good only if it needs the good which it will obtain through the exchange, one always wants to exchange goods for money because money can be used to obtain any good in future exchanges. Money makes it also possible to assign a quantitative value to goods. Goods by definition have value because an organism behaves in ways that allow the organism to have the good. But, in the absence of money, value can only be measured in relative terms: one good has more value than another good. The relative value of goods for a robot can be determined by putting the robot in an experimental laboratory and exposing the robot to two different goods. The good which has more value for the robot is the good which the robot approaches and

reaches. This, as we have said, is the relative value of good and goods have a relative value even for nonhuman animals. But human beings have money, and money is a metrics for measuring the absolute value of goods (or, at least, of the goods which are exchanged for money). Money is made up of identical units and the absolute value of a good is the number of money units which are given in exchange for the good. The absolute value of a good—the price of the good—is determined by the quantity of that good which is offered in exchange for money (selling the good) and the quantity of that good which is asked in exchange for money (buying the good).

Starting with robots that reproduce these basic phenomena, we should reproduce many other phenomena that characterize human economies. Some robots (workers) give their work to another robot (entrepreneur) in exchange for money and the other robot uses their work to produce goods that no single robot would be able to produce by working alone (private enterprises). Borrowing money emerges as a new type of good which is exchanged for money (interest). Another good which exists in human economies is insurance against risk. Risk is the possibility that something which is "bad" for a robot will actually happen. The robot obtains insurance against this risk from another robot, which means that the robot gets some money from the other robot if the risk becomes reality—and, in exchange the first robot gives some money to other robot. And human economies are political economies because in human societies all the members of the society give some of their money (under the form of taxes) to a central authority which uses this money to produce new goods (health, education, and pension systems, infrastructure, a system for discovering and punishing other-damaging behaviours, the capacity to make both defensive and offensive wars) which it then distributes to all the members of the society.

3 Conclusion

Let us summarize. Computers have made it possible to do science in an entirely new way by constructing artefacts that reproduce reality. A scientific theory is no more expressed by using mathematical symbols or the words of common language, perhaps with some redefinitions of terms, but by using the theory to construct an artefact. If the artefact behaves like the piece of reality that we want to understand, the theory which has been used to construct the artefact is confirmed. This new approach to science is particularly useful when science wants to understand and explain the behaviour of human beings. Theories that explain the behaviour of human beings, their mind and their societies, tend to be expressed in words, and words have unclear, ill-defined, and ambiguous meanings, and from verbally expressed theories it is often difficult to derive precise and unambiguous empirical predictions. The artefacts that reproduce the behaviour of human beings—and other animals—are robots. Robots are theories of behaviour, and they are better theories of behaviour than verbally expressed theories because we perfectly know the artefact, we can inspect its internal functioning, and we can observe and measure its behaviour in all sorts of conditions.

But current robots have two problems. They are mostly constructed for practical applications and not to better understand the behaviour of human beings and other animals and they follow the principle "one artefact/one phenomenon" ("on robot/one phenomenon"). Robots for practical applications have different criteria of success than robots as science and robots that reproduce only one single behaviour may be "toys" which reproduce that behaviour in arbitrary ways which do not tell us very much about the real behaviour. So, we have proposed that we explicitly and clearly recognize that robots can be purely scientific tools and that robots as scientific tools should follow the principle "one artefact/many phenomena" ("one robot/many phenomena") because this increases the probability that the robots actually capture what is essential about the observed phenomena and explains them. The principle "one robot/many phenomena" is especially important because it makes it possible to go beyond the disciplinary divisions which are an obstacle to our understanding of the behaviour of human beings. Robots will become a unified science of human beings and they will eliminate the different scientific disciplines that study human beings except as sources of data.

Constructing human robots that respect the principle "one robot/many phenomena" is a very difficult task because human behaviour and human societies are extremely complex phenomena and because the robotic approach to the study of human beings is so new. But we think that the first steps in this direction are already being made and in this chapter we have described some of these steps.

Taming the Complexity of Natural and Artificial Evolutionary Dynamics

Riccardo Poli and Christopher R. Stephens

Abstract The study of complex adaptive systems is among the key modern tasks in science. Such systems show radically different behaviours at different scales and in different environments, and mathematical modelling of such emergent behaviour is very difficult, even at the conceptual level. We require a new methodology to study and understand complex, emergent macroscopic phenomena. Coarse graining, a technique that originated in statistical physics, involves taking a system with many microscopic degrees of freedom and finding an appropriate subset of collective variables that offer a compact, computationally feasible description of the system, in terms of which the dynamics looks "natural". This paper presents the key ideas of the approach and shows how it can be applied to evolutionary dynamics.

1 Introduction

Our understanding of evolution has itself evolved. The journey started with Darwin and Mendel but it was only with the understanding of the structure of the DNA and the formulation of the central dogma of Molecular Biology in the 1950s and 1960s that the microscopic mechanisms of evolution could start to be unravelled. The central dogma postulates that DNA can be seen as a sort of read-only memory which encodes all the features and functionality of adult individuals. Through the process of transcription the information contained in the DNA would then be transferred into RNA. Then through the process of translation, this information would be carried by messenger RNA to the ribosomes. These would finally be responsible

R. Poli (✉)
School of Computer Science and Electronic Engineering, University of Essex, Colchester, UK
e-mail: rpoli@essex.ac.uk

C.R. Stephens
Instituto de Ciencias Nucleares, UNAM, Mexico City, Mexico
e-mail: stephens@nucleares.unam.mx

S. Cagnoni et al. (eds.), *Evolution, Complexity and Artificial Life*,
DOI 10.1007/978-3-642-37577-4_2, © Springer-Verlag Berlin Heidelberg 2014

for transforming the information into proteins. Thus, only the processes involved in reproduction could modify the DNA of an organism.

While of course this picture is a reasonable approximation of a particular function of the DNA and a particular way in which evolution can happen, over the last few decades biologists have discovered that there are many more mechanisms through which the DNA could be modified, even during the lifetime of an individual or of a cell, and thus there are many more mechanisms through which evolution is likely to have happened [1].

It is today clear that evolution is extremely complex. However, even if we stick with the 1970s view of it and we consider the super-simplified forms of evolution in the computer that were inspired by such a view and used within the field of evolutionary computation, there is still a huge amount of potential complexity in it. This is, however, hidden complexity, that we see only if we try to understand evolution at a deeper level: the level of theory.

What is a theory? A theory is a logically self-consistent framework for describing the behaviour of a related set of phenomena. It often originates from, or is supported by, experimental evidence. Thus, a theory is a systematic and formalised expression of previous observations that is predictive, logical and testable. Why is theory useful? A successful theory gives an intuitive *understanding* of the system being modelled, which permits one to deduce new consequences and explain phenomena. It allows quantitative *predictions*, albeit more often than not approximate, about the system.

Do we have a theory of evolutionary systems? Well, yes and no. In many areas of evolutionary computation and classical theoretical population genetics, we have well-defined, complete and precise mathematical frameworks. The models of population genetics and evolutionary algorithms (EAs) have a lot in common, and in some cases, it is even arguable that there has been more progress in modelling *natural* evolution in EA theory than in population genetics. However, making progress both with our understanding of evolutionary algorithms and with making predictions has been an exceedingly difficult task. In this chapter, we will try to illustrate the nature of the difficulties and show how a technique known as coarse graining has helped us make progress.

2 Physics and Probability Preliminaries

Let us start with some simple notions from physics. In many systems one can identify a minimum set of variables, called *degrees of freedom* (d.o.f.), which describe the state of the system. For example, the d.o.f. of a set of static marbles on a table would be the x and y coordinates of each marble. Similarly, the d.o.f. of the molecules of a gas are the x, y, z coordinates of each molecule and the components v_x, v_y, v_z of their velocities.

Note that the x and y of each marble would work as d.o.f. also if we glued the marbles together in some form of geometric arrangement, say a rectangle. However, now if we moved the rectangle, the marbles in the rectangle rigidly move together,

i.e., their trajectories are now constrained. Indeed, the position of each marble is known if we know the position of the rectangle. This has only three d.o.f.—the coordinates x_R and y_R of its centre of mass and its rotation θ_R on the plane.

The x and y coordinates of each marble are said to be the microscopic d.o.f. of the system, while the rectangle's d.o.f., x_R, y_R and θ_R, are what a statistical physicist would call the *effective degrees of freedom* (e.d.o.f.) of the glued marbles. We will give here a personal definition of e.d.o.f.: *a set of effective degrees of freedom for a system is a minimum set of variables which, at a given scale, naturally, possibly approximately, describe the states of a system for many practical purposes*. So, while the x, y, z, v_x, v_y, v_z of a gas molecule's state are the microscopic d.o.f. for a gas, in many practical cases, pressure and temperature are a set of (macroscopic) e.d.o.f. for the gas.

Naturally, all d.o.f. are effective to some degree. Nearly always, in the real world, any chosen set of d.o.f. (even the most microscopic and complete ones) provide only an incomplete representation of reality. When we chose to represent the marbles using their x and y coordinates we had made some assumptions: (a) the marbles' z coordinates are constant, (b) the marbles' rotations are either unimportant or unobservable, (c) the marbles are stationary (no velocities), (d) everything else, e.g., the marbles' temperature, colour, etc., is irrelevant. For some situations this set of d.o.f. is sufficient to represent the behaviour of the real system. However, it wouldn't be appropriate if someone could give a push to a marble.

Related to the notion of effective d.o.f. is the notion of coarse graining. *Coarse graining* means taking a system with many microscopic d.o.f. and finding an appropriate set of e.d.o.f. for it. How do we choose a good set of e.d.o.f.? There are some criteria: we want e.d.o.f. that offer a more compact, appropriate and computationally tractable description of the system, and in terms of which the dynamics looks "natural". (Often this naturalness manifests itself in terms of finding variables that are as independent and uncoupled as possible.)

Normally one describes a systems using d.o.f. for a reason: we want to understand how and why the state of the system changes over time. This is what a physicist would call the *dynamics of the system*. Also, we may want to describe special states, e.g., equilibria, where the state variables (d.o.f.) have particular relationships, e.g., the gas law $PV = RT$ where P is pressure, V volume, T temperature and R is the ideal gas constant. In coarse graining, generally, we pass from a description with one set of d.o.f., and corresponding interactions, to another, where both the e.d.o.f. and their effective interactions are different as we change scale, i.e., as we change from one set of e.d.o.f. to another. So, the dynamics and laws governing a system change as we change d.o.f..

Since a number of sources of randomness influence evolution, models of evolution will need to make use of probabilities. In particular we will use probability/event tree diagrams to model evolution. Tree diagrams allow us to see all the possible outcomes of an event and calculate their probability. Let us briefly discuss these tools.

We will first consider what tree diagrams can do for us using very simple examples. Let us start with modelling repeated coin tosses. Spinning a coin has

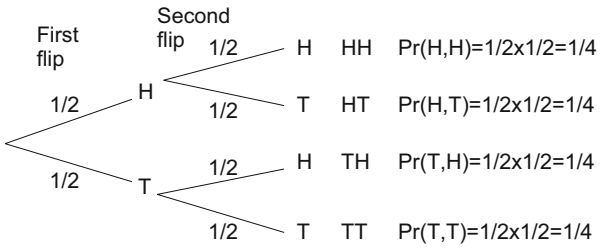

Fig. 1 Spinning a coin twice can be modelled using a tree diagram with two levels and four possible outcomes. The probability of an outcome of a double flip is the product of the probabilities $(1/2)$ encountered along the edges connecting the root node to the outcome

two outcomes: heads (H) or tails (T). Spinning it twice can be modelled using a tree diagram with two levels and four possible outcomes: HH, HT, TH and TT. In general, in a tree, we label each edge with the probability of an event following that edge and resulting in the corresponding outcome. The probability of an outcome (e.g., heads followed by another heads) is the product of the probabilities along the path connecting it to the root of the tree. If the coin is fair, the probabilities along each edge are all $1/2$ as shown in Fig. 1. The probability of the four outcomes of a double coin flip are the products of the probabilities encountered along the edges connecting the root node to an outcome: in this case $1/4$ for all outcomes.

Once the tree diagram is instantiated, one can use it to compute the probability of more complex events, such as the probability of having at least one head in two coin flips, by simply labelling as "success" all the relevant outcomes and adding together the probabilities of such "successful" outcomes.

Naturally, probability trees can also handle multiple outcomes, as in the case of drawing beads from a bag containing 2 blue, 3 red and 5 green beads, and, again, the product rule for computing the probability of outcomes applies. Also in this case, one can compute the probability of more complex events by simply adding up the probabilities of the relevant outcomes.

More generally, with probability trees we can find answers to questions such as: If we have a α_R (α_B, α_G) probability of getting a red (blue, green) bead in one draw, what's the probability of getting exactly n_R red, n_B blue and n_G green beads after M draws? For $M = 2$ we can use a tree diagram to answer the question. However, for bigger M and also for general values of α_R, α_B and α_G, we would really need a formula to find out the answers. Fortunately, the answer is given in probability textbooks. The repeated draw of beads from the bag follow a *multinomial distribution*. In other words,

$$\Pr(n_{\mathrm{R}}, n_{\mathrm{B}}, n_{\mathrm{G}}) = \binom{M}{n_{\mathrm{R}}, n_{\mathrm{B}}, n_{\mathrm{G}}} \alpha_{\mathrm{R}}^{n_{\mathrm{R}}} \alpha_{\mathrm{B}}^{n_{\mathrm{B}}} \alpha_{\mathrm{G}}^{n_{\mathrm{G}}}$$

where $\binom{M}{n_{\mathrm{R}}, n_{\mathrm{B}}, n_{\mathrm{G}}} = \frac{M!}{n_{\mathrm{R}}! n_{\mathrm{B}}! n_{\mathrm{G}}!}$ are multinomial coefficients.

Naturally, tree diagrams are useful also to model sequences of events that are not all of the same type. For example, we can alternate coin flips and bead draws, and find what's the probability of the event T,R,T,R from the corresponding tree. The only problem is that we just cannot use one of the multinomial distribution shortcuts to add up probabilities for us. Also, tree diagrams work even if events are not independent. For example, if we were performing bead draws but did not put the bead drawn back into the bag, the probabilities of drawing beads of different colour would change after each draw. Nonetheless, the probability of an outcome (e.g., *R*,*R*) is still the product of the probabilities along the path connecting it to the root.

Note that, in the case just described, in the second level of the tree we are performing a different kind of draw depending on the result of the first draw. We could be even more radical, and in fact consider a completely different set of events and outcomes for each outcome of the first draw. For example, we might draw another bead (with outcomes R, B and G) if the first bead drawn was red, we might flip a coin (with outcomes H and T) if the bead was blue and we might roll a dice (with outcomes 1, 2, 3, 4, 5 and 6) if the bead was green. While the exercise and the resulting tree (with its set of inhomogeneous outcomes) might seem odd, the product rule to compute the probability of outcomes would still apply.

3 Models of Evolutionary Algorithms

Modelling EAs means modelling the different events that take place during the creation of offspring, then modelling the iteration of such events which lead to the creation of a new generation and, finally, modelling the iterated construction of a generation to model a full run of the algorithm. We will do this in the following subsections.

3.1 Modelling the Genetic Operators

For simplicity, we will assume that we use a binary fixed-length representation and that offspring can be created by either the selection of one parent followed by mutation or the selection of two parents followed by crossover. More complex forms of creation can be modelled following the same principles.

The first question we need to answer is: What happens when an offspring is created in one particular generation? Irrespective of the genetic operators used, the creation of an offspring at a given time depends only on: what's in the population at that particular time (which is variable), the fitness function (which we will assume to be fixed) and the parameters of the EA, such as the population size (which we will also assume to be fixed). So, the thing on which offspring creation depends is the current population.

Table 1 Degrees of freedom *(left)* and possible configurations *(right)* for a population of three binary strings of length four

d_1	d_2	d_3	d_4
d_5	d_6	d_7	d_8
d_9	d_{10}	d_{11}	d_{12}

0	0	0	0	0	0	0	0	⋯	1	1	1	1
0	0	0	0	0	0	0	0		1	1	1	1
0	0	0	0	0	0	0	1		1	1	1	1

4,096

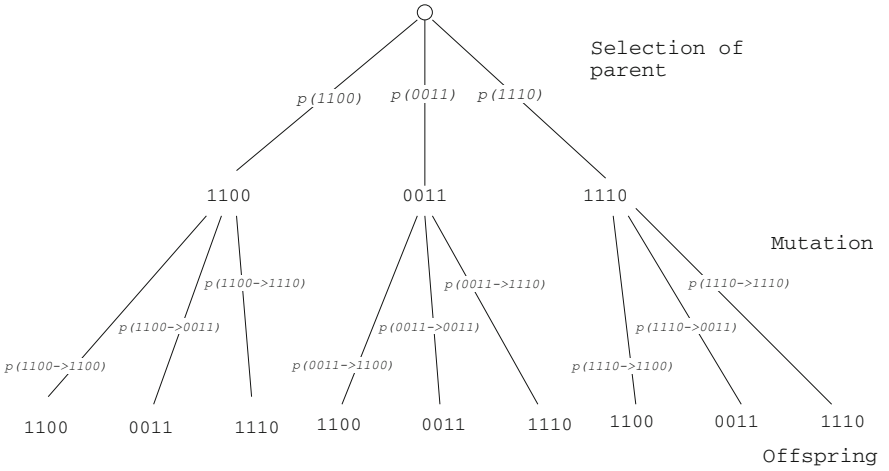

Fig. 2 Simplified tree model of the selection of a parent followed by its mutation

We will need to formalise this dependency in some way. The next question then is: What are the microscopic d.o.f. of a binary population? Clearly, the d.o.f. of a population are the bits in every individual of the population. For example, a population of three four-bit strings has 12 d.o.f. and there are $2^{12} = 4{,}096$ different configurations, as shown in Table 1.

Let us first consider the case of selection followed by point mutation. Suppose our current population is one of the configurations in Table 1(right), namely: $\{1100, 0011, 1110\}$. The creation of offspring by selection and mutation is represented by the tree diagram in Fig. 2, which is simplified for display purposes in that it assumes that only strings 1100, 0011 and 1110 can ever be generated (which, of course, isn't true).

Since point mutation acts on every bit independently, in the diagram, the probability (along the bottom edges) of mutating a string y to a string x is given by $p(y \to x) = p_m^{h(x,y)} \times (1 - p_m)^{\ell - h(x,y)}$ where $h(x, y)$ is the Hamming distance between x and y and ℓ is the string length. Let us further assume that we use fitness proportionate selection to select parents. In this form of selection, the selection probability for a string x, $p(x)$, is simply the ratio between the fitness of x and sum of the fitnesses of all individuals in the population.

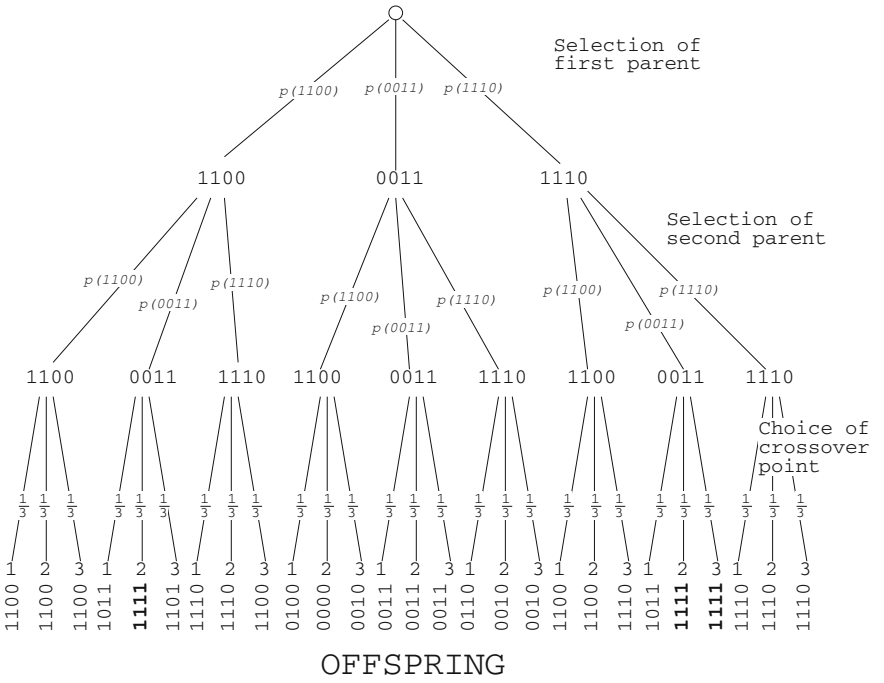

Fig. 3 Tree diagram for the selection of two parents followed by crossover

Then, if we know the mutation rate and the fitness function, we can turn the generic tree in Fig. 2 into a concrete probability tree. For example, if $p_m = 0.25$, then $p(1100 \to 1100) = 0.25^0 \times 0.75^4 = 0.316$, $p(1100 \to 0011) = 0.25^4 \times 0.75^0 = 0.004$, $p(1100 \to 1110) = 0.25^1 \times 0.75^3 = 0.105$, etc. So, all lower level edges have numerical probabilities associated with them. Also, suppose we are solving the OneMax problem,[1] then $f(1100) = f(0011) = 2$, $f(1110) = 3$ and the sum of fitnesses in the population is seven. So, $p(1100) = p(0011) = \frac{2}{7} = 0.286$ and $p(1110) = \frac{3}{7} = 0.429$. So, the probabilities in the upper part of the tree diagram are also defined. Of course, the multiplication rule still applies, but there are multiple paths leading to the same outcome (offspring). So, the offspring creation probabilities, α, are *sums* of products.

Let us now consider the process of generating offspring by selection followed by one-point crossover. The selection-crossover diagram for $\{1100, 0011, 1110\}$ is shown in Fig. 3. Clearly there are now three events taking place (two selections and one crossover), so the tree has three levels. The first two levels are exact copies of the first level of Fig. 2, since they simply represent the selection of the parents (with reselection allowed). As for the bottom level, since we use four-bit strings,

[1] In OneMax, fitness is the number of 1s in a bit string and the objective is to maximise that number.

one-point crossover can choose among three cut points. Each has a probability of $\frac{1}{3}$ of occurring.

If we look at the outcomes at the bottom of the tree, we can see that while we started from a population containing the three strings 1100, 0011, 1110, crossover can produce nine different strings: 0000, 0010, 0100, 0110, 1011, 1100, 1101, 1110 and 1111. Naturally, the sum-of-products rule still applies even if some probabilities in the tree are left unspecified. So, we can use the diagram to compute the creation probability $\alpha(x)$ for the nine outcomes (offspring strings): $\alpha(0000) = \frac{1}{3}p(0011)p(1100)$, ..., $\alpha(1111) = \frac{1}{3}p(1100)p(0011) + \frac{1}{3}p(1110)p(0011) + \frac{1}{3}p(1110)p(0011)$.

As for mutation, if we know the fitness, we can work out selection probabilities $p(x)$, and from these the creation probability for all strings that can be created in the next generation. Again, if we just focus on the outcomes, we can represent the process with a tree diagram with just one level.

It is traditional for the process of creation of offspring via selection and crossover to first require the selection of the two parents and then execute the crossover operation, which in turns requires selecting a random crossover point. It is, however, quite clear that choosing the crossover point is totally independent from the selection of parental types. So, one could reorder these operations without altering the outcome. For example, selecting crossover points before selecting parents doesn't affect results in any way. Naturally, this different way of ordering events leads to a different but equivalent tree diagram model.

3.2 Coarse Graining and Generalising Models

Having developed models for the process of creating individuals via selection-mutation and selection-crossover for the specific population $\{1100, 0011, 1110\}$ we may ask: What if we had the population $\{1100, 1100, 0011, 1110\}$ which contains two copies of the string 1100 instead of just one?

Of course, we could just follow the same steps as before and redevelop tree diagrams for the events associated with such a population. For example, Fig. 4 shows the diagram for offspring generation via selection-crossover for the new population. We should note, however, that there is a lot of duplication (identical sub-trees, identical outcomes) in this diagram with respect to the one for the original population (Fig. 3).

Let us try to simplify the tree a little. First, we should note that if we doubled some of the probabilities labelling some of the edges in the tree, we could obtain the entirely equivalent, but more compact model in Fig. 5.

We now see clearly that this is the same diagram as for $\{1100, 0011, 1110\}$ except four probabilities have doubled. What is this diagram trying to tell us? Why the factor 2? Note: 2 is also the number of copies of the string 1100 in the population. Is this a coincidence?

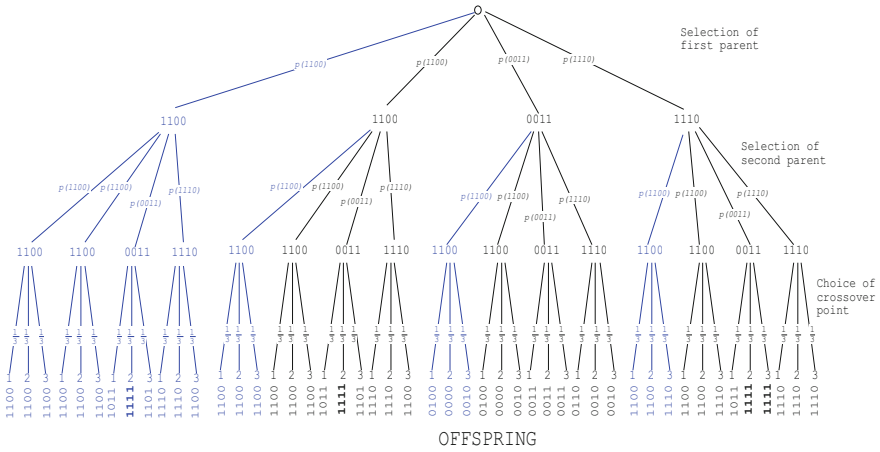

Fig. 4 Tree diagram model of the creation of offspring via selection-crossover for the population {1100, 1100, 0011, 1110}

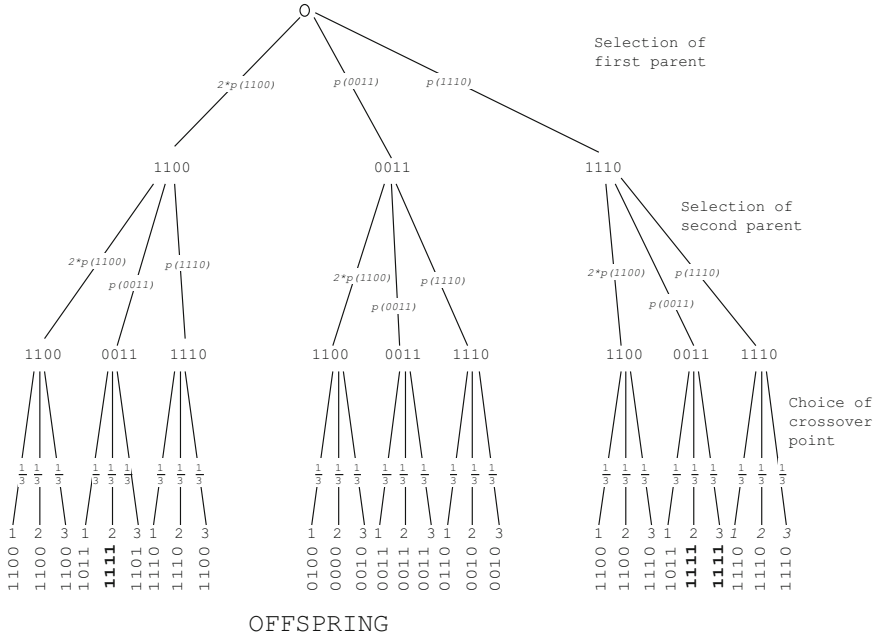

Fig. 5 More compact, but equivalent, version of the model in Fig. 4

Naturally, it is not a coincidence. The original tree diagram in Fig. 3 is *valid for both populations* provided we interpret the selection of first and second parent events as the *selection of first and second parental types* not of particular individuals and we interpret the $p(x)$'s as probabilities of selecting a particular *type*.

What is a type? The notion of type is best explained with an example. The population {A, B, B, C, A, A, B} contains seven individuals but only three distinct types: A, B and C. So, the selection probability for a type is defined as

$$p(\text{type } x) = \sum_{s \text{ of type } x \text{ in pop}} p(s) \;=\; \#(s \text{ of type } x) \times p(s),$$

which for fitness proportionate selection becomes

$$p(x) = \frac{\#\text{ individuals of type } x \text{ in population} \times f(x)}{\text{sum of fitnesses of all individuals in population}}.$$

With these changes of interpretation, the model of crossover in Fig. 3 works for all populations having any number of copies of 1100, 0011 and 1110, e.g., $\{1100, 1100, 1100, 0011, 0011, 1110, 1110, 1110\}$. Similarly we could coarse grain selection-mutation models.

A question that immediately comes to mind is: Would the model work for also the population $\{0011, 1110\}$ which contains zero copies of the string 1100? The answer is yes. There are extra outcomes in the tree in Fig. 3 which for the population $\{0011, 1110\}$ have a zero probability of occurring, but the model is still formally correct.

In other words, the probability of selection of types automatically adjusts for the number of copies (including 0) of a type. Thus, we could generalise the selection part of the model to any population if we added all possible string types of a given length as outcomes of the selection process. Naturally, only a small subset of outcomes would have non-zero probability for any given population.

If we also generalise crossover to strings of a generic length ℓ, we would get a *general model of the selecto-recombination operator* which is both independent from the particular population at hand and from the length of the representation. This is shown in Fig. 6. The same approach would produce a general selecto-mutation model.

Note that the models in Figs. 2 and 6 can be collapsed down to a tree with a single level if we consider the selection-mutation and selection-crossover processes as a single (composite) event, respectively. Naturally, as shown in Fig. 7, we need to use an appropriate set of probabilities to label the edges that lead to the outcomes (offspring), namely the quantities $\alpha(x)$ for all possible values of x.

Let us reconsider at this point the question of how we compute the creation probabilities α. The tree we have just defined has 2^ℓ nodes at the first level, $2^\ell \times 2^\ell$ at the second and $(\ell - 1) \times 2^\ell \times 2^\ell$ at the third level. So, for any realistic value of ℓ it is *immense*. Interestingly, however, it can have "only" 2^ℓ distinct outcomes (offspring). Since $2^\ell \ll (\ell - 1) \times 2^\ell \times 2^\ell$, we should expect to have to multiply and add an exponential number of probabilities to compute $\alpha(x)$ for each outcome.

Obviously, we cannot do this by hand. So, we introduce a function to help us do it: $\lambda(y, z, n, x) = 1$ if crossing over y and z at position n produces x, and 0

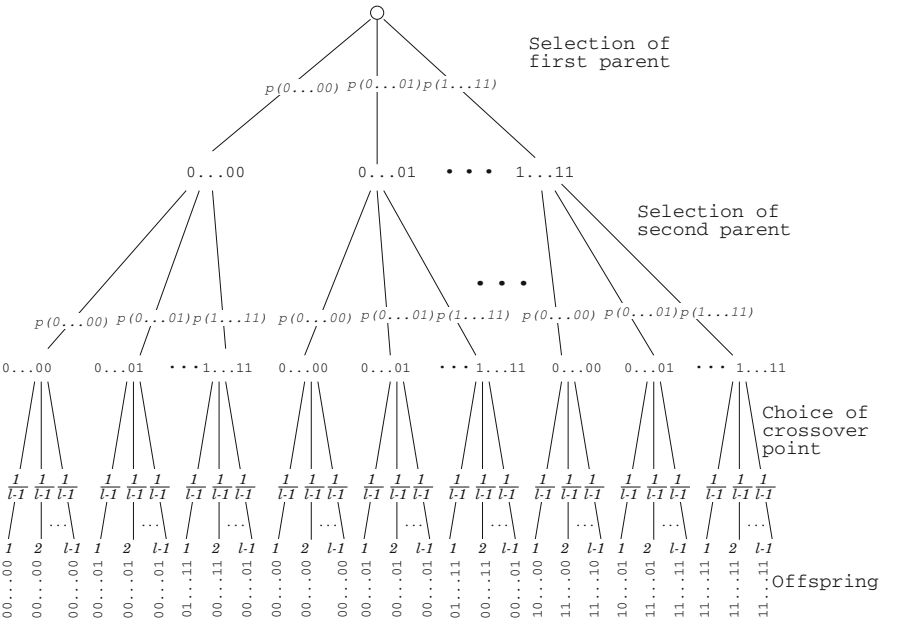

Fig. 6 General population- and length-independent model of the process of creating offspring via selection and recombination

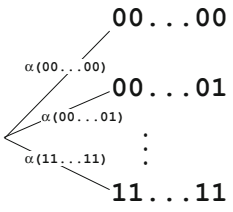

Fig. 7 The tree representation for a general selection-crossover process and a general selection-mutation process, represented as a single (*composite*) event

otherwise. With this, we can now write the creation probability via selection and crossover for a generic type x, in a general form:

$$\alpha(x) = \sum_{y\in\Omega}\sum_{z\in\Omega}\sum_{n=1}^{\ell-1}\lambda(y,z,n,x)\left(\frac{p(y)p(z)}{\ell-1}\right) \tag{1}$$

where Ω is the space of all possible strings of length ℓ.[2]

[2]Naturally, we would like to have an explicit form for $\lambda(y,z,n,x)$. It exists, but for now we will not look at it.

Note that Fig. 7 represents also a general model for the selection-mutation process, although now we have $\alpha(x) = \sum_{y \in \Omega} p(y)p(y \to x)$ where $p(y \to x)$ is the probability of mutation transforming an individual of type y to an individual of type x (which we have already computed).

3.3 Modelling One Generation

Now that we have models of the process of creating one offspring, we are in a position to start constructing models of the iterated application of the offspring-creation process, in view of modelling a generation.

What happens when an offspring is created in one particular generation? We have already seen that, irrespective of the genetic operators used, offspring creation is similar to drawing beads from a bag. Generally, the process is modelled in Fig. 7 for a *generic* operator or set of operators (where $\alpha(x)$ is the probability of creating individual x with the chosen operators). Naturally, the creation process will have many more than the three possible outcomes of the bead-draw process, but the principles at work are exactly the same. Then, what happens if we iterate the creation process, e.g., to create a full new generation? In a generational EA, *within a generation the α's, i.e., the offspring creation probabilities, are constant*. So, for a generation we have a probability tree diagram very much like the one for iterated bead draws as illustrated in Fig. 8 for a population of three individuals.

Naturally, in general, the general model in Fig. 7 would need to be used at each level of the tree diagram representing one generation. In either case, this time the outcomes of the process are populations. For example, for the model in Fig. 8 there are 27 outcomes:

1111	1111	1111	1111	1111	1111	1111	**1111**	1111	...	0000
1111	1111	1111	0000	0000	0000	1010	**1010**	1010		0000
1111	0000	1010	1111	0000	1010	1111	**0000**	1010		0000

Again, like for multiple bead draws, the multiplication rule applies: when we want to compute the probability of an outcome, we simply need to multiply the probabilities along the edges of the path that goes from the root of the tree to the outcome of interest. So, for example, the probability of the next generation being the eighth outcome (in boldface) is $\alpha(1111) \times \alpha(1010) \times \alpha(0000)$.

At this stage we note the possibility of a further coarse graining: coarse graining on positional symmetries. In other words, do we generally care about the differences between populations, such as the second and the fourth above, which have exactly the same strings but in a different order?

We might, if some genetic operator depends on position, e.g., if selection only allowed mating of neighbouring individuals. However, typically *genetic operators are position independent*, so we don't care about positional differences. Because these populations are effectively equivalent, we don't really need to distinguish them using the original set of d.o.f.. We only care about how many individuals of any

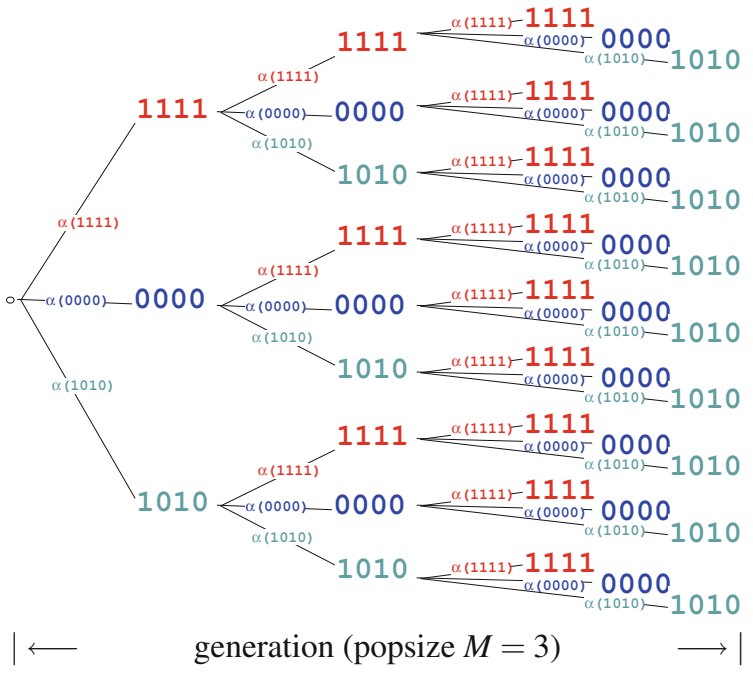

Fig. 8 Simplified tree diagram model of a generation for a population of three individuals

given *type* are present in the population. E.g., two of type 1111 and one of type 0000. So, we can define new (coarse grained) e.d.o.f.: using the number n_x of individuals of a given type x (for all x) as a representation. So, a population with two individuals of type 1111 and one of type 0000 can be represented as $n_{1111} = 2$, $n_{0000} = 0$ and $n_{1010} = 1$.

More generally, for $\ell = 4$ and a population of size M, our e.d.o.f. are

0000	n_{0000}	0100	n_{0100}	1000	n_{1000}	1100	n_{1100}
0001	n_{0001}	0101	n_{0101}	1001	n_{1001}	1101	n_{1101}
0010	n_{0010}	0110	n_{0110}	1010	n_{1010}	1110	n_{1110}
0011	n_{0011}	0111	n_{0111}	1011	n_{1011}	1111	n_{1111}

with the constraint $\sum_x n_x = M$.

Having now coarse-grained on positional symmetries, have we actually saved anything in terms of complexity? Simple counting arguments can show that with strings of length ℓ there are $\binom{M+2^\ell-1}{2^\ell-1}$ possible populations of M individuals. This is in general much smaller than the $2^{M\times\ell}$ populations we would have to consider if we did not coarse grain. So, the saving is huge.

Of course, as for the beads, if x_i is the i-th possible offspring, the probability that in the next generation one gets n_1 individuals of type x_1, n_2 of type x_2, etc., is just given by the multinomial distribution

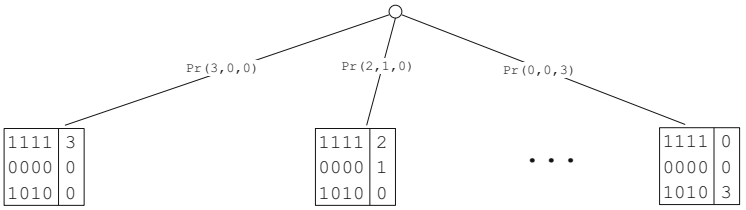

Fig. 9 Tree model of the process of creating a generation as a single *(composite)* event. (Here, for simplicity, we assume that only strings 1111, 0000 and 1010 can be generated. So, there are only ten possible outcomes)

$$\Pr(n_1, n_2, \cdots) = \binom{M}{n_1, n_2, \cdots} \alpha(x_1)^{n_1} \alpha(x_2)^{n_2} \cdots$$

Effectively this equation is a representation for the *dynamics* of the system in terms of the new e.d.o.f. n_x. With it, we can get (probabilistic) information about the next generation. For example, we know all the moments of the distribution of next-generation populations. For instance, the expected number of copies of any x_i in the next generation is simply $M\alpha(x_i)$.

As we did before with the α's, knowledge of $\Pr(n_1, n_2, \cdots)$ allows us to model a generation, which is the result of a series of offspring-creation events, as one (composite) generation-creation event, the outcomes of which are populations. A sample (simplified) tree diagram for the population $\{1111, 0000, 1010\}$ is shown in Fig. 9.

3.4 Modelling Runs

Having now "tamed" the complexity of the generation-creation process using the $\Pr(n_1, n_2, \cdots)$ and tree diagrams such as Fig. 9, we are now in a position to model entire runs.

It is clear that tree diagrams can be used also to model multiple generations and, thus, runs as illustrated in simplified form in Fig. 10, where we assumed that we start runs from a given (known) population. In the figure, different edges in the tree leading to the *same population* are labelled by *different probabilities*. The reason for this is that the action of the genetic operators (e.g., selection) depends on who is in the population, so the α's and consequently the probabilities $\Pr(n_1, n_2, \cdots)$ depend on it. So, in general we should expect $\Pr(3, 0, 0) \neq \Pr'(3, 0, 0)$ and similarly for most other labels.[3]

[3] As a result, we cannot use the multinomial distribution to predict the future over multiple generations.

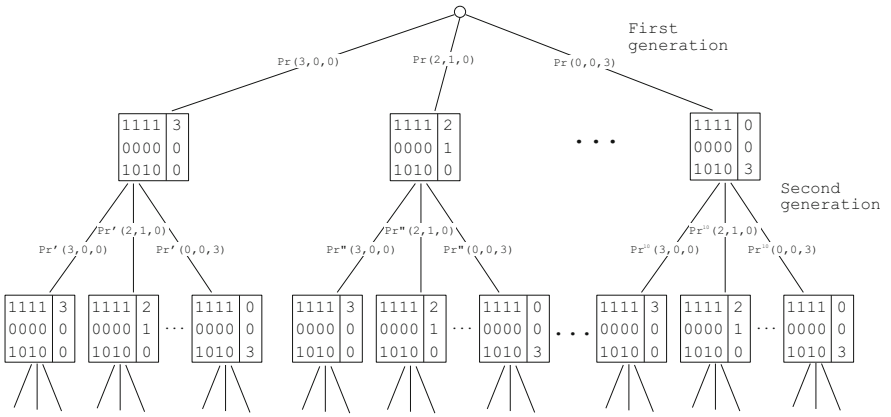

Fig. 10 Tree model of the process of running an EA for multiple generations

Of course, outcomes are now trajectories in population space. For example, if we ran an algorithm for three generations, an outcome could be

$$\text{trajectory} = \Big(\text{Initial population} \Rightarrow \begin{array}{|cc|}\hline 1111 & 1 \\ 0000 & 2 \\ 1010 & 0 \\ \hline \end{array} \Rightarrow \begin{array}{|cc|}\hline 1111 & 2 \\ 0000 & 1 \\ 1010 & 0 \\ \hline \end{array} \Rightarrow \begin{array}{|cc|}\hline 1111 & 3 \\ 0000 & 0 \\ 1010 & 0 \\ \hline \end{array} \Big)$$

Applying the multiplication rule to this sample trajectory we obtain

$$\Pr(\text{trajectory}) = \Pr(1,2,0) \times \Pr'(2,1,0) \times \Pr''(3,0,0)$$

where

$$\Pr(1,2,0) = \Pr\Big(\text{getting} \begin{array}{|cc|}\hline 1111 & 2 \\ 0000 & 1 \\ 1010 & 0 \\ \hline \end{array} \text{from the initial population} \Big),$$

$$\Pr'(2,1,0) = \Pr\Big(\text{getting} \begin{array}{|cc|}\hline 1111 & 2 \\ 0000 & 1 \\ 1010 & 0 \\ \hline \end{array} \text{from population} \begin{array}{|cc|}\hline 1111 & 1 \\ 0000 & 2 \\ 1010 & 0 \\ \hline \end{array} \Big),$$

$$\Pr''(3,0,0) = \Pr\Big(\text{getting} \begin{array}{|cc|}\hline 1111 & 3 \\ 0000 & 0 \\ 1010 & 0 \\ \hline \end{array} \text{from population} \begin{array}{|cc|}\hline 1111 & 2 \\ 0000 & 1 \\ 1010 & 0 \\ \hline \end{array} \Big),$$

are *conditional probabilities*. In other words, the probability on the edge from a population $\mathscr{P}$ to population $\mathscr{P}'$ is the probability of $\mathscr{P}'$ being the next generation *when we use the creation probabilities* α *computed for population* $\mathscr{P}$. We write this probability as $\Pr(\mathscr{P}' \mid \mathscr{P})$.

Now, let us imagine we numbered all possible populations: $\mathscr{P}_1$, $\mathscr{P}_2$, and so on. The probabilities of the edges between all possible pairs of populations could be

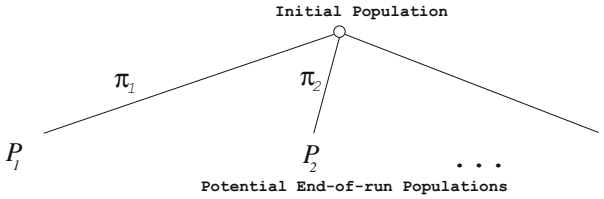

Fig. 11 Tree model of the process of performing a run of an EA as a single (*composite*) event

represented as a matrix $\mathbf{Q} = \left(\Pr(\mathscr{P}_i \mid \mathscr{P}_j)\right)$, which is effectively the *transition matrix for a Markov chain* whose states are all possible populations: $\mathscr{P}_1$, $\mathscr{P}_2$, and so forth. So, the probability of the population following a particular trajectory during a run is the product of a set of elements of $\mathbf{Q}$.

However, in many cases one is not particularly interested in the trajectory followed by a run, but, more simply, in the population obtained at the end of that run. In this case, there is a further opportunity for coarse graining: we could consider populations as outcomes of runs of the EA and ignore all other aspects of the dynamics that led to such end-of-run populations. For the properties of Markov chains, we can easily compute the probability of each such outcome. Indeed, the probability distribution, π_t, of the EA being in any state (having a particular population) at a future generation t is simply

$$\pi = \mathbf{Q}^t \pi_0$$

where π_0 is the initial probability distribution over states, which for a known initial population is simply a vector with one unitary component (representing the initial population) and zeros everywhere else.[4] So, if we know the initial population, again, we can treat the complex chain of events taking place in a run as one (composite) event which we can model with a tree diagram with only one level. This tree is represented in Fig. 11 where π_1, π_2, etc. are first, second, etc. component of the vector π, respectively.

Naturally, many EAs start from a random population, not a known population. It should, however, be clear by now, that the process of randomising the initial generation before starting a run simply adds an extra level to the tree-diagram model of runs in Fig. 10. This does not change at all the resulting model of runs as single composite events shown in Fig. 11. All that changes is the initial distribution over populations, π_0. So, the case of initial random populations, too, is covered by our analysis.

[4]This is Michael Vose's model for a genetic algorithm [2].

4 Coarse Graining and the Emergence of Schemata

So far we have already used coarse-graining a number of times: we coarse-grained over types when performing selection, we coarse-grained over positions in the population and, finally, we coarse-grained over population dynamics focusing only on end-of-run populations.

In theory, with **Q** one can compute everything that can be computed about the future of a run. For example, we could compute the probability of solving a problem in say 50 generations. In practice, however, **Q** is just too big for one to be able to create and use it. We can only study its properties mathematically. So, while the theory presented above is an exact theory, it is a theory that is hard to use to make predictions and to understand why an EA behaves the way it does. The problem is that the model and its e.d.o.f. are still too *microscopic* to show us the regularities of an algorithm. We need something else.

4.1 In Search of New Effective Degrees of Freedom

Earlier we expressed the creation probability via selection and crossover as in Eq. (1). It is now time to learn more about $\lambda(y, z, n, x)$.

Let us fix the type of interest as $x = 11$ ($\ell = 2$). From the selection-crossover event diagram[5] we find that $\lambda(y, z, n, x) \equiv 0$ except for

$$\lambda(10, 01, 1, 11) = \lambda(10, 11, 1, 11) = \lambda(11, 01, 1, 11) = \lambda(11, 11, 1, 11) = 1.$$

So, $\alpha(11) = p(10)p(01) + p(10)p(11) + p(11)p(01) + p(11)p(11)$, which includes only 4 terms out of the possible 16. $\alpha(11)$ can thus can be written as

$$\alpha(11) = (p(10) + p(11)) \times (p(01) + p(11)).$$

Is this factorisation a coincidence, or is the equation trying to tell us that $A = p(10) + p(11)$ and $B = p(01) + p(11)$ would lead to a more natural description of the creation process as $\alpha(11) = A \times B$? To answer these questions we will need to find (and work with) an explicit form of λ.

The function λ is 1 only if there is a match between the offspring's bits and the first parent bits before the crossover point *and* there is a match between the offspring's bits and the second parent bits after the crossover point. We can write this conjunction of multiple requirements as

$$\lambda(y, z, n, x) = \prod_{i \leq n} \delta(x_i = y_i) \prod_{i > n} \delta(x_i = z_i)$$

[5] For $\ell = 2$ there is only one valid crossover point ($n = 1$).

where x_i, y_i and z_i are the bits in x, y and z, respectively, and $\delta(\textbf{expression}) = 1$ if *expression* is true, and 0 otherwise.

Substituting this description of λ into Eq. (1) yields

$$\alpha(x) = \sum_{y\in\Omega}\sum_{z\in\Omega}\sum_{n=1}^{\ell-1}\left(\prod_{i\leq n}\delta(x_i = y_i)\prod_{i>n}\delta(x_i = z_i)\right)\left(\frac{p(y)p(z)}{\ell-1}\right).$$

We can then reorder the calculation in an interesting way:

$$\alpha(x) = \sum_{n=1}^{\ell-1}\frac{1}{\ell-1}\left(\sum_{y\in\Omega}p(y)\prod_{i\leq n}\delta(x_i = y_i)\right)\times\left(\sum_{z\in\Omega}p(z)\prod_{i>n}\delta(x_i = z_i)\right)$$

That is, like $\alpha(11) = A \times B$, also $\alpha(x)$ can be expressed in a simpler form:

$$\alpha(x) = \sum_{n=1}^{\ell-1}\frac{1}{\ell-1}A_n \times B_n,$$

for $A_n = \sum_{y\in\Omega} p(y)\prod_{i\leq n}\delta(x_i = y_i)$ and $B_n = \sum_{z\in\Omega} p(z)\prod_{i>n}\delta(x_i = z_i)$. It is then natural to ask: What are the factors A_n and B_n? Why do things look so much simpler if we calculate α in this way?

4.2 Coarse Graining from Types to Sets

The action of the δ's in A_n and B_n is to zero some terms in the corresponding summations over Ω, i.e., they *limit* the range of the summations. That is

$$A_n = \sum_{y\in L_n(x)} p(y) \quad \text{and} \quad B_n = \sum_{z\in R_n(x)} p(z),$$

for an appropriate choice of the two sets $L_n(x)$ and $R_n(x)$, namely $L_n(x) = \{y \in \Omega : y_1 = x_1, \cdots, y_n = x_n\}$ and $R_n(x) = \{z \in \Omega : z_{n+1} = x_{n+1}, \cdots, z_\ell = x_\ell\}$. This suggests that there is a further level of coarse-graining that we can do to tame the complexity of evolution: moving from types to sets of types.

We can easily extend the definition of $p(x)$ *from types to sets* as follows:

$$p(A) = \Pr\{\text{selecting a individual of a type belonging to set } A\}.$$

Because all events in A are mutually exclusive $p(A) = \sum_{x\in A} p(x)$. Thus, we can express $A_n = p(L_n(x))$, $B_n = p(R_n(x))$, and

$$\alpha(x) = \sum_{n=1}^{\ell-1}\frac{1}{\ell-1}p(L_n(x)) \times p(R_n(x)).$$

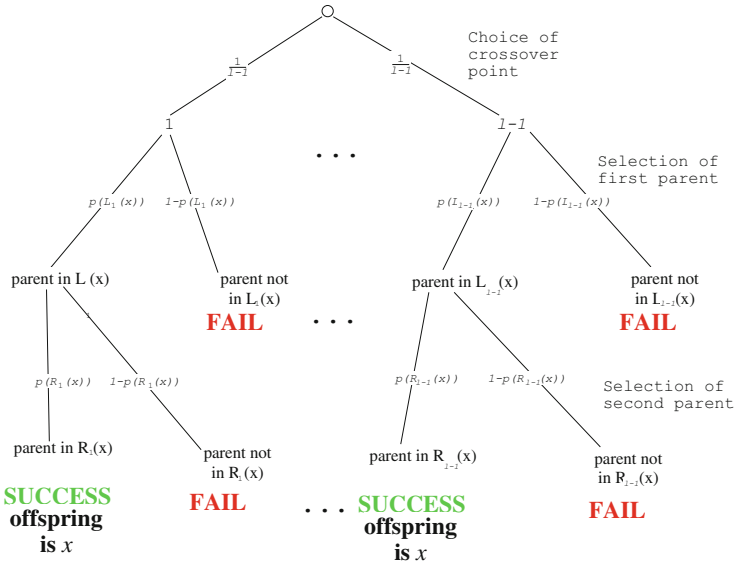

Fig. 12 Tree model of the selection-crossover process coarse-grained using sets of types

Let us analyse this result. We see that the probability of creating an offspring of type x is now decomposed as a sum of products of three probabilities: the probability of choosing a particular crossover point $\frac{1}{\ell-1}$, the probability of selecting a first parent belonging to the set $L_n(x)$ and the probability of selecting a second parent belonging to the set $R_n(x)$. Note that these two sets depend on both x and the choice of crossover point n. This suggests that the most coarse-grained tree diagram we can construct to represent these events is one that starts with the choice of a crossover point at the first level of the tree. Then it focuses on different kinds of selection events depending on the chosen crossover point, thereby forming a hybrid hierarchy of the same kind considered at the end of Sect. 2. The different kinds of selection are the selection of types belonging to the sets $L_n(x)$ for the second level of the tree and the sets $R_n(x)$ for the third level of the tree. Naturally at every level we must consider not only just the positive outcome (the selected type is in the set) but also its corresponding negative one (the type is not in the set).

The resulting tree diagram is shown in Fig. 12. Note how much smaller and simpler than the original in Fig. 6 this is. There are only $O(\ell)$ nodes in it as opposed to the original $O(2^\ell)$. Clearly, $L_n(x)$ and $R_n(x)$ are really good e.d.o.f. for describing the creation of instances of x. But what are these sets?

4.3 Schemata as Effective Degrees of Freedom

Holland [3] introduced the notion of *schema* as a tool for analysing EAs. A schema is a set of individuals represented with a particular pattern: a string of symbols from

the alphabet $\{0, 1, *\}$. The semantics of the symbols is this: (a) all individuals in the schema must have the specified pattern of 0s and 1s within them, and (b) the $*$ symbols mean we "don't care" to check the corresponding bit in the individuals. For example, the schema $11**$ represents all strings of length 4 which start with 11, i.e., $11** = \{1100, 1101, 1110, 1111\}$.

With this notion in hand, it is now easy to see that for a fixed x and n, the two sets $L_n(x)$ and $R_n(x)$ are particular *schemata*, namely $L_n(x) = x_1 \cdots x_n * \cdots *$ and $R_n(x) = * \cdots * x_{n+1} \cdots x_\ell$. Hence

$$\alpha(x) = \sum_{n=1}^{\ell-1} \frac{1}{\ell - 1} p(x_1 \cdots x_n * \cdots *) p(* \cdots * x_{n+1} \cdots x_\ell).$$

In other words, schemata are the right type of coarse-graining to represent the operation of creating an individual of a particular type through selection and crossover. Note, however, that there is a slight asymmetry in this equation: we have used schemata as effective d.o.f. for the right-hand side, but types as e.d.o.f. for the left-hand side. Could we coarse grain creation events even further by extending the interpretation of the domain of α from types to *sets of types*? If A is a schema, i.e., $A = s_1 s_2 \cdots s_\ell$ with $s_n \in \{0, 1, *\}$, and we define $\alpha(A) = \sum_{x \in A} \alpha(x)$, it is possible to prove [4] that this coarse graining leads to an equation of exactly the same form as that for $\alpha(x)$, namely:

$$\alpha(s_1 s_2 \cdots s_\ell) = \sum_{n=1}^{\ell-1} \frac{1}{\ell - 1} p(s_1 \cdots s_n * \cdots *) p(* \cdots * s_{n+1} \cdots s_\ell).$$

This reveals the hierarchical nature of creation events across multiple generations. For example, the probability of creating individuals of type 111 at generation t is determined by the selection probabilities of individuals of the sets $1**$, $*11$, $11*$ and $**1$ at that generation. These selection probabilities depend not only on the fitness function but also on the number of individuals within each set at generation t. These were created in the previous generation, $t - 1$. Their distribution is entirely determined by the probability of creating individuals in each sets at generation $t-1$. For example, the number of individuals in $*11$ depends on $\alpha(*11)$ at generation $t - 1$. In turn $\alpha(*11)$ is controlled by the individuals in sets $*1*$ and $**1$ at generation $t - 1$. Their number is stochastic, but, of course, depends on $\alpha(*1*)$ and $\alpha(**1)$ at generation $t - 2$.[6]

[6]Note that $\alpha(s_1 s_2 \cdots s_\ell)$ becomes particularly simple when all $s_i = *$ except one. In that case it is easy to verify that $\alpha(s_1 s_2 \cdots s_\ell) = p(s_1 s_2 \cdots s_\ell)$.

5 Conclusions

Here we have shown that what makes even the simplest forms of evolution so complicated to analyse mathematically is the explosive number of possible outcomes of each operation or sequence of operations. We have also shown that coarse graining has helped us formulate more intuitive models of the dynamics of EAs, thereby taming their complexity, at least to some degree.

Naturally, many people have attempted to model EAs for many years. So, different models and different types of coarse grainings have been used. For example, the models presented here have been extended to variable-length strings, to non-binary alphabets, to more general forms of crossover and mutation, to tree-like structures, to diploidy/polyploidy and multiple chromosomes, to more recently discovered genetic operations (inversion, transposition, gene duplication, gene deletion, etc.), etc. [5–8]. Also, there are systems where the natural e.d.o.f. are Fourier (Walsh) modes and systems that can be characterised by interpreting the crossover operation as a low-pass filter. It is also possible to apply the Renormalisation Group [9] to model EAs. Also, search in continuous spaces can be modelled using the finite element method (coarse-graining the states of the system).

There are, however, also a number of still unresolved issues including, for instance, deriving convergence proofs using coarse-grained variables, deriving problem-difficulty indicators based on such variables, relating no-free-lunch theory to coarse-grained models, and many others.

While coarse graining the d.o.f. and dynamics of evolution is difficult, most of what has been achieved for EAs is relevant for biological evolution, too. For example, the key notion of schema and the hierarchical nature of creation via crossover has been almost completely neglected in population genetics. Much might be learnt about natural evolution by applying such notion.

References

1. Shapiro, J.A.: Evolution: A View from the 21st Century. FT Press, Upper Saddle River (2011)
2. Vose, M.: Modeling simple genetic algorithms. In FOGA-92, Foundations of Genetic Algorithms, pp. 24–29. Vail, Colorado, (1992)
3. Holland, J.H.: Adpatation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
4. Stephens, C.R., Waelbroeck, H.: Schemata evolution and building blocks. Evol. Comp. **7**, 109–124 (1999)
5. Poli, R., Stephens C.R.: Understanding the biases of generalised recombination: Part I. Evol. Comput. **14**(4), 411–432 (2006)
6. Poli, R., Stephens C.R.: Understanding the biases of generalised recombination: Part II. Evol. Comput. **15**(1), 95–131 (2007)
7. Poli, R., McPhee, N.F.: General schema theory for genetic programming with subtree-swapping crossover: Part I. Evol. Comput. **11**(1), 53–66 (2003)
8. Poli, R., McPhee, N.F.: General schema theory for genetic programming with subtree-swapping crossover: Part II. Evol. Comput. **11**(2), 169–206 (2003)
9. Kadanoff, L.P.: Statistical Physics: Statics, Dynamics and Renormalization. World Scientific, Singapore (2000)

Part II
Biological Modeling

Models of Gene Regulation: Integrating Modern Knowledge into the Random Boolean Network Framework

Christian Darabos, Mario Giacobini, Jason H. Moore, and Marco Tomassini

Abstract Kauffman's random Boolean networks are abstract, high level models for dynamical behavior of gene regulatory networks. They simulate the time-evolution of genetic regulation within living organisms under strict conditions. The original model, though very attractive by its simplicity, suffered from fundamental shortcomings unveiled by the recent advances in genetics and biology. Using these new discoveries, the model can be improved to reflect current knowledge. Artificial topologies, such as scale-free or hierarchical, are now believed to be closer to that of gene regulatory networks. We have studied actual biological organisms and used parts of their genetic regulatory networks in our models. We also have addressed the improbable full synchronicity of the event taking place on Boolean networks and proposed a more biologically plausible cascading scheme. Finally, we tackled the actual Boolean functions of the model, i.e. the specifics of how genes activate according to the activity of upstream genes, and presented a new update function that takes into account the actual promoting and repressing effects of one gene on another. Improved models demonstrate the expected, biologically sound, behavior of previous GRN model, yet with superior resistance to perturbations. We believe they are one step closer to the biological reality.

C. Darabos (✉) · J.H. Moore
Computational Genetics Laboratory, Geisel School of Medicine at Dartmouth College, Hanover, NH 03755, USA
e-mail: christian.darabos@dartmouth.edu; jason.h.moore@dartmouth.edu

M. Giacobini
Computational Epidemiology Group, Department of Veterinary Sciences, and Complex Systems Unit, Molecular Biotechnology Center, University of Torino, Italy
e-mail: mario.giacobini@unito.it

M. Tomassini
Information Systems Department, Faculty of Business and Economics, University of Lausanne, Switzerland
e-mail: marco.tomassini@unil.ch

S. Cagnoni et al. (eds.), *Evolution, Complexity and Artificial Life*,
DOI 10.1007/978-3-642-37577-4_3,

1 Genes

Life's information is encoded in *genes* [1,2]. In modern biology, a gene is defined as the basic unit component encoding the heredity in living organisms. Genes contain the genetic information to build and maintain the cells of an organism. A gene is made of a sequence of *nucleic acid* (usually *Deoxyribonucleic acid* or *DNA*). Nucleic acids are macromolecules formed by the basic *nucleotides*: *adenine* (A), *cytosine* (C), *guanine* (G), and *thymine* (T). This sequence of DNA is composed of both *coding* parts, i.e. the actual information on what the gene does, and *non-coding* parts that determine whether the gene is *active*, i.e. *expressed* or not. This second part is also called the *regulating* part. Finally, there is a certain amount of extra DNA that is neither coding nor regulating.

To clarify, a *chromosome* is an organized form of DNA encoding several genes, along with regulating information and other non-coding sequences of *nucleotides*. Figure 1 gives an idea as to which elements of the genetic material is a building block of the next.

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

The molecules resulting from gene expression, whether RNA or protein, are known as gene products and are responsible for the development and functioning of all living things. They are also a central element of the regulation of downstream genes later during the life of the cell (see Fig. 2). The physical development and phenotype of organisms can be thought of as a product of genes interacting with each other and with the environment [3]. Some proteins serve only to activate other genes, and these are the *transcription factors*. By binding to the promoter region at the start of other genes transcription factors activate a gene. On the other hand, some transcription factors are inhibitory.

2 Genetic Regulatory Networks

In recent years, high throughput sequencing techniques, such as microarrays, have allowed biologists not only to sequence the entire genome of living organisms but also to shed some light on the interactions between these genes and how they regulate each other. Nevertheless, the regulation aspect between genes can

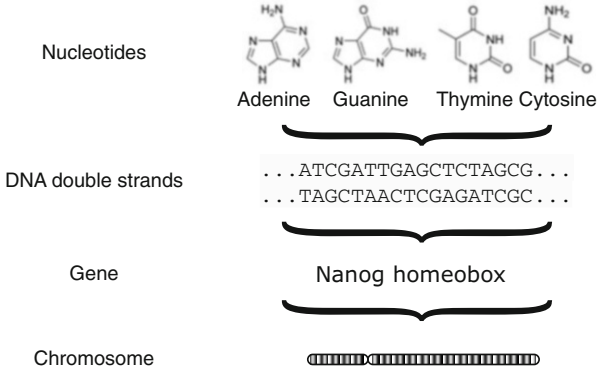

Fig. 1 Genetic material: how they all fit together

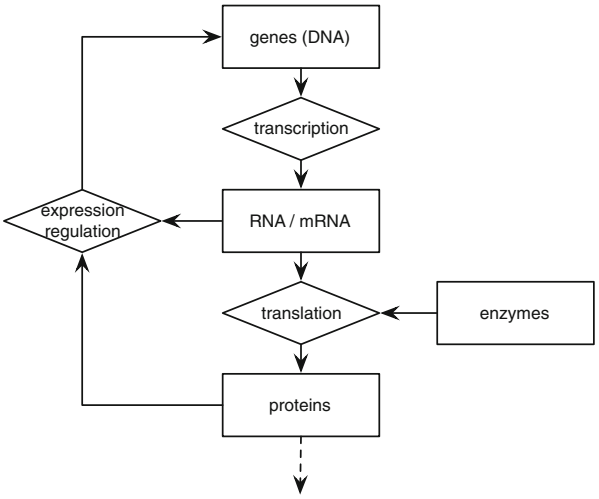

Fig. 2 Gene regulation: how they regulate each other and themselves

oftentimes only be inferred by gene *co-expression* data, but not directly witnessed. In a few words, co-expression and co-regulation data compare the expression level of genes at consecutive time intervals, and these time-series are analyzed to find emerging patterns in expressions of genes, which in turn reveals possible regulatory interactions amongst genes.

Gene regulatory networks are formed by genes, messenger RNA (mRNA), and proteins. The interactions between these elements include transcription, translation, and transcriptional regulation [4]. Usually, vertices represent genes and directed edges are the regulating influence (promotion or inhibition) of a gene on another via a protein or an mRNA sequence, as described above. These networks have particular structural topologies that help maintain the stability of the system, while allowing it

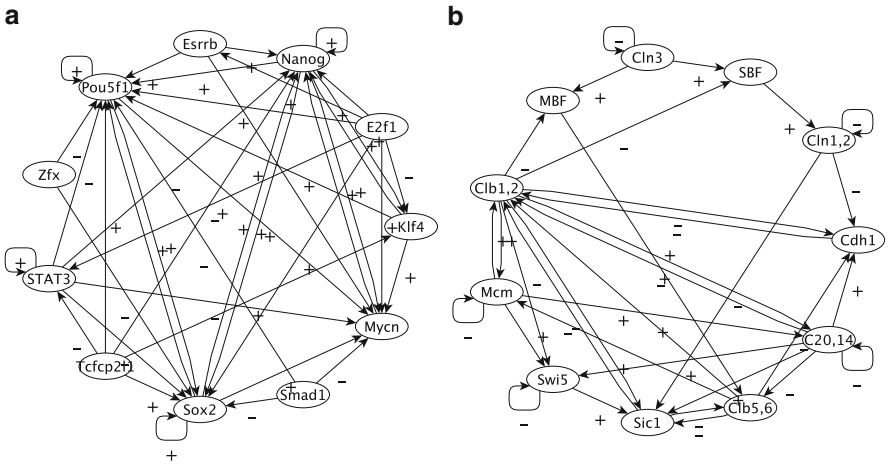

Fig. 3 Genetic Regulatory Networks. A representation of (**a**) the transcriptional regulatory network in ES cells and (**b**) the yeast cell-cycle regulatory network. Arrows point from transcription factor to the target gene. *Plus* signs "+" (respectively, *minus* signs "−") represent activating (respectively, repressing) links

to evolve. Figure 3 below shows two genetic regulatory sub-networks of biological organisms: yeast and mouse stem cell.

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

3 Modeling Genetic Regulatory Networks

Models, as mentioned earlier, are simplified representations of the reality submitted to constraints and conditions. In the case of genetic regulatory networks there are essentially four types of models [5]. Boolean networks that we will describe in detail in the next section, and three more:

- *differential equations systems* used to describe the reaction kinetics of the constituent parts. These models usually involve an in-depth understanding of the temporal variation of the concentration of the network's substances. The functions are ultimately derived from basic principles of chemical or enzymatic kinetics [6];
- *continuous networks* are an extension of the Boolean model described below, only this time the genes expression level is assumed to be a continuous function

of time. It has been argued that using a continuous representation captures several properties of gene regulatory networks not present in the Boolean model [7];
- *stochastic gene networks* are GRNs models reflecting recent experimental results [8, 9] hinting that gene expressions are stochastic processes. Some formalisms of this phenomenon have been proposed [10], and several works on single genes and small synthetic networks have been conducted [11–13].

The structural topology of the GRNs, and consequently of models thereof, is still open to discussion, but one property has been agreed by all parties that GRNs are sparse networks and that the mean number of upstream-regulator per gene is less than two [14]. In the early random Boolean network model [15], the network topology was random. Later, more complex network topology with different input and output degree distributions were proposed [16] (see next section).

Another subject of discord in the community is the timing of events in the models, more specifically, the fact that most models assume that the regulation, activation, and expression of the genes is taking place instantaneously and that these phenomena take the same time for all genes. Although the order of magnitude is comparable amongst, biologists feel this approximation might be a real weakness of the models. Alternate models were proposed where these reactions were delayed reactions in order to account for the time it takes for the entire process to be complete [17].

4 Random Boolean Networks

Introduced by Kauffman in the late 1960s, Random Boolean Networks (RBNs) are early models of genetic regulatory networks [15]. RBNs have been studied in detail by analysis and by computer simulations of statistical ensembles of networks and they have been shown capable of surprising dynamical behavior. An excellent review on the topic can be found in [18].

In Kauffman's RBNs (known today as Classical RBNs) with N nodes, a node represents a non-discriminate gene and is modeled as an Boolean *on/off* device, meaning that a gene can only either be expressed if it is on (1), and it is not expressed otherwise (0). Each gene receives exactly K randomly chosen inputs from other genes (see Fig. 4).

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

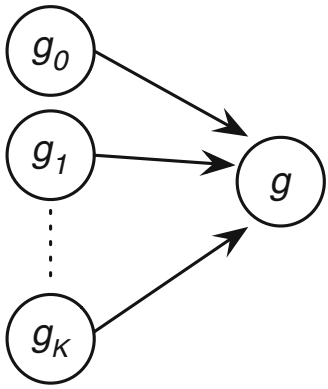

Fig. 4 Gene interaction in RBN: the activation state of genes $g_1, g_2, \ldots, g_K$ have an influence on gene g

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

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

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

A basin of attraction is made of three types of configurations: *garden-of-Eden*, *transitory*, and *attractor* states. Garden-of-Eden states cannot be reached by the dynamics of the systems itself, it can only be set as an initial configuration. Transitory states are traversed only once. Finally, attractor states are part of the cycle the system goes through once it has relaxed to stability.

Figure 5 shows the possible time-evolution of an example of RBN of size $N = 4$ and $K = 3$. This system has therefore $2^4 = 16$ possible configurations. In this example, we do not specify the Boolean functions, but instead show how the RBN transitions from any possible state. Garden-of-Eden states are represented in dark

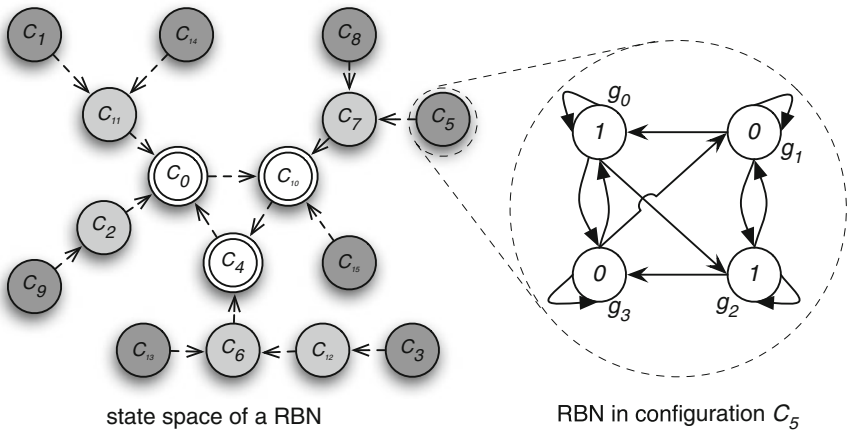

Fig. 5 State-space of an RBN. An example with $N = 4$, and all $2^4 = 16$ states of the state-space belong to the same attractor of length three

gray, transitory states are light gray and double lines identify the attractor states. In this particular example, the entire state-space belongs to the same basin of attraction leading to the unique attractor of this specific system. One can assume that if the Boolean function were different, the entire state-space transitions would be completely different as well.

RBN systems evolve dynamically over time in either the *ordered regime* or *ordered phase* or the *chaotic regime/phase*. The regime in which the RBN has been set can be identified according to the proportion of nodes that are actively participating in an attractor by flipping their states "often." In other words, assume that we can define two categories for the nodes of a system in an attractor: *frozen* and *twinkling* [19]. Frozen nodes are those whose state remains unchanged for a long time, say fifty time-steps. On the contrary, twinkling ones change their state frequently. In the ordered regime, the proportion of frozen nodes grows linearly with the network's size N, and a vast majority of the nodes are frozen. In the chaotic regime a majority remain twinkling. Finally at the critical regime, or so-called edge-of-chaos, the number of twinkling and frozen nodes is comparable. Another critical feature distinguishing the ordered form the chaotic regime is that in the first one, the length of the attractors scales as a linear or superlinear [18] function with the size of the network, whereas in the chaotic regime, it grows exponentially.

In Kauffman's original RBN systems, the edge between order and chaos is achieved when $K = 2$ and the bias $p = 0.5$. Kauffman speculates that living organisms operate at this *critical regime*, at the edge between the ordered and the chaotic phase. This condition helps organisms to achieve a trade-off between the stability of order and the robustness of chaos.

5 Extension of Random Boolean Networks

With the considerable advances in molecular biology that we have witnessed within recent years, some aspects of the original RBN have been questioned. For each of these aspects, one or more new models, or adaptations of the RBN model, have been proposed. Without discussing the Boolean state of expression, for which continuous models have been mentioned at the beginning of this chapter, in the following subsections, we mention a few.

5.1 The Topological Structure

As more and more details are unveiled about GRNs, the certitude is that their structure is not that of a random graph. Recent findings hint to a network topology with asymmetrical input and output degree distributions. Indeed, Aldana proposes an RBN model with scale-free topologies, that is, with a *power-law* output degree distribution and a *Poisson* or *normal* input degree distribution [16, 20]. This new model shows singular robustness that can directly be attributed to the topological properties of the underlying network [21].

In this work we focus on the structural and dynamical aspects of Boolean models for GRNs. Indeed, we propose the use of *generalized Boolean networks*, a broadening of the random Boolean model, where the topology does not have to be based on a random graph. Instead, we apply the concepts coined by Aldana and generate networks of the scale-free type as substrate for our Boolean models. These scale-free Boolean networks (SFBNs) have a long-tailed power-law output degree distribution and a Poisson-like input degree distribution, close to that of random graphs. These topologies are believed to be much closer to biological reality, where a handful of genes (i.e., hubs) produce proteins that regulate a large number of genes, and most of the genes only have an extremely limited regulating effect (i.e., leaves). Nevertheless, in order for a network to be scale-free, its degree distribution ought to cover several orders of magnitude, and Aldana's original ones were limited to a maximum of 19 genes. In our case, we build SFBNs with a maximum of 200 genes. Although still rather small, networks of this size approach power law behavior in RBNs but it is already very demanding in computational resources. Therefore, we had to resort to statistical sampling. In [22], we have studied the possibility of using sub-networks of GRNs of biological organisms. We faced several challenges when selecting candidate GRNs. The portions we needed had to be as self-standing as possible, with minimal external output, and the confidence in the gene interaction and their kind had to be acceptable. We finally opted for two portions of GRN, one of the yeast cell-cycle and one of the mouse embryonic stem cells (see Fig. 3). The straightforward use of these networks is to replace the nodes by Boolean models of gene expression values, and use them as Boolean model. Although using real-life biological GRN structures overcomes in a new way the topological flaw of the original RBN model, it raises a new problem. Until now, the critical regime was

achieved by tuning the gene expression probability p in the Boolean function, the network average degree, or the power-law exponent. These latter parameters are now fixed, and as consequence we have to tune the parameter p in order to guarantee the correct critical behavior. Thus, we can rephrase the problem into an optimization one, by changing p and minimizing a distance (that we call criticality distance) between the dynamical regime of our system (measured empirically by means of the so-called Derrida plot [30]) and the critical behavior itself. This procedure allowed us to tune the p value so that our systems operate indeed in the critical regime.

5.2 The Synchronicity

When considering biological phenomena, the synchronicity of the events taking place in organisms is a questionable assumption. Nevertheless, it was an understandable simplification of Kauffman's original model. More recently, asynchronous models have been proposed and display dynamical behaviors that are in agreement with previous results, yet more biologically interesting [23, 24].

As a second improvement we have fitted the model with is a more biologically plausible timing of events. Indeed, as argued in the previous chapter, the full synchronicity is clearly a biological improbability. Thanks to microarray experiment data, biologists believe that the activation of a gene induces a change of activation status in a subset of genes, and so on. Therefore, the regulating effect of genes could tend to impact regions of the genetic regulatory network, instead of the whole system simultaneously. This can be seen as a cascading effect, and therefore we introduce a novel update timing we call Activated Cascade Update (ACU). This update scheme is explained in detail in [25], and mimics the phenomenon witnessed in real-life. The dynamical behavior and robustness of Boolean networks with both random and scale-free topologies using ACU are compared to those of synchronous classical RBNs and SFBNs. We run extensive simulations and thoroughly study all different scenarios. Results detailed in the article support the relevance of our new, more biologically relevant model.

5.3 The Instantaneity

In real-life, gene activation time is in the order of seconds or minutes, protein decay time is between minutes and hours, and those time are dependent on the gene or protein in question. Approximating those times to instantaneous happenings is a gross oversight of biological reality. On the other hand, in order for time delays to actually be an asset to the model, a deeper understanding of the biochemical properties of genes, proteins, and mRNA is crucial, and we only now begin to have access to these data. Several models account for the time it takes biochemical reactions to occur, notably by taking an arbitrary chosen number of time-steps to refresh a gene's expression state [17].

5.4 The Random Boolean Update Function

Although the exact combined effect of the proteins and/or mRNA is still unclear, and thus the actual Boolean update function is unknown, it is safe to assume that these functions are not random. We already have mentioned stochastic networks as an alternative. Another interesting scheme was suggested that takes into account the actual promoting or repressing effect of genes known today, and combining them in an additive way [26].

We worked on closing another gap of Kauffman's original model by proposing an alternative Boolean function that is closer to current biological knowledge. From the literature and our collaboration with biologists, we were able to leverage the extra information in the studied GRNs presented above. Indeed, the yeast and stem cell partial GRNs we used not only provided us with the existence of gene on gene influence, but also their kind: *promoting* or *repressing*. We propose a Boolean function that adds the combined effect of the genes and uses a threshold T value for gene activation or deactivation, the Activator Driven Additive (ADA) function detailed in [22]. In every case we were able to determine a T value that makes our systems critical. Results analyzed in the article are very encouraging, both in terms of the kind of attractors found, that agree with our expectations according to the regime, and in terms of robustness, which has not suffered from the biological input to the model.

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

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

6 Biologically Inspired Faults

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

Kauffman defines one type of perturbation to RBNs as "gene damage" [19], that is the transient reversal of a single gene in the network. These temporary changes in the expression of a gene are extremely common in the normal development of an organism. The effect of a single hormone can transiently modify the activity of a gene, resulting in a growing cascade of alterations in the expression of genes influencing each other. This is believed to be at the origin of the cell differentiation process and guides the development.

The effect of a damaged gene can be measured by the size of the avalanche resulting from that single gene changing its behavior from active to inactive or vice versa. The size of an avalanche is defined as the number of genes that have changed their own behavior at least once after the perturbation happened. Naturally, this change of behavior is compared to an unperturbed version of the system that would be running in parallel. The size of the avalanche is directly related to the regime in which the RBN is; in the ordered regime, the cascades tend to be significantly smaller than in the chaotic regime. In real cells, where the regime is believed to lie on the edge of chaos, the cascades tend to be small too. Moreover, the distribution of the avalanche sizes in the ordered regime follows a power law curve [19], with many small and few large avalanches. In the chaotic regime, in addition to the power law distribution, 30–50 % of all avalanches are huge. The distribution of avalanches size of RBNs in the ordered regime roughly fits the expectations of biologists, where most of the genes, if perturbed, are only capable of initiating a very small avalanche, if any. Fewer genes could cause bigger cascades, and only a handful can unleash massive ones.

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

Naturally, one can imagine more sophisticated failure schemes on models of genetic regulatory networks such as RBNs. These failures are usually inspired by scientific experiments conducted on biological organisms. For example, the gene knockout experiment measures the expression level of all genes, in cells which a knockout gene and in normal cells, using cDNA microarray data. Serra and coauthors [27, 28] used this type of failure on RBNs to predict the size of real avalanches on microarray data. They showed that a very simple model with few inputs and random topologies can approximate the distribution of perturbation in gene expression levels with respect to microarray data. Moreover, they present a theoretical study showing that this simple model is actually valid in a particular type of network topologies.

Another notable perturbation inspired by real biological regulatory networks applied to RBNs is the gene duplication phenomenon suggested by Aldana and coauthors [29]. They study the robustness of genetic regulatory networks using RBNs and explore their behavior when exposed to nature-inspired genetic

perturbations: gene duplications. They show that an intrinsic property of such networks is to tend to preserve and multiply previous phenotypes, encoded in the attractor state-space of the network.

7 Discussion, Conclusions, and Future Work

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

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

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

Taking into account recent years' advances in the field of cellular biology, we have proposed to identify under what conditions Kauffman's hypothesis that living organism cells operate in a region bordering order and chaos holds. This property confers to organisms both the stability to resist transcriptional errors and external disruptions, and, at the same time, the flexibility necessary to evolution. We studied two particular cases of genetic regulatory networks found in literature in terms

of complex dynamical systems derived from the original RBN model. Therefore, we compared the behavior of these systems under the original update function and a novel additive function that we believe is closer to the actual role of living organisms.

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

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

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

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

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

Acknowledgments This work was partially supported by NIH grants LM009012, LM010098, AI59694, by the Swiss National Science Foundation grant PBLAP3-136923, and by Neuroscience Program of the Compagnia di San Paolo in Torino. The authors are grateful to Luca Ferreri for his precious help with statistical elaborations and the corresponding figures, and to Joshua L. Payne for his invaluable contribution to the discussions.

References

1. Pearson, H.: Genetics: what is a gene? Nature **441**(7092), 398–401 (2006)
2. Pennisi, E.: GENOMICS: DNA study forces rethink of what it means to be a gene. Science **316**(5831), 1556–1557 (2007)
3. Nowak, MA.: Evolutionary Dynamics: Exploring the Equations of Life. Harvard University Press, Cambridge, MA (2006)
4. Albert, R.: Boolean modeling of genetic regulatory networks. In: Ben-Naim, E., Frauenfelder, H., Toroczkai, Z. (eds.) Complex Networks. Lecture Notes in Physics, vol. 650, pp. 459–479. Springer, Berlin (2004)
5. Bower, J.M., Bolouri, H. (eds.): Computational Modeling of Genetic and Biochemical Networks. MIT, Cambridge, MA (2001)
6. Edwards, R., Glass, L.: A calculus for relating the dynamics and structure of complex biological networks. In: Berry, R.S., Jortner, J. (eds.) Advances in Chemical Physics, vol. 132, pp. 151–178. Wiley, New York (2006)
7. Vohradsky, J.: Neural model of the genetic network. J. Biol. Chem. **276**(39), 36168–36173 (2001)
8. Blake, W.J., Kaern, M., Cantor, C.R., Collins, J.J.: Noise in eukaryotic gene expression. Nature **422**(6932), 633–637 (2003)
9. Elowitz, M.B., Levine, A.J., Siggia, E.D., Swain, P.S.: Stochastic gene expression in a single cell. Science **297**(5584), 1183–1186 (2002)
10. Arkin, A., Ross, J., McAdams, H.H.: Stochastic kinetic analysis of developmental pathway bifurcation in phage {lambda}-infected escherichia coli cells. Genetics **149**(4), 1633–1648 (1998)
11. Gardner, T.S., Cantor, C.R., Collins, J.J.: Construction of a genetic toggle switch in escherichia coli. Nature **403**(6767), 339–342 (2000)
12. Elowitz, M.B., Leibler, S.: A synthetic oscillatory network of transcriptional regulators. Nature **403**(6767), 335–338 (2000)
13. Raser, J.M., O'Shea, E.K.: Noise in gene expression: origins, consequences, and control. Science **309**(5743), 2010–2013 (2005)
14. Leclerc, R.D.: Survival of the sparsest: robust gene networks are parsimonious. Mol. Syst. Biol. **4**, 214–218 (2008)
15. Kauffman, S.A.: Metabolic stability and epigenesis in randomly constructed genetic nets. J. Theor. Biol. **22**, 437–467 (1969)
16. Aldana, M.: Boolean dynamics of networks with scale-free topology. Phys. D. **185**, 45–66 (2003)
17. Roussel, M.R., Zhu, R.: Validation of an algorithm for delay stochastic simulation of transcription and translation in prokaryotic gene expression. Phys. Biol. **3**(4), 274–284 (2006)
18. Drossel, B.: Random boolean networks. Rev. Nonlinear Dynam. Complex. **1**, 69–110 (2008)
19. Kauffman, S.A.: Investigations. Oxford University Press, New York (2000)
20. Serra, R., Villani, M., Agostini, L.: On the dynamics of random boolean networks with scale-free outgoing connections. Phys. A Stat. Mech. Appl. **339**(3–4), 665–673 (2004)
21. Aldana, M., Cluzel, P.: A natural class of robust networks. Proc. Natl. Acad. Sci. USA **100**(15), 8710–8714 (2003)

22. Darabos, C., Di Cunto, F., Tomassini, M., Moore, J.H., Provero, P., Giacobini, M.: Additive functions in boolean models of gene regulatory network modules. PLoS One **6**(11), e25110 (2011)
23. Mesot, B., Teuscher, C.: Critical values in asynchronous random boolean networks. In: Banzhaf, W. (ed.) Advances in Artificial Life, ECAL2003. Lecture Notes in Artificial Intelligence, vol. 2801, pp. 367–376. Springer, Berlin (2003)
24. Gershenson, C.: Updating schemes in random Boolean networks: do they really matter? In: Pollack, J. (ed.) Artificial Life IX Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems, pp. 238–243. MIT, Cambridge, MA (2004)
25. Darabos, C., Tomassini, M., Giacobini, M.: Dynamics of unperturbed and noisy generalized boolean networks. J. Theor. Biol. **260**(4), 531–544 (2009)
26. Li, F., Long, T., Lu, Y., Ouyang, Q., Tang, C.: The yeast cell-cycle network is robustly designed. Proc. Natl. Acad. Sci. USA **101**(14), 4781–4786 (2004)
27. Serra, R., Villani, M., Semeria, A.: Genetic network models and statistical properties of gene expression data in knock-out experiments. J. Theor. Biol. **227**, 149–157 (2004)
28. Serra, R., Villani, M., Graudenzi, A., Kauffman, S.A.: Why a simple model of genetic regulatory networks describes the distribution of avalanches in gene expression data. J. Theor. Biol. **246**, 449–460 (2007)
29. Aldana, M., Balleza, E., Kauffman, S.A., Resendiz, O.: Robustness and evolvability in genetic regulatory networks. J. Theor. Biol. **245**, 433–448 (2007)
30. Kauffman, S.A.: The Origins of Order. Oxford University Press, Oxford (1993)

Attractors Perturbations in Biological Modelling: Avalanches and Cellular Differentiation

Marco Villani and Roberto Serra

Abstract We describe here and discuss in detail the model of random Boolean networks (RBNs). Although these models have been widely studied, they still present some unexpected mathematical features, and we discuss in particular their stability properties, introducing and commenting a new measure (attractor sensitivity) that seems particularly relevant for their application to the dynamics of gene regulatory networks. We also review some results that show that RBNs can properly account for data on perturbations induced by gene knock-out in real organisms. Moreover, we show that this comparison between model and data also sheds light on the important hypothesis that living beings tend to live in, or close to, critical states. Last but not least, we show that adding noise to RBNs can lead to a nice model of cell differentiation.

1 Introduction

Random Boolean networks (RBNs for short) were introduced four decades ago as models of gene regulatory networks [1, 2] and they became one of the major models of complex systems due to their interesting dynamical behavior [3–8]. The interest later faded, but in recent years it has been renewed by important theoretical advances [9–12] and also, as far as the application to genetics is concerned, by the availability of genome-wide expression data which can be properly described by RBNs [13–16] and by the possibility to describe complex phenomena, like cell differentiation [17–19], and whole organisms or tissues [20–23].

There have been a few attempts to use them as the basis of artificial learning systems, which met limited success and were soon abandoned [24]. However,

M. Villani (✉) • R. Serra
University of Modena e Reggio Emilia, v.Allegri 9, 42121 Reggio Emilia, Italy

European Centre for Living Technology, Ca' Minich, S. Marco 2940, 30124 Venezia, Italy
e-mail: marco.villani@unimore.it; mvillani@unimore.it; rserra@unimore.it

S. Cagnoni et al. (eds.), *Evolution, Complexity and Artificial Life*,
DOI 10.1007/978-3-642-37577-4_4, © Springer-Verlag Berlin Heidelberg 2014

recently the availability of sophisticated meta-heuristics has allowed to develop RBNs able to perform nontrivial tasks, like reaching certain dynamical states in a predefined number of steps and performing the majority classification task [25].

The reasons of this blooming of studies and applications reside on the dynamics of this kind of object that despite its apparently simple structure (a) can support several different asymptotic states and (b) exhibits different dynamical regimes. The combination of these characteristics allows the study of numerous—and not only biological—systems and processes.

Initially in this chapter we deepen the theme of dynamical regimes measurements of RBNs: this first part stresses once more the importance of the attractors as the most significant dynamical features of these systems. The following paragraphs then will present the consequence of the perturbations of these systems' asymptotic states: very interestingly, these studies will lead us to study interesting phenomena as gene knock-out events (Sect. 4.1), interactions with the environment (Sect. 4.2), and cellular differentiation (Sect. 4.3).

2 Random Boolean Networks

A classical RBN is a dynamical system composed of N genes, or nodes, which can take the value either 0 (inactive) or 1 (active). Let $x_i(t) \in \{0,1\}$ be the activation value of node i at time t, and let $X(t) = [x_1(t), x_2(t), \ldots x_N(t)]$ be the vector of activation values of all the genes. Real genes influence each other through their corresponding products and through the interaction of these products with other chemicals, by promoting or inhibiting the activation of target genes. In the corresponding model network these relationships are lumped in directed links (directed from node A to node B, if the product of gene A influences the activation of gene B) and Boolean functions (which model the response of each node to the values of its input nodes), chosen at random for every node, by assigning to each set of input values the outcome one with probability π (a parameter commonly known also as "bias"). In a classical RBN each node has the same number of incoming connections $\mathbf{k_{in}}$, and its $\mathbf{k_{in}}$ input nodes are chosen at random with uniform probability among the remaining $N-1$ nodes. Within the *quenched* strategy, both the topology and the Boolean function associated with each node do not change in time. The network dynamics is discrete and synchronous, so fixed points and cycles are the only possible asymptotic states in finite networks; typically a single RBN owns more than one attractors.

The model shows two main dynamical regimes: a common observation is that the average number of attractors and the average cycle length grow as a power law with the number of nodes N (in the ordered region) or could diverge exponentially (in the disordered region), mainly depending upon the value of the parameter $\mathbf{k_{in}}$ and the bias π; the dynamically disordered region shows also a significant sensitivity to different initial conditions, the opposite the ordered one. RBNs temporal evolution undergoes a second order phase transition between order and disorder, governed by the following relation between $\mathbf{k_{in}}$ and π: $\mathbf{k_{in\_c}} = [2\pi_c(1-\pi_c)]^{-1}$, where the

subscript c denotes the critical values [26]. Systems along this critical line show equilibrium between robustness and adaptiveness [27]; for this reason, they are supposed to be reasonable models of the living systems organization. Recent results support the hypothesis that biological genetic regulatory networks operate close to the critical region (the so-called edge of chaos) [13, 16, 28].

More careful analysis of some known real biological control circuits has shown that:

1. Boolean functions with a low probability of activation (i.e., a relatively high number of outputs which are 0) are more frequent than the others [29]
2. In most cases the functions are limited to those which are:
   (a) Canalizing, where at least one input has at least one state which suffices to determine the state of the regulated element [30]
   (b) Based on weighted sums of the values of other nodes [29]

It is possible therefore to individuate several groups of RBN ensembles, each one having different dynamical characteristics. In general, each group is characterized by a topology $\boldsymbol{T}$, a predefined set of allowed transition functions $\mathscr{F}$ that are randomly chosen with a given probability distribution p; in such a way we can therefore consider a family (a statistical ensemble) $\mathscr{M}$ of networks with the same topological features, including the same number of nodes N. In this chapter (generalizations are straightforward) we will assume that each node has the same number of incoming links $\mathbf{k_{in}}$, and the origin of these links is chosen at random with uniform probability among the remaining $N-1$ nodes, prohibiting multiple connections. As previously mentioned, by varying $\mathscr{F}$, p or $\boldsymbol{T}$ is possible to pass from disordered to ordered dynamical regimes, the critical ones corresponding to the onset of the percolation on the system of the fraction of non-oscillating nodes. Different families have therefore different dynamical regimes. Despite its generality, this description (the usual one) is driven by the RBN's structural properties and does not explicitly take into account the system dynamics, introducing in such a way mistakes and misinterpretations.

3 The Measure of RBN Dynamical Regimes

3.1 Static and Dynamic Estimates

The main static methods to measure the RBN dynamical regimes indeed implicitly presume ergodicity, that is, all inputs can arise with the same probability during evolution, and time average over the states visited by the network yields the same result as average over the whole phase space.

A very interesting measure is the "average sensitivity" of the RBN's Boolean functions, proposed by Shmulevich and Kauffman [10] (see more details in the

following), and used to explain the particular dynamical behavior caused by the presence of canalizing functions.

The alternative to the static measures is that of explicitly exploring the dynamical behavior of the system: in particular, an interesting and well-known method exploits the spreading of perturbations through the network. This measure involves two parallel runs of the same system, having the initial states different for only a small fraction of the units. This difference is usually measured by means of the so-called Hamming distance $h(t)$, defined as the number of units that have different activations on the two runs at the same time step. If after a transient the two runs are likely to converge to the same state (the measure is performed on many different initial condition realizations), i.e. $h(t) \to 0$, then the dynamics of the system is robust with respect to small perturbations (a signature of the ordered regime). If on the contrary the difference is likely to never converge, then the dynamics is sensitive to small perturbations and the corresponding regime is disordered. In other words, a system is ordered when the Hamming distance is continuously decreasing from one step to the next one, or mathematically:

$$\lambda = \lim_{h(t)\to 0} \frac{dh\,(t+1)}{dh(t)} < 1 \tag{1}$$

whereas is critical or disordered if the limit (the λ parameter being called "Derrida parameter") is equal or greater than 1 [5].

Following this idea, a common practice to measure the dynamical regime of an RBN is that of randomly generate a great number of pairs of initial conditions differing each other for one or more units, perform one step, measure the Hamming distance of the two resulting states, take the averages for each perturbation size, and compute the limit of the slope of the tangent of the curve as the perturbation size tends to zero. This is the so-called Derrida procedure [26].

However, the ordinary functioning of most interesting systems happens on their asymptotic states (the attractors): measures taken on randomly chosen states therefore do not necessarily allow a correct estimate of the effective system dynamical behavior. In other words, the ergodicity assumption does not hold for the dynamics of arbitrary RBNs [31, 32].

3.2 The Attractor Sensitivity

A simple way to avoid this difficulty is that of applying the classical Derrida procedure (the result being indicated in the following by the acronym DA) only on the states belonging to the attractors [33]. We can therefore define the sensitivity on attractor i (SA_i) as the result of the Derrida procedure performed only on the states belonging to the attractor i, and the sensitivity on attractors (SA) as the average

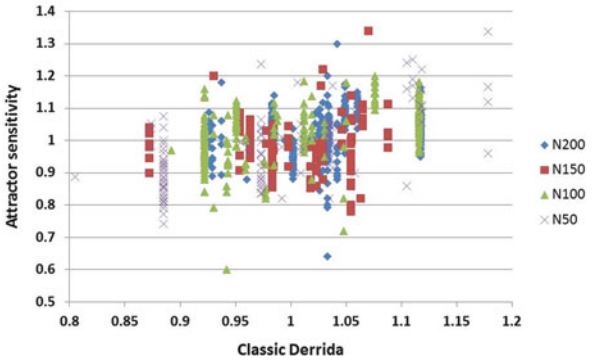

Fig. 1 The attractor sensitivities of 40 networks, grouped in four sets having different number of nodes. The sensitivities of each net (visible as vertical columns of marks) can have very different values, that in turn could be very different from the corresponding DA values

of the SA_i, each SA_i being weighted on the dimension of its attraction basin.[1] In order to appreciate the differences among DA, SA, and SA_i, we can perform these measures on different statistical ensembles (families).

First, we can use the usual ensemble, by taking into account families composed by networks with random topology, $\mathbf{k_{in}} = 2$ and all the Boolean functions allowed: we analyze four sets having different N (four families), ten networks each, plotting all the SA_i belonging to the attractors of each net vs. their classical DA measure. As you can observe in Fig. 1, the attractors of each net (visible as vertical columns of marks) can have very different SA_i, that in turn could be very different from the corresponding DA values: these dissimilarities are evident in several networks of each family, despite their different number of nodes.[2] However, one can see from Fig. 2 that the SA (i.e., the weighted average of the attractors' sensitivities) correlates with the DA so that, neglecting transients, the system behaves as an ergodic one (time averages being close to ensemble averages). Indeed, there is a clear—although a little bit noisy—proportionality between SA and DA (Fig. 2), and even more remarkably in each set the averages of DA and of SA are practically equal[3] (Table 1b).

Let us now consider two families of RBNs, indicated in the following with $\mathscr{M}_1$ and $\mathscr{M}_2$, with $\mathbf{k_{in}} = 2$, in which only a subset of canalizing Boolean functions is allowed. The interest for these particular families is inspired by a study on random threshold networks (RTNs), networks in which each node receives the weighted sum

[1] An attractor basin is the set of states whose evolution lead to the attractor, its size (or dimension) being the cardinality of the set.

[2] For example, one of the nets with $N = 200$ has DA = 1.033 and 24 attractors with SA_i that span from 0.64 to 1.20, with a final SA equal to 0.967.

[3] In this case all the averages are equal to 1, the ensemble being the first historical example of critical systems (DA = 1).

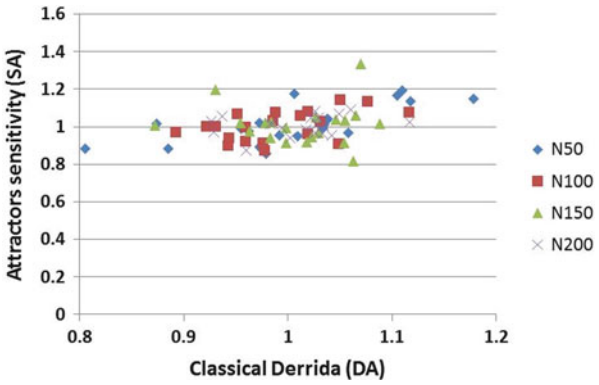

Fig. 2 The SA values of the same nets of Fig. 1. SA correlates with DA so that there is a clear—although noisy—proportionality between SA and DA

of its input (the weights being ± 1 in the present case); if the sum exceeds a given threshold, the node acquires the values of "1," and "0" otherwise.

It is possible to map these dynamical rules on RBN framework, obtaining a set $\mathscr{F}$ of rules for each threshold. In this work we wish to highlight the dynamical behaviors of the RTNs having homogeneous thresholds $h = +0.5$ and $h = -0.5$ (identical for each node in the network), or likewise of the corresponding RBN families $\mathscr{M}_1$ and $\mathscr{M}_2$: incidentally, the two sets of Boolean functions identified in this way are complementary to each other—see Table 1 for the details.

It is interesting to notice that the two families of networks so defined show different dynamical behaviors. This may appear somewhat surprising, given that the Boolean functions are complementary in the two cases: while one would naively expect that two networks with complementary Boolean functions behave in the same way, this is not the case. The simulations indeed show that the dynamics is different, and that complementary functions don't necessarily lead to complementary evolution. In this case the differences are considerable: the average number of the attractors of RTNs with $\mathbf{k_{in}} = 2$ and $h = 0.5$ ($\mathscr{M}_1$) is significantly higher than the average number of networks with same connectivity and $h = -0.5$ ($\mathscr{M}_2$); one also finds considerable differences in length of cycles and number of frozen nodes [33]. The two families of RBN show therefore very different behaviors despite the fact that both have the same DA, which for networks of 70 nodes is found to be 0.74. Hence, the Derrida parameter isn't able to correctly describe the dynamics in these particular cases, where there is a remarkable difference between DA and SA. The attractor sensitivity indeed (in network with $N = 70$) turns out to be 0.90 for the $\mathscr{M}_1$ family and 0.65 for the $\mathscr{M}_2$ family. This difference detects and makes sense of the different behaviors of the two situations and is a general characteristic of these families, as we will see in next paragraphs.

An even extreme case, where DA and SA radically diverge, concerns a situation where the system performs a global computation, based on units that can process

Table 1 (a) The averages of DA and of SA of the four sets of Figs. 1 and 2, and (b) the Boolean functions allowed in the $\mathscr{M}_1$ and $\mathscr{M}_2$ families

(a)

N	DA	SA
50	1.00 ± 0.09	1.0 ± 0.1
100	0.99 ± 0.06	1.00 ± 0.05
150	1.00 ± 0.05	1.01 ± 0.05
200	1.00 ± 0.05	1.01 ± 0.05

(b)

		$\mathscr{M}_1$				$\mathscr{M}_2$			
A	B	OR	!A&B	A&!B	FALSE	NOR	!AorB	Aor!B	TRUE
1	1	1	0	0	0	0	1	1	1
1	0	1	0	1	0	0	0	1	1
0	1	1	1	0	0	0	1	0	1
0	0	0	0	0	0	1	1	1	1
Derrida parameter		0.74				0.74			

information only at local level. The RBN in this case has to solve the so-called density classification problem, which requires that a discrete dynamical system recognizes whether an initial binary string contains more 0s or more 1s. The problem is that of designing simple rules, governing the dynamics of each node, in such a way the system is driven to a uniform state, consisting of all 1s if the initial configuration contains more 1s and all 0s otherwise [34]. It can be shown that the simple majority rule applied on random topologies outperforms all human or artificially evolved rules running on ordered lattice [35, 36], a performance difference that increases by increasing the number of nodes [35]. The majority rule states that the value of a node at time $t + 1$ is 0 (resp. 1) if the majority of its neighbors has value 0 (resp. 1) at time t.

In this context we studied a family of RBN, evolved to solve this task (see [25] for the details) with $\mathbf{k_{in}} = 3$ and $N = 71$ (an odd number of nodes avoiding the cases with equal quantity of 0s and 1s) [34]. According to the Derrida parameter this family is chaotic (with a DA = 1.50), but the attractor sensitivity leads to a very different conclusion: indeed, this measure results lower than 0.001, indicating that the system is deeply in the ordered region. And indeed the system is very ordered, having few very short attractors (typically, only two fixed points) with regular basins of attraction (nearby initial conditions evolve toward the same attractor).

Figure 3 shows the distributions of DP and AS in 200 realizations, revealing the impressive difference existing in this case between these parameters.

Interestingly, the probability of unit i to spread an incoming perturbation in its neighbors can be related to the influence of their variables on its function F_i; by means of this relation, it is possible to relate static and dynamical measures. Indeed by definition the influence of a jth input variable of a function F_i, $I_j(F_i)$, is the probability that the function F_i changes its value when the value of the jth variable

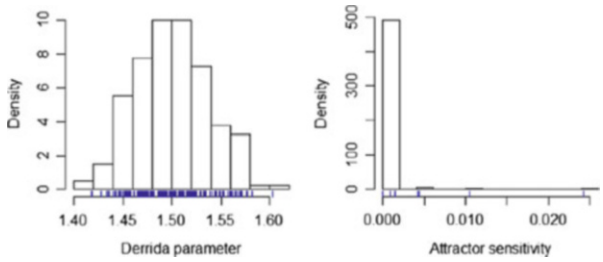

Fig. 3 Histogram of distribution of classical Derrida values and of attractor sensitivity. Two hundred networks, $N = 71$

is changed (a concept linked to Boolean derivatives and Lyapunov exponents of RBNs [37, 38] and to the activity [10]). Strictly connected with the influences of the variables of function F_i is the sensitivity $s^{\text{fi}}(\mathbf{x})$ of the Boolean function itself, which measures how sensitive the output of the function F_i is to changes in their inputs: the sensitivity $s^{\text{fi}}(\mathbf{x})$ indeed is defined as the number of Hamming neighbors of $\mathbf{x}$ on which the F_i function values are different than on $\mathbf{x}$ (two vectors are Hamming neighbors if they differ in only one component) [10]. Finally, the average sensitivity of the function F_i, $I(F_i)$ is the expectation value of $s^{\text{fi}}(\mathbf{x})$ with respect to the distribution of $\mathbf{x}$. Under the hypothesis of uniform input distribution, the average sensitivity of function F_i, $I(F_i)$, is equal to the sum of the influences of all its input variables (a number that spans from 0 to $\mathbf{k_{in}}$), whereas the average sensitivity of an ensemble of Boolean functions is the average of the sensitivities of all its functions. This last average coincides with the Derrida parameter [31].

The dynamics enter into the computation when we calculate the sum of the influences of the input variables of a function F_i: it is an easy computation under the ergodicity assumptions (and in this case the result is the classical DA), but it requires more attention if this hypothesis doesn't apply.

We can, for example, estimate the SA for $\mathscr{M}_1$ and $\mathscr{M}_2$ families from time series (without the need of expensive additional perturbations). We have to compute the occurrence probability for each input configuration (the dynamical part, where are involved the time series), which on average depends on the fraction of "1" within the system (its occurrence probability b) (Table 2), and weight in such a way the influence of the input variables of each function; finally, it is enough to sum the influences of the input variables and make the global average of the system (weighted by the presence of the functions within the system) (Table 3).

Sometimes it is possible to analytically estimate the value of b and therefore the value of λ by using the so-called annealed approximation, a sort of mean field approximation that holds for annealed networks [26] having an infinite number of nodes but that nevertheless can give reasonable guesses [32]; please note, however, that this sort of approach in any case cannot take into account the effects of the finite number of nodes that sometimes are present, as it is possible to note for the family $\mathscr{M}_1$ on last rows of Table 4.

Table 2 The table shows the probability of occurrence of each input configuration vs. the "1" occurrence probability b, and reports a "1" if a change on the first input has as consequence a change in the corresponding function output

Input			Family $\mathscr{M}_1$				Family $\mathscr{M}_2$			
A[a]	B	P(A, B)	OR	A&!B	!A&B	F	NOR	!AorB	Aor!B	T
0	0	$(1-b)^2$	1	1	0	0	1	1	0	0
0	1	$b(1-b)$	0	0	1	0	0	0	1	0
1	0	$b(1-b)$	1	1	0	0	1	1	0	0
1	1	b^2	0	0	1	0	0	0	1	0
A	B[a]		OR	A&!B	!A&B	F	NOR	!AorB	Aor!B	T
0	0	$(1-b)^2$	1	0	1	0	1	1	1	0
0	1	$b(1-b)$	1	0	1	0	1	0	1	0
1	0	$b(1-b)$	0	1	0	0	0	1	0	0
1	1	b^2	0	1	0	0	0	1	0	0

[a]Flipped input

Table 3 The table shows the influence of the input variables, computed by taking into account the effective occurrence probability of the possible input configurations, and the resultant function sensitivity; eventually, the critical parameter λ is estimated (in each family the four Boolean functions have the same occurrence probability by construction)

	Family $\mathscr{M}_1$ ($b = 0.08$)				Family $\mathscr{M}_2$ ($b = 0.67$)			
	OR	A&!B	!A&B	F	NOR	!AorB	Aor!B	T
Influence of A	0.92	0.08	0.92	0	0.33	0.22	0.67	0
Influence of B	0.92	0.92	0.08	0	0.33	0.78	0.33	0
Function sensitivity	1.84	1.00	1.00	0	0.66	1.00	1.00	0
λ parameter	0.96				0.67			

Table 4 For each family the table shows the theoretical estimate of the critical parameter λ (computed by means of the procedure explained in the text) and the corresponding experimental value (estimated by effectively performing the Derrida procedure on random initial conditions and only on the attractor states, respectively, for DA and SA measures)

			Family $\mathscr{M}_1$			Family $\mathscr{M}_2$	
Measure	N	b	Theoretical	Experimental	b	Theoretical	Experimental
DA	70	0.50	0.75	0.74	0.50	0.75	0.74
SA	70	0.08	0.96	0.90	0.67	0.66	0.64
SA	700	0.05	0.97	0.95	0.66	0.67	0.67
SA	∞	0.00	1.00	–	0.67	0.67	–

4 Perturbations on Attractors

RBNs can behave differently in differently portions of the state space: therefore, the more correct approach to measure the dynamical regime of the system involves the more common system's conduct, that is, the system's asymptotic states (its attractors). As we have seen on the previous paragraphs each attractor can behave

differently: so, if we desire to analyze the system responses to perturbations, we need to act on the attractors states. The next paragraphs will discuss the effects of three kinds of perturbations made on attractors: permanent (in which one or more nodes have the activity fixed to "1" or to "0," irrespective of its Boolean function), semipermanent (in which one or more nodes have the activity fixed for a long time, and then its habitual behavior is resumed), and temporary (in which one or more nodes have the activity fixed to "1" or to "0" for only one time step). Interestingly, each kind of perturbation corresponds to a real biological situation.

4.1 Gene Knock-Out Experiments

A permanent perturbation event corresponds to a gene (or to a set of genes) which continuously produces—or stops the production of—its protein: this situation could have several consequences on the activities of the genes that have this protein among their drivers, and that can in turn propagate this unusual behavior. In molecular biology literature there are several examples of these events, and during the last years these processes are increasingly better documented by means of the use of the cDNA microarray technology.

A useful and public set of cDNA microarray measurements of gene expression profiles is analyzed in [39]: it regards 227 single knock-out experiments on *Saccharomyces cerevisiae*, where one compares the expression levels of all the genes, in cells with a knocked-out gene, with those in normal ("wild-type") cells. In this way, all the experimental data can be cast in matrix form E_{ij}, $i = 1 \ \ldots$ 6,325 (the number of genes of *Saccharomyces*), $j = 1 \ \ldots \ 227$ (the number of experiments); E_{ij} is the ratio of the expression of gene i in experiment j to the expression of gene i in the wild-type cell. Microarray data are noisy; therefore, to make precise statements about the number of genes perturbed in a given experiment, we can define a threshold, such that the difference is regarded as "meaningful" if the ratio is greater than the threshold τ or smaller than $1/\tau$ and neglected otherwise. There is arbitrariness in defining this threshold, associated with the transformation of a continuous set of values into a Boolean one, but several experiences indicate that a value close to seven is satisfactory [13].

Let Y be the Boolean matrix which can be obtained by E by posing $y_{ij} = 1$ if $E_{ij} > \tau$, or $E_{ij} < 1/\tau$; $y_{ij} = 0$ otherwise ($y_{ij} = 1$ therefore means that the modification of the expression level of gene i in experiment j is accepted as "meaningful"). In order to describe the global features of these experiments, we introduced the notion of avalanche, the size of the perturbation induced by a particular experiment (in experiment j, $V_j = \Sigma_i \ y_{ij}$), and of gene susceptibility, the number of times a particular gene is involved on an avalanche (for gene i, $\mathrm{Su}_i = \Sigma_j \ y_{ij}$) [14]. In such a way we obtain an avalanche distribution where 45 % of the avalanches have size 1 and the biggest one involves 190 genes, whereas very few genes are involved more than four times [14].

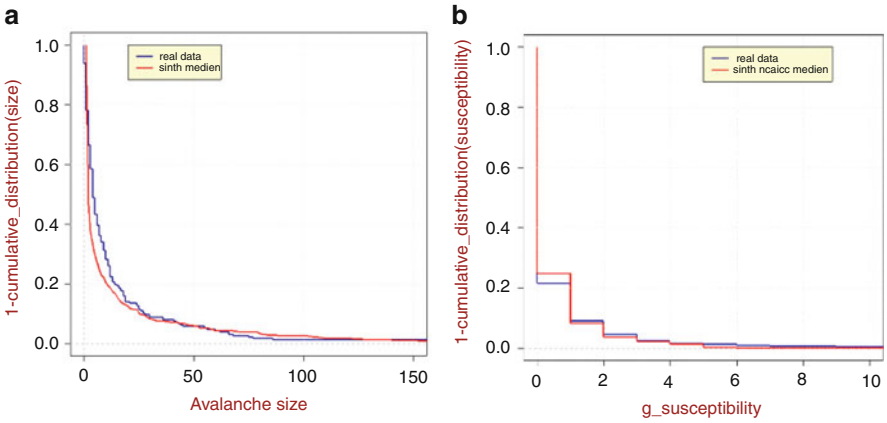

Fig. 4 Comparison between (**a**) avalanche size of *S. cerevisiae* (with threshold = 7) and median avalanche size of ten synthetic networks and (**b**) between gene susceptibility of *S. cerevisiae* (with threshold = 7) and median gene susceptibility of ten synthetic networks. In order to suppress noise, both plots show one-cumulative distribution of the variables

We can reproduce in silico the knock-out process: at a certain time point (for the reason before shown, we use a state of one attractor), the state of one of the nodes of an RBN with 6,325 nodes is permanently clamped to the value 0. The evolution of the unperturbed network is compared to that of the perturbed one; a gene is said to be affected (or perturbed) if its value differs in the two networks in at least one time step. The avalanche corresponding to a given knock-out is the set of perturbed genes (including the one which has been knocked-out): in such a way the size of an avalanche in the model cannot be smaller than 1, nor larger than the number N of nodes. By repeating this schema for several times we proved through simulations [14] (and later also theoretically, finding that for Erdos–Renyi topologies the probability of having an avalanche of size equal to n is $p_n = B_n \lambda^{n-1} e^{-n\lambda}$ [13]) that RBNs with λ slightly smaller than 1 can correctly reproduce the experimental data (Fig. 4).

4.2 Information Exchange with the Environment

Cells continuously exchange matter, energy, and information with the environment (the other cells being a particular case of "environment," taking into account, for example, colonies or multicellular beings). In particular, the most frequent way to communicate with the environment is that of allowing chemicals to cross the cellular membrane (a) directly, passively or thanks to the active actions of specialized proteins or (b) indirectly, the encounter of the signaling molecule with the membrane enacting an internal chemical reaction cascade whose final target are

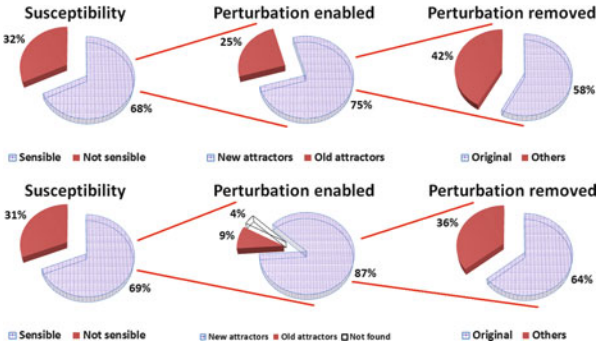

Fig. 5 Consequences of semipermanent perturbation on RBN's attractors. *First row*: nets with ten nodes; *second row*: nets with 100 nodes. See text for explanations

the genes of the cell. In such a way, for example, cells are able to detect the presence of particular substances and activate the corresponding metabolic pathways able to metabolize them. When these substances disappear, cells are able to come back to the previous pathways [40]. A simple way to simulate these processes by means of RBN is that of fixing to "1" or to "0" one or more genes (the targets of the signals coming from outside), till the chemicals are present on the environment, and allowing the usual genes behavior when the chemicals disappear. In other words, we can use semipermanent perturbations: let us therefore consider the effects of semipermanent perturbations on the attractors of the RBN system.

Note that this kind of perturbation actually changes the original RBN (indeed it can be observed that the perturbed node is now ruled by a different Boolean function, i.e. true or false): so the attractors of the perturbed network can be different from those of the original one (apart from the obvious difference concerning the state of the perturbed node itself).

In order to observe these aspects we analyzed two groups of networks having $N = 10$ and $N = 100$ nodes, each composed by 100 networks. To find the RBN's attractors we exhaustively checked all the possible initial conditions for the nets with $N = 10$ and performed a random sampling for the nets with $N = 100$. For the nets with $N = 10$ we perturbed all the nodes by starting in all the phases, whereas for the nets with $N = 100$ we perturbed the 20 % of the total possibilities; the main results are shown in Fig. 5.

The graphs in the first row are referring to nets with $N = 10$ nodes, whereas graphs in the second row are referring to nets with $N = 100$ nodes. The first column shows the susceptibility, defined as the fraction of experiments where the RBN, initially on the attractor A, when a permanent perturbation is applied, goes to an attractor A' not equivalent to A (we define equivalent two attractors that are equal in all the nodes, with the exception of the perturbed one). The second column shows that, from all the cases where A' is not equivalent to A, the largest part of A' attractors are not equivalent to any attractor of the original RBN (they are totally new attractors). The third column refers only to the "new attractors" A', and

describe what happens when the perturbation is removed and the system is allowed to relax toward the attractors of the original not perturbed net. The graph shows how many times the final attractors B coincide with the original attractors A, and how many times B differs from A ($B \neq A$). Note that in a limited number of cases (with $N = 100$) it was not possible to individuate the attractors because of computational vincula.

The main considerations we can derive from these simulations are:

1. The susceptibility (as before defined) seems to be almost independent of the net size
2. The bias toward already known A' attractors decreases with the net size
3. The perturbed nets can exhibit attractors different from those of the original nets
4. The permanent perturbations have significant consequences also after the perturbation has been removed, when in more than 20 % of the cases the final attractors B are different from the original ones A

4.3 Cell Differentiation

It is possible to perturb attractors by means of a temporary disturbance, in which for only one time step one gene doesn't correctly react to its input (or the same for more than one genes): if this perturbation acts only once we obtain one step of the Derrida procedure, but if the perturbation repeatedly appears, we obtain a noisy RBN [47], a system that corresponds to the real situations better than simply deterministic RBNs (cells in effect are very noisy systems [41–46]).

The introduction of noise has as its main consequence the fact that the asymptotic behavior of a noisy RBN is not the single attractor, but is the set of attractors where the system subjected to noise can pass through the so-called ergodic set [47]. Unfortunately, noisy RBNs have typically only a single ergodic set, losing in such a way one of their most interesting characteristic, that is, that of supporting several different asymptotic behaviors with only one regulatory structure. In Kauffman [1–3] original interpretation, this fact means that RBN cannot represent a model for cells belonging to multicellular organisms. Our group proposed in a series of articles [17–19] a way to avoid this difficulty: unexpectedly, this solution allowed us to describe cellular differentiation, i.e. the process whereby stem cells, which can develop into different types, become more and more specialized.

The proposal is based on the observation that the transition probabilities among attractors are not all equal (Fig. 6): if we neglect the transitions having probability too low to occur during a cell lifetime, we recover the propriety of having different asymptotic behaviors. The so identified sets are called threshold ergodic sets (briefly TES_θ, θ being the threshold above which the transitions are neglected). These sets are robust under noise and "close," that is, the system once entered on a TES_θ cannot leave its state space corresponding area.

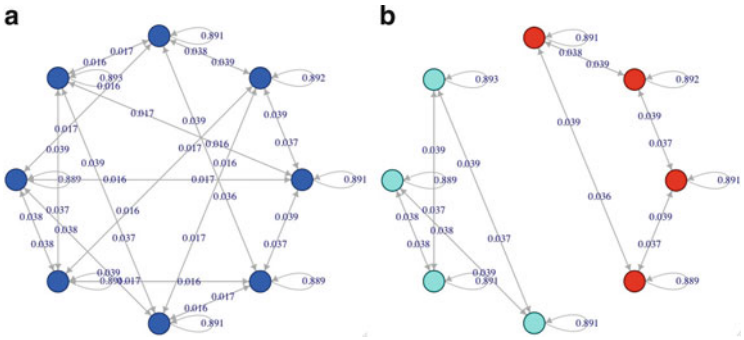

Fig. 6 Attractor transition graph in an RBN. *Circles* represent attractors; *arrows* represent transitions among attractors induced by single spin flips. The numbers on each arrow are the probability that, by flipping at random the state of a node in an attractor, that transition takes place. (**a**) The complete attractor transition graph; (**b**) the same graph, where links below the threshold θ=0.02 are removed

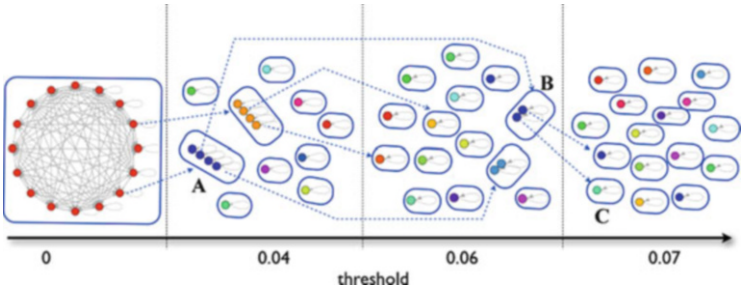

Fig. 7 TESs and differentiation. As the threshold is increased the single TES_0 breaks into smaller disjoint TESs, corresponding to more differentiated cells, until eventually final cell types are reached. Examples of stochastic transitions are shown by *dotted lines*. By acting on particular genes at each noise reduction event, it is possible to select the particular pathway that links the TES_θ A, B and C. Please note that the (semi) permanent perturbations that allow this selection slightly change the attractors' patterns

Moreover, if we assume that cells can modify their internal level of noise (a real situation, see for example, [45]), we can deduce that cells could regulate its differentiation state by means of noise regulation.

By following this idea, stem cells are noisy systems, wandering through a very ample portion of the state space (the TES_0, the set of attractors forming a strong connected component that in RBN usually collects a significant fraction of the possible attractors [18, 19]): by lowering the internal noise level (in the model, by increasing θ), each cell of a population remains blocked in the TES_θ that contains the particular attractor where the cell is at the moment of noise reduction. In such a way we obtain the so-called stochastic differentiation, in which for each lineage the proportions among the resulting different cellular types are constant (Fig. 7) [18, 19].

Interestingly, we found that in approximately one third of critical RBNs a (semi)permanent fixing of particular genes during the noise reduction event can drive these transitions and force cells toward particular final destinations (particular TES_θ): in other words, the (repeated) combinations of noise reduction and cells communication processes can individuate a particular differentiation pathway, giving birth to the so-called deterministic differentiation (see Fig. 7 for an example) [18, 19]. We called the nodes able to drive the transitions "switch nodes."

Differentiation is almost always irreversible, but there are limited exceptions under the action of appropriate signals [48, 49]: also in the model (but only in a small fraction of the cases, as in real systems) we can obtain similar effects, by removing the initial semipermanent perturbation or by fixing the activity of a different switch gene [18, 19]. In very special cases after an action on a switch gene it was possible to come back to a pluripotent state (a TES_{θ_2} with more attractors with respect to the starting TES_{θ_1}, despite $\theta_1 < \theta_2$, and similar to the TES_θ from which TES_{θ_1} derived [18, 19], simulating in such a way a famous experiment of Yamanaka on induced pluripotency [50, 51]) or to directly jump from one differentiated cell type into another (simulating in such a way experiments as [52]).

With the same theoretical framework it is therefore possible to describe the most relevant features of the cell differentiation: (1) different degrees of differentiation; (2) stochastic differentiation; (3) deterministic differentiation in well-defined lineages; (4) limited reversibility; (5) induced pluripotency; and (6) induced change of cell type. Please note that in doing this we used only critical RBNs: ordered RBNs have too stable attractors (and therefore negligible transitions among attractors) whereas in disordered RBNs noise induces continuous transitions inhibiting in such a way any coherent behavior.

5 Conclusions

In this chapter we introduced the theme of RBN dynamical regimes, and we stressed the importance of the attractors as the most significant dynamical features of these systems. By means of theory and examples we noted that RBNs can support attractors having different dynamical properties: this fact has important consequences when we want to couple simulations and biological experiments.

An important theme strongly linked with the previous considerations is that of the dynamics of perturbations on the system: by inducing perturbations on attractors we were able to study important phenomena as gene knock-out, interactions with environment and cell differentiation.

Thanks to this theoretical approach and to the comparison between simulations and experimental results we have been able to give important clues about the dynamical regimes of living biological cells, and to interpret their information exchange with the environment. Moreover, the hypothesis about cell differentiation implies that stem cells should have high level of internal noise, a forecast that could be verified by using modern technologies ,for example, cDNA microarray.

The RBN framework therefore is revealing once again a fortunate and formidable source of scientific interpretation and creativity.

Acknowledgments We thank S.A. Kauffman, Anna Maria Colacci, Alessia Barbieri, Alex Graudenzi and Davide Campioli for fruitful discussions. We thank also the MIUR-FISR project nr. 2982/Ric (Mitica project) for support.

References

1. Kauffman, S.A.: Metabolic stability and epigenesis in randomly constructed genetic nets. J. Theor. Biol. **22**(3), 437–467 (1969)
2. Kauffman, S.A.: Gene regulation networks: a theory of their global structure and behaviour. Curr. Top. Dev. Biol. **6**, 145–182 (1971)
3. Kauffman, S.A.: At Home in the Universe. Oxford University Press, Oxford (1995)
4. Kauffman, S.A.: The Origins of Order: Self-Organization and Selection in Evolution, 1st edn. Oxford University Press, Oxford (1993)
5. Aldana, M., Coppersmith, S., Kadanoff, L.P.: Boolean dynamics with random couplings. In: Kaplan, E., Marsden, J.E., Sreenivasan, K.R. (eds.) Perspectives and Problems in Nonlinear Science, pp. 23–89. Springer, New York (2003)
6. Bastolla, U., Parisi, G.: The modular structure of Kauffman networks. Physica D **115**, 219–233 (1998)
7. Bastolla, U., Parisi, G.: Relevant elements, magnetization and dynamical properties in Kauffman networks: a numerical study. Physica D **115**(3–4), 203–218 (1998)
8. Aldana, M.: Dynamics of Boolean networks with scale free topology. Physica D **185**, 45–66 (2003)
9. Socolar, J.E., Kauffman, S.A.: Scaling in ordered and critical random Boolean networks. Phys. Rev. Lett. **90**, 068702 (2003)
10. Shmulevich, I., Kauffman, S.A.: Activities and sensitivities in Boolean network models. Phys. Rev. Lett. **93**, 048701 (2004)
11. Drossel, B.: Number of attractors in random Boolean networks. Phys. Rev. E **72**, 016110 (2005)
12. Drossel, B.: Random Boolean networks. In: Schuster, H.G. (ed.) Reviews of Nonlinear Dynamics and Complexity, vol. 1. Wiley, Weinheim (2008)
13. Serra, R., Villani, M., Graudenzi, A., Kauffman, S.A.: Why a simple model of genetic regulatory networks describes the distribution of avalanches in gene expression data. J. Theor. Biol. **246**, 449–460 (2007)
14. Serra, R., Villani, M., Semeria, A.: Genetic network models and statistical properties of gene expression data in knock-out experiments. J. Theor. Biol. **227**, 149–157 (2004)
15. Serra, R., Graudenzi, A., Villani, M.: Genetic regulatory networks and neural networks. In: Apolloni, B., Bassis, S., Marinaro, M. (eds.) New Directions in Neural Networks, pp. 109–117. IOS Press, Amsterdam (2009)
16. Shmulevich, I., Kauffman, S.A., Aldana, M.: Eukaryotic cells are dynamically ordered or critical but not chaotic. Proc. Natl. Acad. Sci. U. S. A. **102**, 13439–13444 (2005)
17. Villani, M., Serra, R., Barbieri, A., Roli, A., Kauffman, S.A., Colacci, A.: Noisy random Boolean networks and cell differentiation. In: Proceedings of ECCS2010 – European Conference on Complex Systems (2010)
18. Serra, R., Villani, M., Barbieri, B., Kauffman, S.A., Colacci, A.: On the dynamics of random Boolean networks subject to noise: attractors, ergodic sets and cell types. J. Theor. Biol. **265**, 185–193 (2010)
19. Villani, M., Barbieri, A., Serra, R.: A dynamical model of genetic networks for cell differentiation. PLoS One **6**(3), e17703 (2011)

20. Villani, M., Serra, R., Ingrami, P., Kauffman, S.: Coupled random Boolean network forming an artificial tissue. In: Cellular Automata. Lecture Notes in Computer Science, vol. 4173. Springer, Berlin (2006)
21. Serra, R., Villani, M., Damiani, C., Graudenzi, A., Colacci, A., Kauffman, S.A.: Interacting random Boolean networks. In: Jost, J., Helbing, D. (eds.) Proceedings of ECCS07: European Conference on Complex Systems (2007)
22. Serra, R., Villani, M., Damiani, C., Graudenzi A., Colacci, A.: The diffusion of perturbations in a model of coupled random Boolean networks. In: Umeo H. (ed.) ACRI 2008, Lecture Notes in Computer Science, pp. 315–322. Springer, Berlin (2008)
23. Damiani, C., Serra, R., Villani, M., Kauffman, S.A., Colacci, A.: Cell–cell interaction and diversity of emergent behaviours. IET Syst. Biol. **5**(2), 137–144 (2011)
24. Patarnello, A., Carnevali, P.: Learning capabilities of Boolean networks. In: Aleksander, I., et al. (eds.) Neural Computing Architectures, pp. 117–129. MIT Press, Cambridge (1989)
25. Benedettini, S., Villani, M., Roli, A., Serra, R., Manfroni, M., Gagliardi, A., Pinciroli, C., Birattari, M.: Dynamical regimes and learning properties of evolved Boolean networks. Neurocomputing **99**, 1 (2013) (Elsevier)
26. Derrida, B., Pomeau, Y.: Random networks of automata: a simple annealed approximation. Europhys. Lett. **1**, 45–49 (1986)
27. Aldana, M., Balleza, E., Kauffman, S., Resendiz, O.: Robustness and evolvability in genetic regulatory networks. J. Theor. Biol. **245**, 433–448 (2007)
28. Balleza, E., Alvarez-Buylla, E., Chaos, A., Kauffman, S., Shmulevich, I., Aldana, M.: Critical dynamics in genetic regulatory networks: examples from four kingdoms. PLoS One **3**, e2456 (2008)
29. Raeymaekers, L.: Dynamics of Boolean networks controlled by biologically meaningful functions. J. Theor. Biol. **218**, 331–341 (2002)
30. Harris, S.E., Sawhill, B.K., Wuensche, A., Kauffman, S.: Complexity **7**(4), 23–40 (2002)
31. Moreira, A.A., Amaral, L.A.N.: Canalizing Kauffman networks: nonergodicity and its effect on their critical behavior. Phys. Rev. Lett. **94**, 218702 (2005)
32. Szejka, A., Mihaljev, T., Drossel, B.: The phase diagram of random threshold networks. New J. Phys. **10**, 063009 (2008)
33. Campioli, D., Villani, M., Poli, I., Serra, R.: Dynamical stability in random Boolean networks. In: Apolloni, B., Bassis, S., Esposito, A., Morabito, C.F. (eds.) Neural Nets WIRN11, p. 120–128. IOS Press, Amsterdam (2011)
34. Packard, N.: Adaptation toward the edge of chaos. In: Kelso, J., Mandell, A., Shlesinger, M. (eds.) Dynamic Patterns in Complex Systems, pp. 293–301. World Scientific, Singapore (1988)
35. Mesot, B., Teuscher, C.: Deducing local rules for solving global tasks with random Boolean networks. Physica D **211**, 88–106 (2005)
36. Serra, R., Villani, M.: Perturbing the regular topology of cellular automata: implications for the dynamics. In Chopard B., Tomassini M., Bandini S. (eds.) Cellular Automata. Lecture Notes in Computer Science, vol. 2493, pp. 168–177. Springer, Heidelberg (2002)
37. Luque, B., Solé, R.V.: Lyapunov exponents in random Boolean networks. Physica A Stat. Mech. Appl. **284**(1), 33–45 (2000)
38. Bagnoli, F., Rechtman, R., Ruffo, S.: Damage spreading and Lyapunov exponents in cellular automata. Phys. Lett. A **172**(1–2), 34–38 (1992)
39. Hughes, T.R., et al.: Functional discovery via a compendium of expression profiles. Cell **102**, 109–126 (2000)
40. Samal, A., Jain, S.: The regulatory network of *E. coli* metabolism as a Boolean dynamical system exhibits both homeostasis and flexibility of response. BMC Syst. Biol. **2**, 21 (2008)
41. McAdams, H.H., Arkin, A.: Stochastic mechanisms in gene expression. Proc. Natl. Acad. Sci. U. S. A. **94**, 814–819 (1997)
42. Swain, P.S., Elowitz, M.B., Siggia, E.D.: Intrinsic and extrinsic contributions to stochasticity in gene expression. Proc. Natl. Acad. Sci. U. S. A. **99**, 12795–12800 (2002)
43. Blake, W.J., Kaem, M., Cantor, C.R., Collins, J.J.: Noise in eukaryotic gene expression. Nature **422**, 633–637 (2003)

44. Raj, A., van Oudenaarden, A.: Nature, nurture, or chance: stochastic gene expression and its consequences. Cell **135**(2), 216–226 (2008)
45. Lestas, I., Paulsson, J., Ross, N.E., Vinnicombe, G.: Noise in gene regulatory networks. IEEE Trans. Automat. Control **53**, 189–200 (2008)
46. Eldar, A., Elowitz, M.B.: Functional roles for noise in genetic circuits. Nature **467**, 167–173 (2010)
47. Ribeiro, A.S., Kauffman, S.A.: Noisy attractors and ergodic sets in models of gene regulatory networks. J. Theor. Biol. **247**(4), 743–755 (2007)
48. Baron, M.H.: Reversibility of the differentiated state in somatic cells. Curr. Opin. Cell Biol. **5**(6), 1050–1056 (1993)
49. Johnson, N.C., Dillard, M.E., Baluk, P., McDonald, D.M., Harvey, N.L., et al.: Lymphatic endothelial cell identity is reversible and its maintenance requires Prox1 activity. Genes Dev. **22**, 3282–3291 (2008)
50. Takahashi, K., Yamanaka, S.: Induction of pluripotent stem cells from mouse embryonic and adult fibroblasts cultures by defined factors. Cell **126**(4), 663–676 (2006)
51. Takahashi, K., Tanabe, K., Ohnuki, M., Narita, M., Ichisaka, T., et al.: Induction of pluripotent stem cells from adult human fibroblasts by defined factors. Cell **131**(5), 861–872 (2007)
52. Vierbuchen, T., Ostermeier, A., Pang, Z.P., Kokubu, Y., Sudhof, T.C., et al.: Direct conversion of fibroblasts to functional neurons by defined factors. Nature **463**, 1035–1041 (2010)

Automatic Design of Boolean Networks for Modelling Cell Differentiation

Stefano Benedettini, Andrea Roli, Roberto Serra, and Marco Villani

Abstract A mathematical model based on Random Boolean Networks (RBNs) has been recently proposed to describe the main features of cell differentiation. The model captures in a unique framework all the main phenomena involved in cell differentiation and can be subject to experimental testing. A prominent role in the model is played by cellular noise, which somehow controls the cell ontogenetic process from the stem, totipotent state to the mature, completely differentiated one. Noise is high in stem cells and decreases while the cell undergoes the differentiation process. A limitation of the current mathematical model is that RBNs, as an ensemble, are not endowed with the property of showing a smooth relation between noise level and the differentiation stages of cells. In this work, we show that it is possible to generate an ensemble of Boolean networks (BNs) that can satisfy such a requirement, while keeping the other main relevant statistical features of classical RBNs. This ensemble is designed by means of an optimisation process, in which a stochastic local search (SLS) optimises an objective function which accounts for the requirements the network ensemble has to fulfil.

1 Introduction

Cell differentiation is the process whereby stem cells, which can develop into different types, become more and more specialised. A mathematical model of cell differentiation has been recently proposed by Serra et al. [1, 2]. The model is an

S. Benedettini (✉) · A. Roli
DEIS-Cesena *Alma Mater Studiorum* Università di Bologna, Bologna, Italy
e-mail: s.benedettini@unibo.it; andrea.roli@unibo.it

R. Serra · M. Villani
Faculty of Mathematical, Physical and Natural Sciences Università di Modena e Reggio Emilia & European Centre for Living Technology, Venice, Italy
e-mail: rserra@unimore.it; mvillani@unimore.it

S. Cagnoni et al. (eds.), *Evolution, Complexity and Artificial Life*,
DOI 10.1007/978-3-642-37577-4_5,

abstract one (i.e., it does not refer to a specific organism or cell type) and aims at reproducing the most relevant features of cell differentiation, which are the following:

1. there exist different degrees of differentiation that span from totipotent stem cells to fully differentiated cells;
2. there are both *deterministic* differentiation, where signals trigger the progress of multipotent cells into more differentiated types, in well-defined lineages, and *stochastic* differentiation, where populations of identical multipotent cells stochastically generate different cell types;
3. limited reversibility: differentiation is almost always irreversible, but there are limited exceptions under the action of appropriate signals;
4. induced pluripotency: fully differentiated cells can come back to a pluripotent state by modifying the expression of some genes;
5. induced change of cell type: modification of the expression of few genes can directly convert one differentiated cell type into another.

The differentiation model is based on a noisy version of a well-known model of gene networks, that is, the Random Boolean Network (RBN) model. In spite of the assumption of discreteness, RBNs have been proven to describe important experimental facts concerning gene expression [3–5]. The dynamics of "classical" RBNs is discrete and synchronous, so fixed points and cycles are the only possible asymptotic states in finite networks; typically, a single RBN owns more than one attractor. Attractors of RBNs are unstable with respect to noise even at low levels, e.g., transient flips of randomly chosen nodes. In fact, even if the flips last for a single time step, one often observes transitions from one attractor to another one. Ribeiro and Kauffman [6] observed that it is possible to identify in the attractors' landscape subsets of attractors, which they called *Ergodic Sets* (ESs), which entrap the system in the long time limit, so the system continues to jump between attractors which belong to the set. Unfortunately, it turns out that most noisy RBNs have just one such set: this observation rules out the possibility to associate them with cell types. The model proposed by Serra et al. overcomes this problem by observing that flips are a kind of fairly intense noise, as they amount to silencing an expressed gene or to express a gene which would otherwise be inactive: this event may happen with a very low probability in the cell lifetime. It is possible therefore to introduce a threshold θ, and neglect all the transitions whose occurrence probability is lower than θ. In such a way, the notion of ES has to be modified in that of *Threshold Ergodic Set* (TES_θ), a set of attractors linked only by jumps having a probability higher than θ, that entrap the system in the long time limit. A TES_θ is therefore a subset of attractors which are θ-reachable[1] from each other, directly or indirectly, and from which no transition can allow escaping. The threshold is related to the level of noise in the cell, and scales with its reciprocal (the frequency of flips) [1]. Hence, cell types are associated with TESs, which represent coherent stable ways

[1]Reachable by means of transition whose probability exceeds the threshold θ.

of functioning of the same genome even in the presence of noise. According to this framework, RBNs can host more than one TES, avoiding in such a way the problem that hampered the straightforward association of cell types with ES. At high noise level the system can jump among all the attractors, modelling stem cells while, as the threshold is increased (i.e., noise is reduced), the cell becomes entrapped in a smaller TES, that represents a multipotent cell. At very high threshold values all the attractors are also TES, a condition likely to describe final cell types. Indeed, there are experimental indications in favour of the key hypothesis that noise in stem cells is higher than in more differentiated ones. In this model, cell differentiation is an emerging property originating from the interactions of many genes: its main features therefore should be shared by a variety of different organisms.

This single model is able to capture all the phenomena encompassed by cell differentiation and its application to real cell differentiation processes is open to validations.[2] Nevertheless, the model could be ameliorated in some important aspects. In fact, while for RBNs it is true that the number of TESs increases with the threshold θ, the largest amount of this increase takes place for a very narrow range of values, necessitating in such a way a very sophisticated control to precisely tune the correct threshold for the required differentiation level. This paper presents a way of overcoming this weakness by providing a method for designing Boolean networks (BNs) such that the range of threshold values over which the number of TESs varies is as large as possible.

This contribution is structured as follows. Section 2 details the limitation of the current model and introduces the revision needed to accomplish the proper relation between number of TES and threshold. In Sect. 3, we illustrate the method we used for obtaining such BNs. Section 4 describes the experiments we made and present a statistical analysis of the results. Finally, Sect. 5 summarises the main outcome of this contribution and outlines future work.

2 Improved Model

The mathematical model previously illustrated can capture all the relevant phenomena of cell differentiation. Precise quantitative analyses can be undertaken depending on the availability of experimental data, which unfortunately are scarce and incomplete at the present time. However, the relevance of the model can be assessed in the context of the so-called *ensemble approach* [7], which aims at finding classes of genetic regulatory network models that match statistical features of living cells. In the case of cell differentiation, the model proposed by Serra et al. succeeds in describing the way in which a lineage tree is hierarchically organised and can also explain the other phenomena involved in the differentiation process from the ensemble approach standpoint. Nevertheless, as already emphasised, it requires

[2]There are some positive but not yet definitive experimental data.

a precise control in a very narrow threshold range, resulting in this aspect not completely satisfactory with respect to biological feasibility. In fact, in the ensemble of noisy RBNs considered in the model, the number of TESs varies approximately as a Heaviside step function of θ: one or very few TESs can be found for $\theta \leq \theta_0$ and the maximum number of TESs (equal to the number of attractors) is achieved with θ just above θ_0.[3] This behaviour is prone to errors in identifying the correct differentiation level within the lineage tree, and therefore biologically not plausible. Therefore, we would like to find an ensemble of BNs such that the main properties characterising RBNs are preserved and the number of TESs scales smoothly with the threshold θ. This goal can be achieved by applying a recently proposed method, which consists in converting the BN design problem into an optimisation one and solve it through stochastic local search (SLS) [8]. This automatic design method has been proven to successfully solve BN design problems [9–11] and will be detailed, for the case at hand, in the following section.

3 Methods

The problem of designing a BN or a set of BNs meeting given dynamical requirements can be stated as an optimisation problem. In particular, one has to define the decision variables and the objective function.[4] In principle, the optimisation problem can be solved by any search method; however, SLS has been shown to be very effective in tackling these kinds of problems and is thus our preferential choice. For this specific case, we assume that the topology of the network is set initially according to a random model [12] and kept fixed during search. The decision variables of the problem manipulate the Boolean functions of the BN nodes: for BNs with N nodes, each with K inputs, we introduce $2^K N$ Boolean decision variables, which define the transition functions of network nodes. Therefore, the local search can explore the space of all possible assignments of Boolean functions to the nodes, trying to minimise an objective function which estimates the distance between a current BN instance and the requirements posted.

The local search algorithm employed here is an Iterated Local Search (ILS), a well-known SLS framework successfully applied to many hard combinatorial optimisation problems. An outline of the high-level algorithm is given in Algorithm 1. In a nutshell, ILS applies a local search to an initial solution until it finds a local optimum (Line 3); then, it perturbs the solution (Line 5) and it restarts the local search (Line 6). A user-supplied acceptance criterion selects between the current best solution, also called *incumbent solution*, and the one found by the local search

[3] Of course, the property is typical of the ensemble and isolated exceptions could be found. The value θ_0 depends on the specific instance considered.

[4] We assume that constraints are either implicitly satisfied or that they are relaxed and included in the objective function.

Algorithm 1 Iterated local search high-level framework

```
1: INPUT: A LOCAL SEARCH
2: s ← generateInitialSolution()
3: s* ← localSearch(s)
4: while termination conditions not met do
5:     s' ← perturbation(s_best)
6:     s'_ls ← localSearch(s')
7:     s* ← acceptanceCriterion(s*, s'_ls)
8: end while
9: return s*
```

(Line 7). This design makes it possible to combine the efficiency of local search with the capability of escaping from the basins of attraction of local optima. An overview on the theory and applications of ILS can be found in [8, 13].

Like many metaheuristic frameworks, we must implement problem-specific choices in order to apply ILS to the problem at hand. Following the successful design described in [10], we committed to the following choices to instantiate the ILS framework.

Acceptance criterion: we accept a new solution if it is better than the incumbent one (extreme intensification).

Perturbation: for each node function, we perform a single random flip in the truth table. This choice makes ILS not too close to random restart, while keeping the perturbation computationally fast and easy to implement. As a drawback, we will see that local search moves can undo the effects of such a perturbation, albeit unlikely.

A clarification on the perturbation step is needed. In the limit case where the perturbation performed in Line 5 is independent of the incumbent solution s_{best}—for instance, s' could be randomly generated—ILS would degenerate into a random restart. Our experiments involve networks with input connectivity $K = \{2, 3\}$, therefore the truth tables have either four or eight elements. With such low figures, many more flips in the perturbation step would reduce the correlation between the incumbent solution s_{best} and the perturbed solution s' too much, thereby making our algorithm a "quasi-random" restart.

The last component to be defined is the embedded local search procedure. We opted for Stochastic Descent (SD), a simple local search in which a neighbour of the current solution is randomly picked and accepted if it is at least as good as the current one. The neighbourhood is implicitly defined by the modifications, or *moves*, that the current solution may undertake. In our implementation, a move consists in randomly choosing a node, then flipping a bit—chosen at random—in the truth table of its function.

As a final algorithmic remark, we can say that, in a sense, our combination of ILS with SD can be regarded as an iterated version of an *adaptive walk* in which restart is not random but performed in such a way that diversification is increased gradually.

3.1 Objective Function

The aim of our local search is to find BNs endowed with the two following properties:

1. the number of TESs should grow smoothly with the threshold θ;
2. attractors should be stable, i.e., the probability of transition $a \rightarrow a$, where a is an attractor, should be high. This property ensures that we can put into relation the attractors of the BN with the cell types of completely differentiated cells. Some attractors may be sensitive to small perturbations, but the majority should be stable [14].

The objective function closely reflects the requirements mentioned above. In particular, we opted for a linear relation between the number of TESs and the threshold θ, which is the simplest, yet effective choice. The computation of the objective function requires first the calculation of the *transition graph*, i.e., a directed graph whose vertices $V = \{v_i\}$ are attractors and edges (indicated as e_{ij}) represent transitions between attractors. Edges are weighted with transition probabilities (weights are denoted as w_{ij}). The transition graph $\mathscr{G}(V, E)$ is calculated by the algorithm specified in [2]. The objective function consists of the following two terms:

Attractor stability: the first contribution to the objective function is given by a term S calculated as the fraction of vertices in $\mathscr{G}$ with a self-loop with weight greater than or equal to 0.8. We chose this value in order to aim at stable enough attractors.

$$S = \frac{1}{|V|} \sum_{v_i \in V} [w_{ii} \geq 0.8] \,. \tag{1}$$

where $[P]$ is the Iverson Bracket and equals to 1 if predicate P is true, 0 otherwise.

Number of TESs as a linear function of θ: the second term E (E stands for error as we see in Eq. (2)) is calculated as follows: let us select a sequence Θ of n equally spaced values from interval $[0, 0.5[$, i.e., $\Theta = 0, \frac{1}{2n}, \frac{2}{2n}, \ldots, \frac{n-1}{2n}$. Let us also define the sequence $s_i, 0 \leq i \leq n$, as the number of TES_{Θ_i} (TESs with threshold Θ_i) in $\mathscr{G}$. This term is defined as:

$$E = \sum_{i=0}^{n} \left| \frac{s_i}{s_n} - \frac{i}{n} \right| . \tag{2}$$

Objective function: the objective function to be *minimised* is

$$(2 - S)E \tag{3}$$

Let us motivate our choices. The term S directly reflects the requirement on attractor stability. We should make clear that the resulting networks do not

necessarily have transition graphs with self-loops of weight 0.8, but they are forced to have most of the attractors with this property. Contribute E addresses the first requirement; basically, we ask for a sequence s_i that is as *smooth* as possible, i.e., we want s_i to gradually grow to its maximum value s_n, the linear growth we are using in this paper being the simplest option among an ampler set of possibilities. In Eq. (2) we divide s_i by s_n so that E is not dependent on the number of TESs. Finally, the two contributes are composed so that S takes the role of a *penalty*: the smaller S, the larger the term by which error E is weighted.

The choice of 0.5 as the maximum threshold value of sequence Θ, although partly arbitrary, is directly related to the first requirement on the stability of network attractors. By definition of transition graph, the sum of the weights of the outgoing edges for each vertex amounts to one. By the first requirement, we seek networks whose transition graphs have self-loops with high weight: this effectively limits the range of weights of the remaining edges.[5] By this argument, we understand that it is useless to calculate TESs by setting too high a threshold because, ideally, the number of TES_θs should be maximal for $\theta = 0.2$ and then it should stay constant for threshold values greater than 0.2. Of course, as remarked above, we cannot guarantee that all transition graphs have self-loops with weight 0.8 or greater, so we calculate TESs also for threshold values greater than 0.2.

We conclude this section with some remarks on our design choices. The choice of the specific objective function is being guided by the requirement of obtaining a smooth grow of the number of TESs as threshold θ increases. Since experimental data concerning the functional relation between number of TESs and θ are not available, we opted for a simple linear model. Hence the function used in Eq. (2). However, this function can be changed according to specific hypotheses, so as to have a better fit with given experimental data. Moreover, the choices of some parameters, for example, the values 0.8 (the self-loop desired weight) or 0.5 (the interval length spanned by sequence Θ), are partly arbitrary; evaluating the robustness of our results with respect to variations in these parameters lies beyond the purpose of the present work and will be the aim of further investigations.

Finally, an algorithmic detail. From a graph theoretical point of view, the number of TESs can be calculated as follows: first we remove from $\mathscr{G}$ all edges with weight less than θ, then we compute the *condensation* of $\mathscr{G}$ [15] and count the vertices with null out-degree.

4 Results

Typical RBNs are characterised by constant input connectivity K and Boolean functions chosen at random with on average $2^K p$ true entries in the truth table, where $p \in [0, 1]$ is called *bias*. Depending on the values of K and p the dynamics

[5] Ideally, the sum of transitions going out of a vertex, except for self-loops, should not be greater than 0.2.

of RBNs is *ordered* or *disordered* (also called *chaotic*, with a slight abuse of terms). In the first case, the majority of nodes in the attractors is frozen; any moderate-size perturbation is rapidly dampened and the network returns to its original attractor. Conversely, in disordered dynamics, attractor cycles are very long and the system is extremely sensitive to small perturbations: slightly different initial states lead to exponentially diverging trajectories in the state space. RBNs temporal evolution undergoes a second order phase transition between order and chaos, governed by the following relation that defines a curve in the bidimensional space of parameters K and p:

$$K_c = \frac{1}{2p_c(1-p_c)} \tag{4}$$

where the subscript c denotes the critical values [16]. The curve defined by Eq. (4) separates the ordered from the disordered regime [17]. RBNs whose parameters are chosen along the *critical* line are the ones that best match living cell features [5,18].

We tested our algorithm on two test sets, both composed of *critical* RBNs with $N = 100$ nodes and constant in-degree. The first test set consists of 30 critical RBNs with in-degree $K = 2$ (whence $p = 0.5$); the second test set contains 30 critical RBNs with in-degree $K = 3$ (whence $p \approx 0.788$). Networks in these two ensembles constitute the initial solutions of our local search and will be collectively referred to as *initial ensemble*. Similarly, the set of BNs obtained after running our local search constitutes our *optimised ensemble*.

In order to compute our objective function we had to compute the transition graph. We initialise the algorithm with attractors found after a sample of 1,000 initial conditions (more attractors may of course be found during algorithm execution and are recursively considered in the algorithm). We considered only trajectories with at most 1,500 steps: if an attractor is not found in this number of steps, the sample is discarded. As for ILS, we set a runtime limit of 3 h per experiment. All experiments were executed on a desktop PC equipped with a Intel Core 2 Quad 2.83 GHz with 8 GB RAM and running Ubuntu Server 10.04; the implementation of the SLS algorithm presented in this paper was written in C++ and was compiled with GCC 4.4.3 with the `-O3` optimisation option turned on.

4.1 Analysis of Network Properties

In order to analyse a BN we sampled its state space in 100,000 random initial conditions, since an exhaustive test would be prohibitive. For each network, we recorded the number of attractors, their relative basin sizes and their periods. In addition, we computed the transition graph and the sequence of the number of TESs (actually, the sequence s_i as defined in Sect. 3.1).

The first remarkable result is that the number of TESs of the networks designed through our local search smoothly increases with the threshold. A typical case is depicted in Fig. 1, where we can observe that the number of TESs increases within

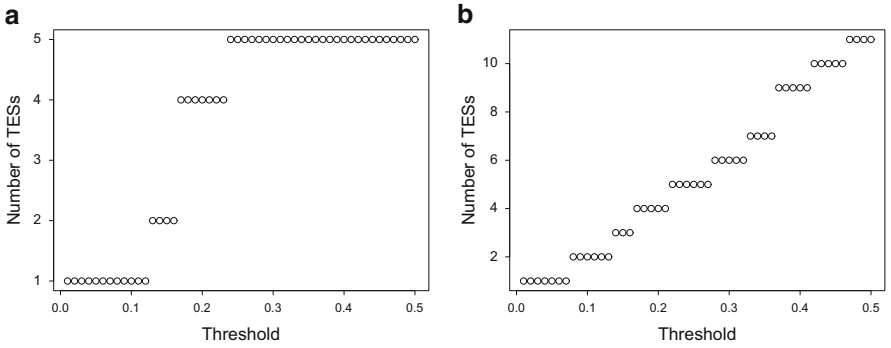

Fig. 1 Number of TES as a function of the threshold in a random BN (**a**) and an automatic designed one (**b**). (**a**) is the typical plot that characterises RBNs and displays their undesirable features, with respect to the cellular differentiation model, as explained at the end of Sect. 1. (**a**) A random BN. (**b**) Automatically designed BN

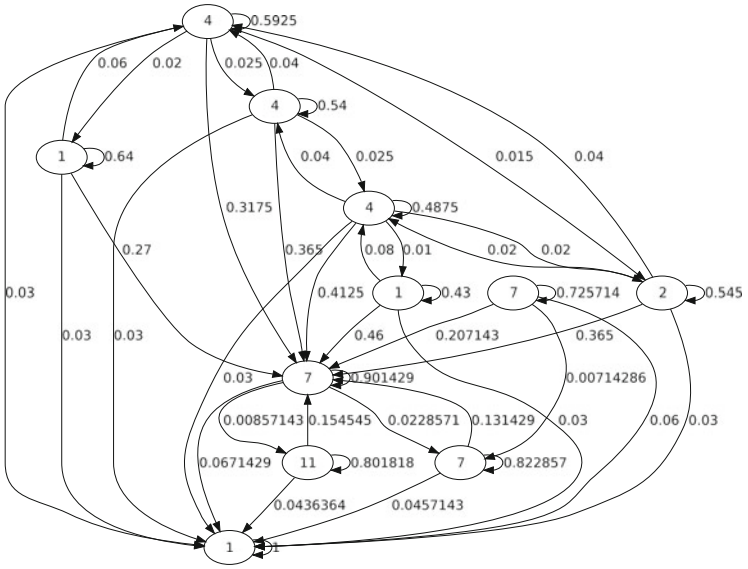

Fig. 2 Attractor transition graph for an automatically designed BN

a wide threshold range. The transition graph corresponding to this automatically designed network is drawn in Fig. 2. This property is common to almost all the networks generated by the search procedure and can be considered as an invariant of the ensemble.[6]

[6]Since the search process is stochastic, we can consider our design method as a biased sampling in the space of BNs.

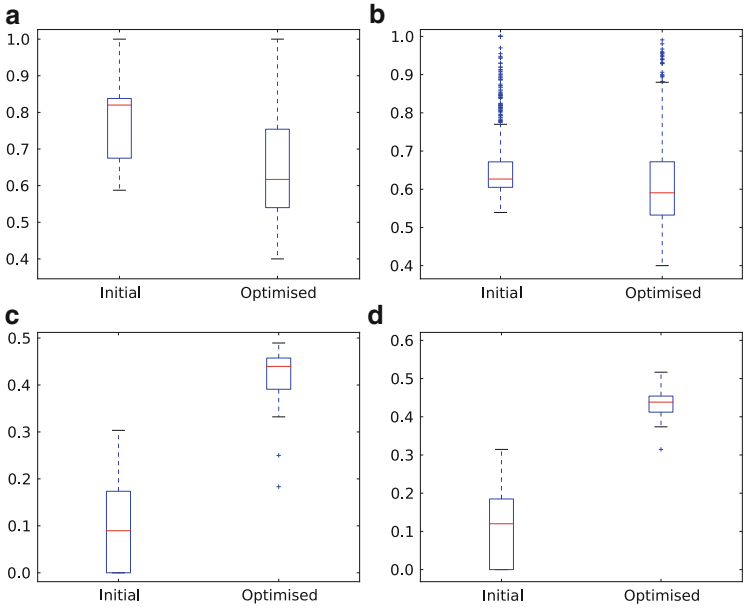

Fig. 3 Attractor stability and transitions for initial and optimised BNs in all test sets. (**a**) and (**b**) depict the distribution of attractor stability across all networks in either test set; the distribution is constructed by merging all data samples gathered. Data depicted in (**c**) and (**d**) characterise the distribution of edge weights in the transition graph as explained in the text. (**a**) Attractor stability ($K = 2$). (**b**) Attractor stability ($K = 3$). (**c**) Transitions ($K = 2$). (**d**) Transitions ($K = 3$)

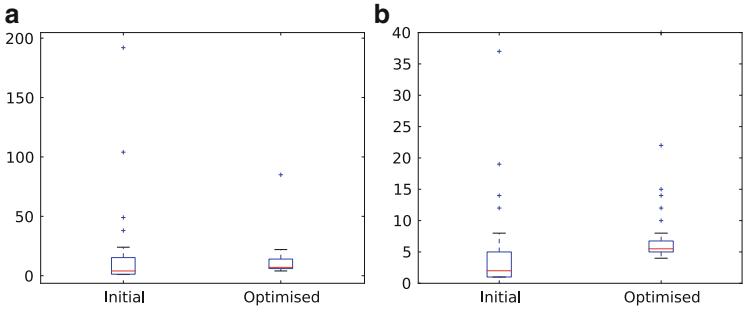

Fig. 4 Distributions of the number of attractors (**a** and **b**) on initial and optimised networks. (**a**) Number of attractors ($K = 2$). (**b**) Number of attractors ($K = 3$)

Statistics on attractors stability, weights in the transition graph, number of attractors, their relative basin sizes and their periods are summarised by boxplots in Figs. 3, 4 and 5. Boxplots graphically summarise the main statistics of a distribution [19]. The values represented are:

– The median (second quartile of the distribution)—line inside the box.
– The lower quartile (first quartile, Q1)—lowermost side of the box.

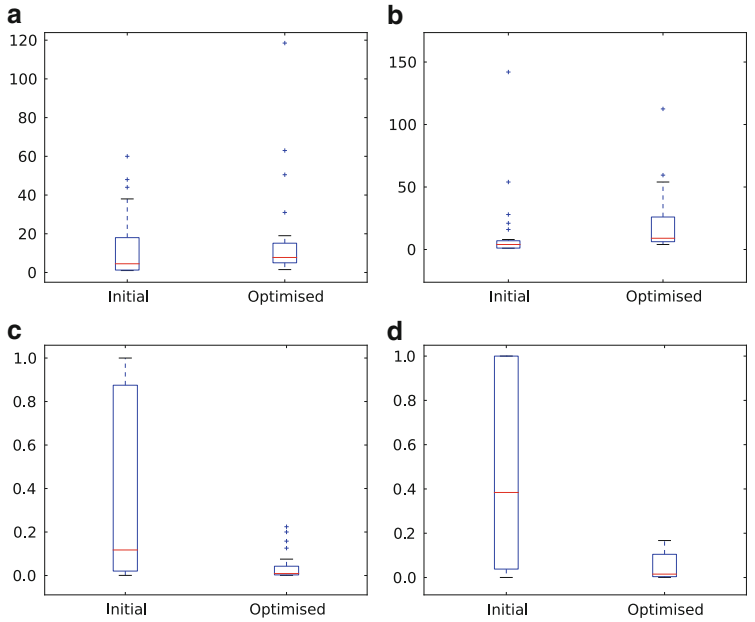

Fig. 5 (**a**)–(**d**) summarise the distribution of *median* attractor period (**a** and **b**) and *median* normalised basin size (**c** and **d**), respectively. (**a**) Attractor periods ($K = 2$). (**b**) Attractor periods ($K = 3$). (**c**) Attractor basins ($K = 2$). (**d**) Attractor basins ($K = 3$)

- The upper quartile (third quartile, Q3)—uppermost side of the box.
- Uppermost and lowermost whiskers are drawn at $1.5 \times (\text{Q3} - \text{Q1})$, from the first and third quartile, respectively.
- Outliers (i.e., values lying far from first and third quartile more than $1.5 \times (\text{Q3} - \text{Q1})$)—open dots.

Each figure depicts the main statistics of a measure on the initial ensemble (left boxplot) and the optimised ensemble (right boxplot) for all test sets.

Figure 3 shows two measures that try to characterise weights of the transition graph and thereby statistically demonstrate the effectiveness of our method in designing BNs with the characteristics stated in Sect. 3.1. The networks' transition graphs can be represented by a weight matrix $\boldsymbol{W} = (w_{ij}), 0 \leq w_{ij} \leq 1$, where w_{ij} represents the probability of the network to go from attractor i to the basin of attractor j after a random flip; $w_{ij} = 0 \Leftrightarrow \nexists e_{ij}$. Self-loops weights w_{ii} indicate how insensitive to random flips the attractor i is; in general, we want this probability to be high.

Figure 3a, b describes the distribution of attractor stability in the initial and optimised ensembles. Since we have one stability figure for each attractor for each network, in order to clearly report summaries of our results, we decided to merge together the data for all networks in each ensemble. Figure 3c, d characterises, instead, the distribution of the weights of edges $e_{ij}, i \neq j$, i.e., we disregard self-loop. For each BN, we collect in a set P all non-zero elements of $\boldsymbol{W}$ outside the

main diagonal; afterwards, we compute the difference $\max P - \min P$. Figure 3c, d shows the distribution of such differences for the initial and the optimised networks. It can be observed that attractor stability (Fig. 3a, b) is lower but close to the initial ensemble (requirement *(1)* in Sect. 3.1); at the same time, edge weights $w_{ij}, i \neq j$, are more spread out, according to our objective (requirement *(2)* in Sect. 3.1). We also remark that, for the reasons explained in Sect. 3.1, the quantity $\min P$ is rather small, so we actually have $\max P - \min P \approx \max P$.

Figure 4 depicts the distribution of the number of attractors. As a result of the search process, the number of attractors does not vary in a statistically significant way, although it seems to grow a bit.

Figure 5 shows basin sizes and attractor periods. Since a network can have more than one attractor, a single BN is characterised by *a set* of attractor periods and basin sizes. To adequately summarise these two statistics, boxplots in Fig. 5 depict the distribution of the *median* basin size and attractor period calculated on each BN. We observe that attractors' period does not statistically vary, but the distribution of basin sizes is remarkably different; specifically, the search process tends to shrink basin sizes. Intuitively, we can say that attractors with small basin sizes are likely to be less robust than attractors with larger basin sizes simply because the basins of the latter contain more states. Therefore, one would expect that networks with smaller basins (a feature typical of optimised networks) are also characterised by unstable attractors. However, Fig. 3a, b clearly shows that attractor stability is essentially unchanged. From these data we can conclude that the search process reorganises the basins of the attractors in such a way as to satisfy our stated requirements: the basins are therefore "rebalanced" so as to have generally stable attractors. It appears that, in order to achieve this goal, our local search had to reduce the size of some of the larger basins. This is the motivation why the median stability in Fig. 3a, b slightly decreases (about a 0.2 decrease for networks with $K = 2$ and a decrement less than 0.1 for networks with $K = 3$).

5 Conclusion and Future Work

In this paper we have proposed an improvement of a mathematical model for cell differentiation that makes use of RBNs. An ensemble of BNs that match the dynamical requirements deriving from biological plausibility has been designed by means of an optimisation process that uses SLS. The BNs generated are characterised by a more realistic relation between the number of TESs and the threshold, conserving the other relevant properties of the RBN ensemble. In particular, results show that in the ensemble generated by the optimisation process the number of TESs grows smoothly with the threshold and attractors are robust.

Future work will address the extension of the model by introducing in the optimisation process further features of cell differentiation, such as properties concerning deterministic and stochastic differentiation. Furthermore, besides the ensemble approach, we are planning to validate the model against experimental data collected for specific organisms.

References

1. Serra, R., Villani, M., Barbieri, A., Kauffman, S.A., Colacci, A.: On the dynamics of random Boolean networks subject to noise: Attractors, ergodic sets and cell types. J. Theor. Biol. **265**(2), 185–193 (2010)
2. Villani, M., Barbieri, A., Serra, R.: A dynamical model of genetic networks for cell differentiation. PLoS One **6**(3), e17703, 1–9 (2011)
3. Serra, R., Villani, M., Semeria, A.: Genetic network models and statistical properties of gene expression data in knock-out experiments. J. Theor. Biol. **227**, 149–157 (2004)
4. Aldana, M., Balleza, E., Kauffman, S.A., Resendiz, O.: Robustness and evolvability in genetic regulatory networks. J. Theor. Biol. **245**, 433–448 (2007)
5. Balleza, E., Alvarez-Buylla, E.R., Chaos, A., Kauffman, S.A., Shmulevich, I., Aldana, M.: Critical dynamics in genetic regulatory networks: Examples from four kingdoms. PLoS One **3**(6), e2456 (2008)
6. Ribeiro, A.S., Kauffman, S.A.: Noisy attractors and ergodic sets in models of gene regulatory networks. J. Theor. Biol. **247**(4), 743–755 (2007)
7. Kauffman, S.A.: A proposal for using the ensemble approach to understand genetic regulatory networks. J. Theor. Biol. **230**, 581–590 (2004)
8. Hoos, H.H., Stützle, T.: Stochastic Local Search: Foundations and Applications. Morgan Kaufmann, San Francisco (2005)
9. Roli, A., Arcaroli, C., Lazzarini, M., Benedettini, S.: Boolean networks design by genetic algorithms. In: Villani, M., Cagnoni, S. (eds.) Proceedings of CEEI 2009 - Workshop on Complexity, Evolution and Emergent Intelligence, Reggio Emilia, Italy, 2009
10. Benedettini, S., Roli, A., Serra, R., Villani, M.: Stochastic local search to automatically design Boolean networks with maximally distant attractors. In: Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcázar, A., Merelo, J., Neri, F., Preuss, M., Richter, H., Togelius, J., Yannakakis, G. (eds.) Applications of Evolutionary Computation. Lecture Notes in Computer Science, vol. 6624, pp. 22–31. Springer, Heidelberg (2011)
11. Roli, A., Manfroni, M., Pinciroli, C., Birattari, M.: On the design of Boolean network robots. In: Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcázar, A., Merelo, J., Neri, F., Preuss, M., Richter, H., Togelius, J., Yannakakis, G. (eds.) Applications of Evolutionary Computation. Lecture Notes in Computer Science, vol. 6624, pp. 43–52. Springer, Heidelberg (2011)
12. Kauffman, S.A.: The Origins of Order: Self-Organization and Selection in Evolution. Oxford University Press, Oxford (1993)
13. Lourenço, H., Martin, O., Stützle, T.: Iterated local search. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics. International Series in Operations Research & Management Science, vol. 57, pp. 320–353. Springer, New York (2003)
14. Li, F., Long, T., Lu, Y., Ouyang, Q., Tang, C.: The yeast cell-cycle network is robustly designed. Natl. Acad. Sci. **101**(14), 4781–4786 (2004)
15. Wikipedia: Strongly connected component — Wikipedia, The Free Encyclopedia, 2011. [Online; accessed November 18, 2013]
16. Derrida, B., Pomeau, Y.: Random networks of automata: A simple annealed approximation. Europhys. Lett. **1**(2), 45–49 (1986)
17. Bastolla, U., Parisi, G.: Closing probabilities in the Kauffman model: An annealed computation. Phys. D. **98**, 1–25 (1996)
18. Shmulevich, I., Kauffman, S.A., Aldana, M.: Eukaryotic cells are dynamically ordered or critical but not chaotic. Proc. Natl. Aca. Sci. USA **102**(38), 13439–13444 (2005)
19. Frigge, M., Hoaglin, D.C., Iglewicz, B.: Some implementations of the boxplot. Am. Stat. **43**(1), 50–54 (1989)

Towards the Engineering of Chemical Communication Between Semi-Synthetic and Natural Cells

Pasquale Stano, Giordano Rampioni, Luisa Damiano, Francesca D'Angelo, Paolo Carrara, Livia Leoni, and Pier Luigi Luisi

Abstract The recent advancements in semi-synthetic minimal cell (SSMC) technology pave the way for several interesting scenarios that span from basic scientific advancements to applications in biotechnology. In this short chapter we discuss the relevance of establishing chemical communication between synthetic and natural cells as an important conceptual issue and then discuss it as a new bio/chemical-information and communication technology. To this aim, the state of the art of SSMCs technology is shortly reviewed, and a possible experimental approach based on bacteria quorum sensing mechanisms is proposed and discussed.

1 Chemical Communication as a Bio/Chemical-Information and Communication Technology

Among the most fascinating novelties introduced by contemporary science, there is the application of biological paradigm to the development of new technologies. One of the latter is the bio/chemical information and communication technology (bio/chem-ICT), which aims at extending the well-known field of ICT, classically based on the transmission of electrical or electromagnetic signals, to the bio/chemical world of molecules.

Recently, Suda and collaborators have highlighted the role of molecular communication in ICT. In their recent review [1]—which is a good starting point for the discussion we would like to undertake in this book chapter—these authors describe the chemical signaling underlying biological communication and provide

P. Stano (✉) • G. Rampioni • F. D'Angelo • P. Carrara • L. Leoni • P.L. Luisi
Department of Sciences, University of Roma Tre, V.le G. Marconi 446, 00146 Rome, Italy
e-mail: stano@uniroma3.it

L. Damiano
CERCO (Research Center on Complex Systems), University of Bergamo, P.le S. Agostino 2, 24129 Bergamo, Italy

S. Cagnoni et al. (eds.), *Evolution, Complexity and Artificial Life*,
DOI 10.1007/978-3-642-37577-4_6, © Springer-Verlag Berlin Heidelberg 2014

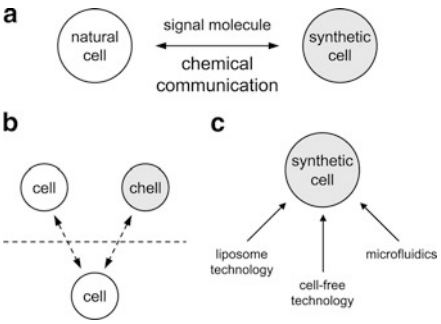

Fig. 1 (**a**) Chemical communication between natural and synthetic cells; (**b**) the Turing-like test for chemical cells (*chells*), see the text; (**c**) semi-synthetic minimal cell technology

a rational description of this molecular communication in engineering terms. They suggest that man-made molecular systems, if endowed with proper interfaces, could communicate with living biological systems, moreover they discuss the potential impact of developing a new bio/chem-ICT paradigm on the medical field (drug targeting/drug delivery).

Bio/chem-ICT is still in its infancy, but we can already outline here one of the most important goals: the creation of synthetic (artificial) cells that are able to communicate—via molecular mechanisms—with natural cells (Fig. 1a). In this scenario, synthetic cells—constructed by assembling separated components (by the so-called bottom-up approach)—will be used as a kind of intelligent *wet-soft robots* capable of communicating with natural cells. The synthetic cells might activate responses in agreement with the "meaning" they associate to the interactions with natural cells (the connection of these arguments with the "bio-semiotic" concepts is a related fascinating facet). As already highlighted by Suda and collaborators, this goal would represent a way for interfacing synthetic (and possibly programmable) systems to natural ones.

With these premises, the main questions are: is it possible, according to the current technology, to imagine a research program aimed at developing a chemical communication between synthetic and natural cells? What would be its impact in basic science and ICTs?

In this chapter, we will try to answer these questions by firstly illustrating the current state of the art in the construction of synthetic cells, and at this aim we will discuss the most successful approach, called *semi-synthetic* [2]. Then we will examine some possible experimental approaches that derive from combining the semi-synthetic minimal cell (SSMC) technology with the mechanisms of bacterial communications, and in particular those involved in quorum sensing (QS).

Before turning into technical discussion, however, let us summarize a few remarks on the conceptual interests toward the communication between synthetic and natural cells. Note also that another version of this chapter has been recently published [3].

2 Life and Communication

2.1 The Autopoietic Perspective

The theory of *autopoiesis* (self-production), developed in the 1970s by Maturana and Varela [4], deals with the most classical question of biology: what is life? The authors developed their theory on the basis of two hypotheses, according to which: (a) the distinctive property of living systems is its autopoiesis, that is, the capability of these systems of producing and maintaining their material identity through an endogenous processes of synthesis and destruction of their own components; (b) autopoiesis is a global property of living systems, which does not rely on their physicochemical components taken separately, but on the way these components are organized within the systems. On this ground, Maturana and Varela addressed the issue of defining life as the problem of determining what kind of organization supports the biological behavior of self-production. They provided a rigorous solution at the level of the minimal cell. This solution consists in the notion of "autopoietic organization," which aims at characterizing the "fundamental" biological organization.

> [The autopoietic organization is] (...) a network of processes of production (transformation and destruction) of components that produces the components which: (i) through their interactions and transformations continuously regenerate and realize the network of processes (relations) that produced them; and (ii) constitute (...) a concrete unity in the space in which they (the components) exist by specifying the topological domain of its realization as such a network [5].

This concept is at the basis of the synthetic biology's interest for autopoiesis, as it provides three theoretical tools to ground the production of minimal synthetic living systems able of communicating with natural living systems. Very schematically, these tools can be described as follows.

First tool: *an operational definition of life characterizing a mechanism able to generate minimal living systems.* The notion of autopoietic organization proposes a "synthetic" or "constructive" definition of living systems, as it characterizes them not by listing a set of properties, but by specifying a mechanism able to generate these systems and their dynamics of self-production.

Second tool: *a theoretical description of the interaction between autopoietic systems and their environment.* According to Maturana and Varela, the systems produced by the mechanism of autopoiesis are not trivial objects, which passively undergo environmental pressures. On the contrary, autopoietic systems can perceive exogenous variations as local alterations of their internal processes of self-production and can react to them through an activity of self-regulation, that is a series of active changes in their elementary processes that compensate external perturbations [6]. Maturana and Varela interpreted this interactive behavior as the basic "cognitive" behavior, as, thorough it, the autopoietic systems generate internal operational meanings for the perceived external variations. These meanings are expressed in terms of dynamical schemes of self-regulation, which externally

appears as actions oriented to conservation (e.g., absorbing a molecule of sugar, overcoming an obstacle . . .). This idea is at the basis of the autopoietic theory of the interaction between living systems and their environment, expressed by the notion of "structural coupling." According to the latter, the autopoietic unit and its environment are two systems permanently involved in a dynamics of reciprocal perturbations and endogenous compensations, in which the autopoietic system continuously generates and associates exogenous variations with operational meanings of self-regulation that allows it to perpetuate its process of self-production in an ever-changing environment.

Third tool: *a theory of communication between living systems.* On the basis of the notions of autopoietic organization and structural coupling, Maturana and Varela [6] proposed a theory of communication between autopoietic systems, which characterizes it as a dynamics of reciprocal perturbations and compensations in which each system generates and associates the exogenous perturbations produced by the other systems with internal operational meaning expressed in terms of endogenous patterns of self-regulation. This implies the possibility for the autopoietic systems of reciprocally influencing and, in this sense, coordinating their respective cognitive activities. They can face environmental variations together, thorough inter-dependent behaviors of self-production, self-regulation, meaning generation and production of external actions.

2.2 The Imitation Game: A Turing-Test Like Approach

Cronin et al. [7] proposed a sort of Turing test for chemical cells (*chells*), as an analogy with the well-known Turing test for assaying artificial intelligence. The goal was to devise a conceptual test that could help in the field of artificial cellularity. The authors aimed to the recognition of life by means of a "cellular imitation game" setup (Fig. 1b). The imitation becomes perfect when a natural cell as interrogator cannot distinguish one of its own kinds from a synthetic cell. In their view, the authors clearly points to the issue of synthetic cell/natural cell recognition and communication, even if not directly referring to molecular communication.

More recently, Davis and coworker [8] also published an experimental report on the first attempt to establish a synthetic communication between "chemical cells" (*chells*) and natural living cells. This report, to the best of our knowledge, is the only one on this very new topic. The precursors of the "formose" reaction were encapsulated within liposomes. The product of these reactions is linear and branched carbohydrates that resemble some naturally occurring sugars. One class of products of the intra-liposomal formose reaction escaped from liposomes through a channel (α-hemolysin), and spontaneously reacted with the borate ions present in the external medium. The resulting furanosyl boronates are structurally very similar to a specific class of bacterial signal molecules known as autoinducer-2 (AI-2), widespread among bacteria for cell–cell communication. Authors demonstrated that the furanosyl boronates produced by the *chells* were effective in triggering light

emission in *Vibrio harveyi*, a phenotype that in this organism is naturally controlled by AI-2 signaling. This work demonstrated the feasibility of generating a synthetic entity able to send a signal to a natural receiver, a breakthrough in the field of bio/chem-ITC.

3 The Concept and the Technology of SSMCs

3.1 The Concept of Minimal Cells: From Origin of Life to Synthetic Biology

Although already present in the literature on origin of life [9, 10], the modern concept of minimal cells was developed in the laboratory of Pier Luigi Luisi at the ETH Zurich, in the 1990s. Intrigued by the autopoietic theory, Luisi [11] tried to construct in the laboratory the first autopoietic minimal cells by using firstly reverse micelles, then normal micelles, and finally vesicles. All these microcompartments consist in self-assembling structures generally formed by surfactants or lipids. By chemical producing boundary-forming molecules, it was possible to observe the autopoietic growth of these microcompartments. A convenient way for establishing a more complex and recursive autopoietic growth is known as the "semi-synthetic" approach. It consists in the encapsulation of the minimal number of compounds, namely DNA, enzymes, ribosomes, and all required macromolecules, inside lipid vesicles. The corresponding structures are known as "semi-synthetic" minimal cells because natural compounds are used for their construction. This differs from the totally "synthetic" approach where not-natural (synthetic) compounds can be employed, at least in principle [12]. The goal of the semi-synthetic approach is setting up a minimal genetic/metabolic dynamics inside such compartments. In this short chapter there is no space to discuss the most intriguing aspects of the minimal cell construction: when a minimal cell can be defined as "alive"? And what does "alive" mean? The interested reader can find a deeper discussion of these aspects in a recent review [2]. These issues are especially important when minimal cells are intended as model of primitive cells.

Recently, after the advent of synthetic biology, there has been a new flourishing interest toward minimal cell research [13]. This is due to the fact that minimal cells, thanks to their minimal complexity, can be built in the laboratory to perform useful functions, without necessarily being related to the origin of life research. In this view, minimal cells are not only a tool for understanding the origin of cells, but can be important tools for diverse applications (as the bio/chem-ICT ones). The last 10 years have been characterized by an intense research activity aimed at understanding and controlling the construction of SSMCs from separated parts. Most of the work has been focused on the production of functional proteins inside liposomes, but also other goals have been achieved. The production of proteins is of special relevance because it allows the generation of new functions inside liposomes (i.e., synthesis

of enzymes that catalyze useful reactions). Not many research groups are currently involved in these studies, but the community is indeed growing (for a review on the latest achievements, see [14]). In the next paragraphs, we will shortly review what is the current knowledge on SSMC construction from the viewpoint of two important aspects: liposome technology and cell-free technology (Fig. 1c).

3.2 Liposome Technology: From Classical Methods to the "Droplet Transfer" Strategy and Beyond (First Vesicles from Microfluidic Devices)

SSMCs are based on liposomes. Liposome technology is a rather well-developed technology that progressed mainly for producing drug-containing liposomes, for drug-delivery applications. There are plenty of methods for liposome preparation, but only few of them have been used in the field of SSMCs studies. In particular, because SSMCs are often intended as models for primitive minimal cells, the preparation method should also be—if possible—compatible with allegedly prebiotic conditions.

Two methods have found a widespread application in SSMC studies. The first one consists in hydrating a previously deposited lipid film with a mixture of solutes of interest. Following the swelling of lipid bilayers, a population of liposomes is formed, heterogeneous size and morphology (e.g., unilamellar, or multilamellar, or multi-vesicular vesicles), witnessing that lipid vesicles form according to individual kinetic paths. As expected, also the *entrapment* of solutes in this heterogeneous population of vesicles is rather heterogeneous and not very efficient. A typical way for improving the solute encapsulation and homogenize the liposome suspension consists in repetitive freezing and thawing cycles, possibly followed by the classical extrusion procedure (a typical example is found in [15]). The problem of solute entrapment becomes critical when one considers the low probability of co-entrapping several compounds in the same lipid vesicle. We have recently investigated this aspect during the studies on the construction of SSMCs with minimal physical size [16], and the effect of "spontaneous crowding" has been reported [17]. Another well-known method for liposome preparation that partially overcome the issue of poor solute entrapment consists in hydrating the ghosts of previously formed liposomes in form of freeze-dried cake [18]. This method also produces a population of vesicles with a broad distribution, and it has been studied in great detail by flow cytometry [19].

Despite the above-mentioned "spontaneous crowding" effect [17], that actually involves about 1 % of vesicle preparation, it is clear that the film hydration and freeze-dried cake hydration method lack the control of solute internalization. They are perfectly suitable when SSMCs are studied as a case of self-organizing microcompartment, but are less valid when a technology for producing SSMCs needs to be developed (as for the case of bio/chem-ICT). A new method rapidly

emerged in the past few years as the method of choice when a complex mixture of molecules needs to be encapsulated inside liposomes. This method was introduced by Weitz and coworkers in 2003 [20], and it is currently studied and developed in almost all laboratories working in this field. The method consists in transferring a water-in-oil lipid-stabilized droplet, easily filled with the solute of interest, through a lipid-containing interface. In this way it is possible to produce giant lipid vesicles (GVs) in a reproducible way with good yield (e.g., 5–10,000 GVs/μL).

The recent advances in microfluidic technology might contribute in the near future to the SSMCs technology. It is worth noting, in fact, that in the last 4 years some interesting reports have shown the possibility of producing GVs directly in microfluidic devices. A short review on these methods is available in [14]—see also [21]. If this technology will become robust and available for most laboratories, it is foreseeable that the next generation of SSMCs could derive from microfluidic controlled assembly—which produces GVs with high reproducibility—rather than from the spontaneous, heterogeneous (yet interesting) self-assembly that has characterized the research done till now.

3.3 Cell-Free Systems as a Typical Synthetic Biology Toolbox

The second ingredient for constructing SSMCs in the laboratory is a cell-free system. This is essential because the semi-synthetic approach foresees the assembly of a cell from separated components and cannot therefore relies on components existing in pre-formed cells, as it happens in many other synthetic biology approaches. The choice of the cell-free system to be entrapped inside liposomes clearly depends on the function to reconstitute. For example, there have been studies on RNA synthesis from DNA- or RNA-template, on DNA amplification via polymerase chain reaction, and, mostly, on the coupled transcription/translation reaction (from DNA to RNA to protein). The interested reader can find technical and more detailed information on these systems in the recent review [14].

Here it is important to remark that after its introduction in 2001 [22], the so-called protein synthesis using recombinant elements (PURE) system is considered the "standard" cell-free system for constructing SSMCs in the laboratory. It fits perfectly with the requirement of full-characterized parts/devices/systems in synthetic biology. The PURE system includes 36 purified enzymes, ribosomes, and a tRNAs mixture—as well as low molecular weight compounds—for a total of about 80 macromolecules. It represents the minimal reconstituted system capable of synthesizing a functional protein, and it is therefore suitable for SSMCs studies. The PURE system has replaced the use of cell extracts, with unknown composition. As already highlighted, the production of proteins inside liposomes is a key intermediate step toward the construction of more complex SSMCs.

The analysis of the literature shows that it is possible to synthesize, inside liposomes, water-soluble proteins and enzymes like green fluorescent protein, T7 RNA polymerase, α-hemolysin, Qβ-replicase, β-galactosidase, and β-glucuronidase

(see [14] for details). In general terms, therefore, it can be said that the synthesis of a water-soluble protein in its folded conformation in SSMCs should not be considered a problem (in absence of important post-translation modifications). Different is the case of membrane associated or integral membrane proteins, where the only available report [23] focused on two acyltransferases shows how the chemical composition of the SSMC membrane strongly affects the synthesis, the structure and the functionality of this kind of proteins.

Interestingly, it has also been reported that α-hemolysin, when produced inside liposomes, spontaneously forms pores (i.e., channels) in the membrane. Due to its specific size, these pores allow small molecules (<3 kDa) to freely enter/exit the liposome, whereas enzymes, RNAs and DNA remain entrapped inside the compartment [24]. This implies that SSMCs can release/uptake small molecules to/from the environment. It should also be remarked that some molecules can freely diffuse across lipid membranes without the need of a pore.

4 A Research Program on Synthetic Cell/Natural Cell Communication

Based on the current state of the art of SSMCs technology, can we imagine a realistic scenario where a chemical communication can be established between SSMCs and natural cells? The previously reported work of Davis and coworker [8] demonstrates that this is an achievable goal. However, to go beyond the simple case of the Davis' *chells* and use SSMCs, several aspects must be considered. In order to understand how communicating SSMCs must be designed, it is useful to make a short survey on the way natural cells communicate. At this aim we believe that bacterial communication should be taken as a paradigmatic example and model to start this enterprise.

4.1 Bacterial Communication and Quorum Sensing

Bacteria live preferably as communities. The discovery of bacterial communication via chemical signaling has been one of the most exciting breakthroughs in microbiology. Thanks to chemical communication, bacteria understand the structure of their population and often respond with cooperative behavior, reaching—as a community—goals that are impossible for each single individual [25, 26].

The most representative example of bacterial communication is quorum sensing (QS), a cell–cell signaling system that allows a coordinate reprogramming of gene expression in response to cell density. QS takes its name from the fact that a response is achieved when a signal compound reaches a certain concentration threshold (corresponding to a certain bacterial cell density, the "quorum") [27].

Bacteria use a very large variety of biochemicals to communicate. This means that bacterial communities, often consisting of different species, communicate thanks to the specificity of the signal production and signal reception pathways.

Among the most well-known types of QS mechanisms, it is known that gram-negative and gram-positive bacteria usually rely on acylated homoserine lactones (AHLs) and small peptides, respectively, as signal molecules. A third class of QS signals, produced by both gram-negative and gram-positive bacteria, is known as autoinducer-2 (AI-2). Can we exploit the simplest of these communication mechanisms to construct SSMCs capable of communicating with bacteria and produce a QS-like response?

4.2 Towards the Communication Between Synthetic and Natural Cells

It is clear that—in order to communicate with natural cells—SSMCs must implement the molecular devices for encoding, sending, receiving, and decoding a chemical signal.

It stems directly from the analysis of how bacteria communicate that the molecular communication consists in five distinct operational steps: namely, (1) encoding a message as a molecule, (2) send/export the molecule, (3) a propagation/transportation step, (4) receive/import the molecule, (5) decoding the message. Different physical or (bio)chemical devices can be associated with each step, depending on the kind of communication implemented by living cells. According to the synthetic biology terminology, we might call the biochemical machineries required to accomplish these steps as "devices." Note, however, that not all the operational steps described above need to be associated with a dedicated device. For instance, signal molecule import can occur by passive diffusion into the receiving cell, and only for some signal molecules a dedicated membrane receptor is required. In a similar way, some signal molecules freely diffuse outside the membrane of the producing bacteria, while others require an export apparatus.

In order to design and construct a minimal set of devices for SSMCs/natural cell communication (and, in a perspective, for the communication between SSMCs), it is useful to shortly review the main communication machineries involved in bacterial QS. Figure 2a, b summarizes in a very schematic way these mechanisms, considering the above-mentioned steps (from 1 to 5).

A first and relatively simple case is the communication based on AHLs. These molecules are produced from appropriate precursors ("Ps" in the Fig. 2a), which are *S*-adenosyl methionine and an acyl-carrier protein. These precursors are processed by a LuxI- or LuxM-type synthases to give the signal molecule (AHL). Thanks to their chemical structure, AHLs (especially those having short acyl chains) can freely diffuse across the membrane of the producing cell (the sender) and propagate in the medium, reaching another cell (the receiver). AHLs can freely permeate into the

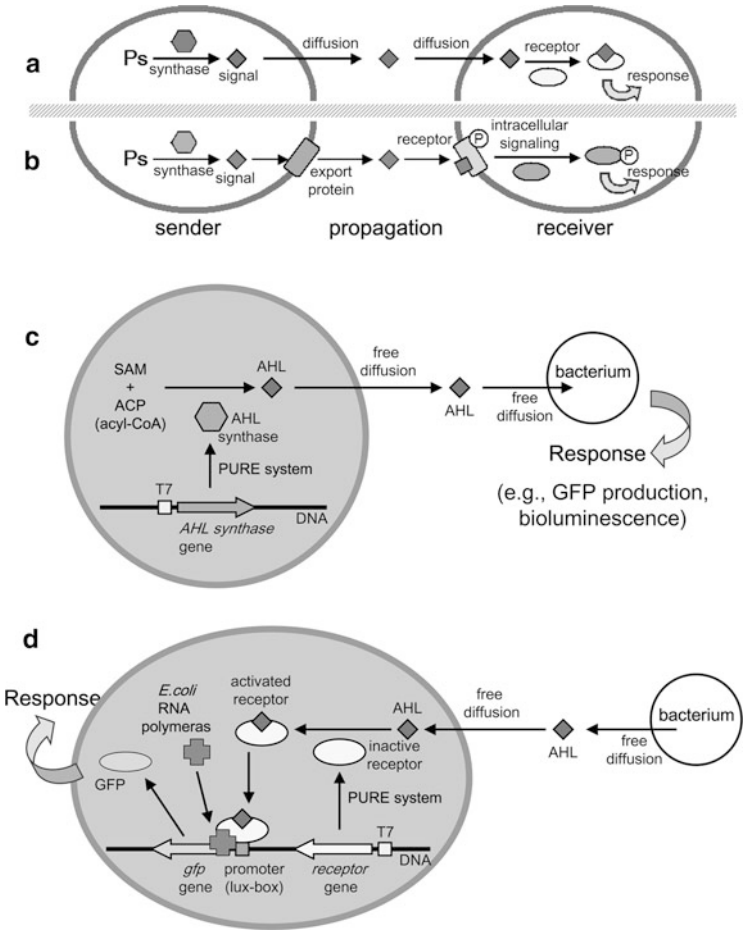

Fig. 2 Schematic representation of the communication mechanisms based on acylated homoserine lactones (AHLs) (**a**), and peptides or AI-2 (**b**); see details in the text. Possible experimental approaches (not yet realized) for the construction of semi-synthetic minimal cells that send (**c**) or receive (**d**) chemical signals from living cells (in this case, bacteria). See explanation in the text. In panels (**c**, **d**), the SSMCs are represented in *gray*

cytoplasm of the receiving cell and bind to an intracellular receptor (a LuxR-type protein). The AHLs/receptor complex binds to the promoter region of target genes, activating their expression [28].

Figure 2b illustrates the mechanisms for peptides- and AI-2-based communication. Nonfunctional pro-peptides are synthesized via ribosomal synthesis. In order to leave the sender cell, pro-peptides need a membrane-associated export protein, which also processes the pro-peptides to short functional linear or cyclic signal peptides. After the propagation, the signal peptides reach the surface of the receiver cell, where they bind to trans-membrane receptors. In many cases, the binding of

the signal induces auto-phosphorylation of the cytosolic portion of the receptor and consequent transfer of the phosphate group to a cytosolic transcriptional regulator, that in this activate form triggers a transcriptional response [28]. A similar mechanism holds also for AI-2-based communication. In this case the signal molecule is produced by the spontaneous cyclization of 4,5-dihydroxy-2,3-pentadione, which is synthesized from *S*-ribosyl-L-homocysteine by a LuxS-type enzyme. AI-2 is detected by the receiver cell via a trans-membrane receptor and the information transduced to the cytoplasm through a phosphorylation cascade, similarly to the peptides-based communication mechanism described above [28].

It is evident that the design of molecular communication between natural and synthetic cells should be designed, as first approach, in the simplest way. In particular, by comparing the mechanisms represented in Fig. 2, it is evident that the communication devices needed for AHLs-based systems are much simpler than the others (based on peptides and AI-2). Indeed, AHLs are produced by a single enzyme, they can freely diffuse outside the sending cell, are stable in the aqueous environment, can freely diffuse inside the receiver, and their decoding device is quite simple (intracellular receptor/regulator).

It is possible to conceive uni- or bi-directional communicating systems between SSMCs and natural cells. For example, "sending" SSMCs, or "receiving" SSMCs can be constructed, by inserting the appropriate molecular device for producing a signal molecule, or for its decoding, respectively. Bi-directional communications can also be implemented by endowing the SSMCs of both sending and receiving devices.

Figure 2c shows the blueprint for constructing SSMCs sending a chemical signal in form of a short-chain AHL. In this case the key function to be developed is the synthesis of AHL. This could be possible by co-entrapping inside a lipid vesicle, together with a transcription/translation machinery (i.e., the PURE system, or cell extracts), a gene encoding for a synthase of the LuxI or LuxM family. The direct precursor(s) of the signal molecule (*S*-adenosin-methionine and an appropriate acyl-CoA—instead of an acyl-carrier protein) could also be encapsulated inside liposomes together with all other compounds. These precursors are expected to be processed by the in situ produced synthase to give an AHL. Being freely diffusible, this signal molecule would reach the receiver cells, which are bacteria. These bacteria should be engineered in order to produce a reporter protein (e.g., GFP) only when the exogenous signal AHL, produced by the SSMCs, binds to their internal receptors.

Figure 2d shows the opposite case: SSMCs that decode a signal sent by bacteria. At this aim, short-chain AHLs are again good candidates as signal molecules. Here, together with a transcription/translation machinery, it should be introduced inside lipid vesicles the DNA encoding for a LuxR-type signal receptor and a reporter gene expressed under the control of this receptor (for instance, a *gfp* gene). The AHL produced by the bacteria are expected to freely cross the lipid membrane of SSMC and bind to the receptor. Once "activated" by AHL binding, the receptor would activate the expression of the reporter gene (*gfp*). In order to function in this

way, however, the presence of bacterial RNA polymerase is mandatory (note that the PURE system and most cell extracts contain the viral T7 RNA polymerase).

Bi-directional systems can also be engineered, in principle, by complementing the sending device with the receiving one (not shown), but it is clear that this should be considered a goal that comes after being capable of constructing and controlling each individual uni-directional systems. Possible interferences (cross-talks) between these devices should also be considered and solved.

In order to extend our possible approach based on AHL to other molecules, like the peptides and the furanones, two main obstacles must be overcome. The first one is the limited permeability of these compounds through the lipid membrane of SSMCs. We have seen that in living cells the secretion of these molecules requires a specific membrane-associated protein. The synthesis of membrane enzymes or transporters inside liposomes has been demonstrated only for two acyltransferases [23] and cannot be considered an easy task. This is due to the complex mechanism of correct membrane insertion and folding of a membrane protein—a process often helped in vivo by other specialized proteins. On the other hand, it is possible to modify the membrane integrity (and therefore enhance the solute permeability) in several ways. One of the most interesting one consists in the reconstruction of membrane pores. For example, it has been shown that α-hemolysin spontaneously form pores (cutoff: 3 kDa) when it is added to liposomes, or synthesized from within [24]. Thanks to the α-hemolysin, small molecules can move across the liposome membrane following their concentration gradient. Clearly, this "shortcut" for exporting a molecule of interest does not solve the issue of post-synthetic processing (needed when peptides are used as signal molecules).

The second obstacle is even more sophisticated and complex to solve, the decoding of peptide/AI-2 signals requiring trans-membrane molecular devices able to transduce the binding of the signal molecule to the receptor into a phosphorylation cascade process. These have not been reconstituted yet in liposomes and certainly represent a challenge for future research.

5 Conclusions and Perspectives

The development of chemical communication between synthetic and natural cells represents one of the next goals of SSMCs research, clearly oriented toward the application of SSMCs in synthetic biology and bio/chem-ICT. As emphasized by Suda and coworkers [1], a technology that can be interfaced with biological systems allows the establishment of a direct communication with cells—in their chemical language—and therefore paves the way to several perspectives for advanced applications, for example biomedical ones.

In this respect, it is interesting to cite here the vision provided by Leduc et al. [29], who proposed the concept of "pseudo-cell factories" or "nanofactories." These are liposome-based systems (actually a kind of SSMCs) designed for medical applications, namely for being administered to the human body with the aim of

targeting toward a specific tissue (by means of surface-bound antibodies, like in "immunoliposomes"). Arrived on their target site, nanofactories would be able to sense their microenvironment (input) and consequently to trigger an internal genetic/metabolic network that might produce a drug or any other chemical with biological effect (output). In other words, Leduc et al. implicitly gave to their nanofactories the capacity of communicating with natural cells. This is indeed the essence of the bio/chem-ICT vision we presented in this chapter. In more general terms, and in a future perspective, SSMCs, thanks to their modular construction, might host programmable regulatory genetic networks in order to respond to different signals. Remarkably, this is also a form of computing.

Acknowledgments This work derived from our recent involvement in studies on the construction of semi-synthetic minimal cells, funded by the FP6-EU Program (SYNTHCELLS: 043359), HFSP (RGP0033/2007–C), ASI (I/015/07/0), PRIN2008 (2008FY7RJ4); and further expanded thanks to networking initiatives as SynBioNT (UK), and the COST Systems Chemistry action (CM0703). Studies about quorum sensing were founded by the Italian Ministry of University and Research (PRIN-2008-232P4H_003 and FIRB-2010-RBFR10LHD1_002) and by the Italian Cystic Fibrosis Research Foundation (Projects FFC 14/2010 and FFC 13/2011).

References

1. Nakano, T., Moore, M., Enomoto, A., Suda, T.: Molecular communication technology as a biological ICT. In: Sawai, H. (ed.) Biological Functions for Information and Communication Technologies, Studies in Computational Intelligence, pp. 49–86. Springer, Heidelberg (2011)
2. Luisi, P.L., Ferri, F., Stano, P.: Approaches to semi-synthetic minimal cells: a review. Naturwissenschaften **93**, 1–13 (2006)
3. Stano, P., Rampioni, G., Carrara, P., Damiano, L., Leoni, L., Luisi, P.L.: Semi-synthetic minimal cells as a tool for biochemical ICT. Biosystems **109**, 24–34 (2012)
4. Maturana, H.R., Varela, F.J.: De Máquinas y Seres Vivos. Editorial Universitaria, Santiago (1973)
5. Maturana, H.R., Varela, F.J.: Autopoiesis: the organization of the living. In: Maturana, H.R., Varela, F.J. (eds.) Autopoiesis and Cognition, pp. 59–134. Reidel Publishing Company, Dordrecht (1980)
6. Maturana, H.R., Varela, F.J.: The Three of Knowledge. The Biological Roots of Human Understanding. Shimbhala, Boston (1987)
7. Cronin, L., Krasnogor, N., Davis, B.G., Alexander, C., Robertson, N., Steinke, J.H., Schroeder, S.L., Khlobystov, A.N., Cooper, G., Gardner, P.M., Siepmann, P., Whitaker, B.J., Marsh, D.: The imitation game – a computational chemical approach to recognizing life. Nat. Biotechnol. **24**, 1203–1206 (2006)
8. Gardner, P.M., Winzer, K., Davis, B.G.: Sugar synthesis in a protocellular model leads to a cell signalling response in bacteria. Nat. Chem. **1**, 377–383 (2009)
9. Morowitz, H.J., Heinz, B., Deamer, D.W.: The chemical logic of a minimum protocell. Orig. Life Evol. Biosph. **18**, 281–287 (1988)
10. Morowitz, H.J.: Beginnings of Cellular Life. Yale University Press, New Haven and London (1992)
11. Stano, P., Luisi, P.L.: Achievements and open questions in the self-reproduction of vesicles and synthetic minimal cells. Chem. Commun. **46**, 3639–3653 (2010)

12. Rasmussen, S., Bedau, M.A., Chen, L., Deamer, D., Krakauer, D.C., Packard, N.H., Stadler, P.F. (eds.): Protocells. Bridging Nonliving and Living Matter. The MIT Press, Cambridge (2009)
13. Forster, A.C., Church, G.M.: Towards synthesis of a minimal cell. Mol. Syst. Biol. **2**, 45 (2006)
14. Stano, P., Carrara, P., Kuruma, Y., Souza, T., Luisi, P.L.: Compartmentalized reactions as a case of soft-matter biotechnology: synthesis of proteins and nucleic acids inside lipid vesicles. J. Mater. Chem. **21**, 18887–18902 (2011)
15. Oberholzer, T., Nierhaus, K.H., Luisi, P.L.: Protein expression in liposomes. Biochem. Biophys. Res. Commun. **261**, 238–241 (1999)
16. Souza, T., Stano, P., Luisi, P.L.: The minimal size of liposome-based model cells brings about a remarkably enhanced entrapment and protein synthesis. Chembiochem **10**, 1056–1063 (2009)
17. Luisi, P.L., Allegretti, M., Souza, T., Steineger, F., Fahr, A., Stano, P.: Spontaneous protein crowding in liposomes: a new vista for the origin of cellular metabolism. Chembiochem **11**, 1989–1992 (2010)
18. Yu, W., Sato, K., Wakabayashi, M., Nakatshi, T., Ko-Mitamura, E.P., Shima, Y., Urabe, I., Yomo, T.: Synthesis of functional protein in liposome. J. Biosci. Bioeng. **92**, 590–593 (2001)
19. Hosoda, K., Sunami, T., Kazuta, Y., Matsuura, T., Suzuki, H., Yomo, T.: Quantitative study of the structure of multilamellar giant liposomes as a container of protein synthesis reaction. Langmuir **24**, 13540–13548 (2008)
20. Pautot, S., Frisken, B.J., Weitz, D.A.: Production of unilamellar vesicles using an inverted emulsion. Langmuir **19**, 2870–2879 (2003)
21. The, S.-Y., Khnouf, R., Fan, H., Lee, A.P.: Stable, biocompatible lipid vesicle generation by solvent extraction-based droplet microfluidics. Biomicrofluidics **5**, 044113 (2011)
22. Shimizu, Y., Inoue, A., Tomari, Y., Suzuki, T., Yokogawa, T., Nishikawa, K., Ueda, T.: Cell free translation reconstituted with purified components. Nat. Biotechnol. **19**, 751–755 (2001)
23. Kuruma, Y., Stano, P., Ueda, T., Luisi, P.L.: A synthetic biology approach to the construction of membrane proteins in semi-synthetic minimal cells. Biochim. Biophys. Acta **1788**, 567–574 (2009)
24. Noireaux, V., Libchaber, A.: A vesicle bioreactor as a step toward an artificial cell assembly. Proc. Natl. Acad. Sci. U. S. A. **101**, 17669–17674 (2004)
25. Parsek, M.R., Greenberg, E.P.: Sociomicrobiology: the connections between quorum sensing and biofilms. Trends Microbiol. **13**, 27–33 (2005)
26. West, S.A., Griffin, A.S., Gardner, A., Diggle, S.P.: Social evolution theory for microorganisms. Nat. Rev. Microbiol. **4**, 597–607 (2006)
27. Fuqua, W.C., Winans, S.C., Greenberg, E.P.: Quorum sensing in bacteria: the LuxR-LuxI family of cell density-responsive transcriptional regulators. J. Bacteriol. **176**, 269–275 (1994)
28. Atkinson, S., Williams, P.: Quorum sensing and social networking in the microbial world. J. R. Soc. Interface **6**, 959–978 (2009)
29. Leduc, P.R., Wong, M.S., Ferreira, P.M., Groff, R.E., Haslinger, K., Koonce, M.P., Lee, W.Y., Love, J.C., McCammon, J.A., Monteiro-Riviere, N.A., Rotello, V.M., Rubloff, G.W., Westervelt, R., Yoda, M.: Towards an in vivo biologically inspired nanofactory. Nat. Nanotechnol. **2**, 3–7 (2007)

Part III
Mind and Society

Cumulative Learning Through Intrinsic Reinforcements

Vieri G. Santucci, Gianluca Baldassarre, and Marco Mirolli

Abstract Building artificial agents able to autonomously learn new skills and to easily adapt in different and complex environments is an important goal for robotics and machine learning. We propose that providing reinforcement learning artificial agents with a learning signal that resembles the characteristic of the phasic activations of dopaminergic neurons would be an advancement in the development of more autonomous and versatile systems. In particular, we suggest that the particular composition of such a signal, determined by both extrinsic and intrinsic reinforcements, would be suitable to improve the implementation of cumulative learning in artificial agents. To validate our hypothesis we performed experiments with a simulated robotic system that has to learn different skills to obtain extrinsic rewards. We compare different versions of the system varying the composition of the learning signal and we show that the only system able to reach high performance in the task is the one that implements the learning signal suggested by our hypothesis.

V.G. Santucci (✉)
Istituto di Scienze e Tecnologie della Cognizione (ISTC), Consiglio Nazionale delle Ricerche (CNR), Laboratory of Computational Embodied Neuroscience (LOCEN)
Via San Martino della Battaglia 44, 00185, Roma, Italia

School of Computing and Mathematics, University of Plymouth
Plymouth PL4 8AA, UK
e-mail: vieri.santucci@istc.cnr.it

G. Baldassarre · M. Mirolli
Istituto di Scienze e Tecnologie della Cognizione (ISTC), Consiglio Nazionale delle Ricerche (CNR), Laboratory of Computational Embodied Neuroscience (LOCEN)
Via San Martino della Battaglia 44, 00185, Roma, Italia
e-mail: gianluca.baldassarre@istc.cnr.it; marco.mirolli@istc.cnr.it

S. Cagnoni et al. (eds.), *Evolution, Complexity and Artificial Life*,
DOI 10.1007/978-3-642-37577-4_7,

1 Introduction

Building artificial agents able to autonomously form ample repertoires of actions and to easily adapt in different and complex environments is an important goal for robotics and machine learning. One of the features that allows agents to achieve autonomous development and high versatility [1] is *cumulative learning*, i.e. the ability to use previously acquired skills to learn new ones and to combine sequences of actions to interact in different and more complex ways with the environment. Implementing cumulative learning in artificial agents presents many difficulties: two of the main and more general problems [2] are (a) the generation of the learning signal that can drive cumulative learning and (b) the type of architecture that can support such a process. In this work we focused on problem (a), trying to suggest a novel way to solve it.

In the computational literature one of the solutions to the problem of cumulative learning has been to replace task-specific learning signals with new non-task-specific learning signals inspired by what psychologists have been calling *intrinsic motivations* (IMs) [3–5]. IMs were introduced in the 1950s in animal psychology to explain experimental data (e.g. [6, 7]), incompatible with the classic motivational theory (e.g. [8]), showing that stimuli not related to (extrinsic) primary drives present a reinforcing value capable of conditioning instrumental responses [9–11]. Some authors focused on learning signals determined by the acquisition of knowledge by the system (e.g. [12–15], while other authors used learning signals based on what the system is doing, and in particular on the acquisition of new competences (e.g. [16, 17]). Although with different solutions, the intrinsically motivated approach influenced many works (for a review, see [18]) focused on the development of more versatile and autonomous systems able to acquire repertoires of skills, possibly in a cumulative fashion [19].

Our idea (first presented in a preliminary version in [20]) is that if we want to solve the problem of which learning signal can be suitable for the implementation of cumulative learning, a good solution is to look at biological organisms: the characteristics that we are trying to implement in artificial systems are typical of biological agents, that are able to cumulatively (and autonomously) learn new skills and to combine them together to optimise their survival chances. What we suggest is to look at those data that can explain how these features are developed in biology, focusing on those signals that can support cumulative learning.

The neuromodulator dopamine (DA) has long been recognized to play a fundamental role in motivational control and reinforcement learning processes [21–23]. In particular, phasic DA activations have been related not only to the presentation of unpredicted rewards [24–27] but also to other phasic, non-reward-related, unexpected stimuli [28–31]. These data led to the formulation of two main hypotheses on the functional role of the DA signal. One hypothesis [32, 33] looks at the similarities of DA activations with the temporal-difference (TD) error of computational reinforcement learning [34], and suggests that phasic DA represents a *reward prediction error* signal with the role of guiding the maximisation of future

rewards through the selection of the appropriate actions. The second hypothesis [35, 36] focuses on the activations for unexpected events and states that phasic DA is a *sensory prediction error* signal with the function of guiding the discovery and acquisition of novel actions.

As we pointed out in another work [37], we consider these two hypotheses both partially true, but at the same time not capable of taking into account all the empirical evidence on phasic DA activations. What we proposed in that work is that phasic DA represents a reinforcement prediction error signal analogous to the computational TD-error, but for a learning system that receives two different kinds of reinforcements: (1) temporary reinforcements provided by unexpected events and (2) permanent reinforcements provided by biological rewards. In our hypothesis, the DA signal has the function of driving both the formation of a repertoire of actions and the maximisation of biological rewards through the deployment of the acquired skills. Moreover, we suggest that phasic DA activations determined by unexpected events may constitute part of the neural substrate of IM: unpredicted events are intrinsic reinforcers that drive the same reinforcement learning processes as extrinsic reinforcers.

In this work we propose that providing artificial agents with a learning signal that resembles the characteristic of the phasic DA signal, determined by both extrinsic and intrinsic reinforcements, would be an advancement in the development of more autonomous and versatile systems. Moving from biology to artificial agents, we can identify extrinsic reinforcements with those determined by the achievement of the tasks decided by the researchers, whereas intrinsic reinforcements are identified with those determined by a category of more general events, such as the unexpected activations of the sensors of the robot, determined by its interactions with the environment. Similarly to what happens in biological systems [38], we believe that intrinsic reinforcements can play a key role in determining a proper signal for the implementation of the cumulative learning of skills and for the acquisition of complex behaviours that would not be learnt simply with extrinsic reinforcements.

To test our hypothesis, we built a simulated robotic system that has to autonomously acquire a series of skills in order to maximise its rewards (Sect. 2). We compare the performance of the system with different compositions of the learning signal and we show (Sect. 3) that the system implementing our hypothesis is the only one that is able to learn the task. We then draw the conclusions (Sect. 4) by analysing the results of the experiments and discussing the implications of our hypothesis.

2 Set-Up

2.1 The Task and the Simulated Robot

The system is a simulated kinematic robot composed of a fixed head with a “mouth”, a moving eye, and a two-degree-of-freedom kinematic arm with a hand that can “grasp” objects. The task consists in learning to eat food (i.e., bring a red object

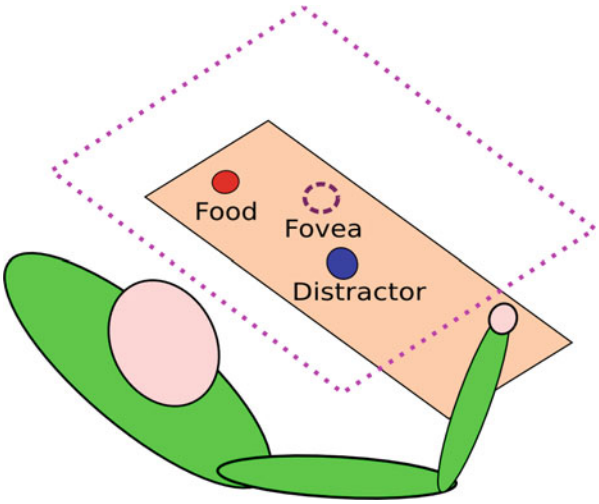

Fig. 1 Set-up of the experiment: the system composed by a two-dimensional arm and a moving eye (*dotted square with a fovea at the centre*). The food and a fixed distractor are positioned on a table in front of the robot. The task consists in eating the food by bringing it to the mouth. See text for details

to the mouth) randomly placed on a rectangular table (with dimensions of 4 and 7 units, respectively) set in front of the robot (Fig. 1). In the middle of the table we add a visual "distractor" of a different colour (blue) that can only be foveated while, for simplicity, it cannot be touched or grasped: interacting with this second object does not increase the chance for the system to achieve the final goal.

In real environments the organisms are surrounded by many different objects with which they can interact in many different ways. However, not every interaction has the same importance: some actions could turn out to be the basis for more complex ones, while others may even result useless. Since we want to improve the versatility of artificial agents, we want to test our hypothesis in an environment that presents, although much simplified, some of the characteristics of the real world. For this reason we put a "distractor" that has no relations with the task in order to provide a set-up where not all the possible interactions with the environment are related to the main task of the experiment.

Since we are focusing on cumulative learning, there is a dependency between the skills that the robot can learn: the arm receives as input what the eye sees, so that learning to systematically look at the food is a prerequisite for learning to reach for it; at the same time, reaching for the food is necessary for grasping and bringing it to the mouth.

The sensory system of the robot is composed of: (a) an artificial retina (a square of 14 units per size; note that this implies that at the beginning of each trial the whole table is always within the eye image) sensible to the two different colours of the objects, encoding the position of the hand, of the food (a circle with 0.3 units diameter) and of the distractor (diameter 0.4) with respect to the centre of the

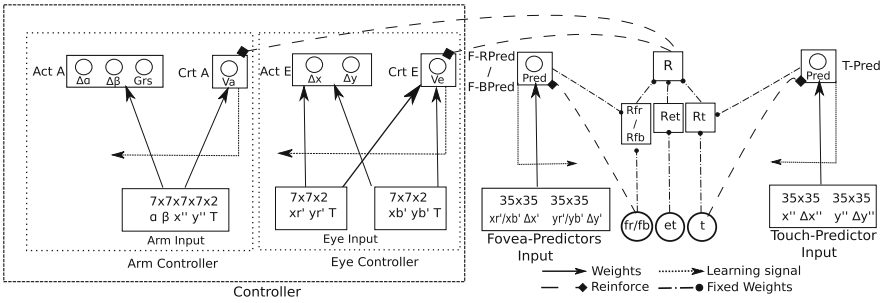

Fig. 2 The controller formed by two components (arm and eye controllers), the two predictors of the fovea sensor (for simplicity, in this schema they are presented as a single structure), the predictor of the touch sensor, and the reinforcement system. α and β are the angles of the two arm joints; x'' and y'' are the hand positions with respect to the fovea on the x and y axes; $\Delta\alpha$ and $\Delta\beta$ are the variations of angles as determined by the arms actor; Grs is the grasping output; Va is the evaluation of the critic of the arm; xr', yr' and xb', yb' are the positions of food and distractor with respect to the fovea on the x and y axes; Δx and Δy are the displacements of the eye determined by the actor of the eye; Ve is the evaluation of the critic of the eye; F-RPred and F-BPred are the predictions of the fovea-predictors; T-Pred is the prediction of the touch-predictor; fr and fb are the activations of the fovea sensor for the two colours; t is the activation of the touch sensor; Rfr, Rfb and Rt are the reinforcements related to sensors activations; Ret is the reinforcement provided by eating the food; R is the total reinforcement. See text for details. Figure modified from [37], copyright (2013), with permission from Elsevier

visual field; (b) a "fovea", encoding whether the food or the distractor is perceived in the centre of the visual field; (c) the proprioception of the arm (composed of two segments of 4 units), encoding the angles of the two arm joints; (d) a touch sensor encoding whether the hand is in contact with the food (i.e., if the hand and the object are overlapping: for simplicity collisions are not simulated). The eye moves along the x and y axes with a maximum step of 8 units. The two joints of the arm move within the interval [0, 180] degrees, with a maximum step of 25°.

2.2 *Architecture and Experimental Conditions*

As we are proposing to look at biological organisms to improve the implementation of cumulative learning in artificial agents, we tried to build the architecture of the system (Fig. 2) following some constraints deriving from the known biology underlying reinforcement learning in real animals. The controller of the system reflects the modular organization of the basal-ganglia-thalamo-cortical loops [39], where the acquisition of new motor skills and the selection of motor commands take place [40]. We implemented the system as an actor-critic reinforcement learning architecture based on TD-learning because there is evidence [41] that the dorsal regions of the basal ganglia reflect the characteristics of this structure and because this solution has also some appealing theoretical properties from the machine learning point of view [34,42]. Moreover, the reinforcement learning signal is unique for

both the sub-controllers because the phasic DA signal is likely to be the same for all sensory-motor subsystems [43]: this simplifies the computation of the learning signal and allows to reinforce some actions also if they determine the activations of sensors not directly connected to the effectors that generated those effects.

As described in Sect. 1, the reinforcement signal is determined by both the extrinsic rewards provided by eating the food and by the intrinsic reinforcements provided by the unpredicted activations of the fovea and the touch sensors. To implement the intrinsic reinforcements, the system includes also three predictors, two for the fovea sensor (one for each colour of the objects) and one for the touch sensor. Each predictor is trained to predict the activation of the corresponding sensor and inhibits the part of the intrinsic reinforcement that depends on the unexpected activation of that sensor. Hence, the total reinforcement (R) driving TD-learning is:

$$R = R_e + R_{ff} + R_{fd} + R_t$$

where R_e is the extrinsic reinforcement provided by bringing the food to the mouth (with a value of 15), while R_{ff}, R_{fd} and R_t are the intrinsic reinforcements provided by the unpredicted activations of the fovea sensor caused by the food (R_{ff}) or by the "distractor" (R_{fd}) and the unpredicted activations of the touch sensor (R_t) caused by the food. In particular, for a generic sensor S, the reinforcement R_S provided by the activation of S is:

$$R_S = \max[0; A_S - P_S]$$

where A_S is the binary activation $\{0; 1\}$ of sensor S and P_S is the prediction generated by the predictor of sensor S. In this way we use only the positive reinforcements generated when the activation of A_S is not fully predicted by P_S.

To test our hypothesis, we compare the described condition (called *intrinsic* condition), with two different conditions, where we vary the composition of the learning signal. In the *extrinsic* condition the reinforcement is given only by the extrinsic reinforcements provided by eating the food (R_e). This condition is useful to test if extrinsic reinforcements by themselves are able to drive the cumulative learning of skills. In the *sub-tasks* condition, the additional reinforcements provided by the activations of the sensors (R_{ff}, R_{fd} and R_t) are also "permanent", in the sense that they are not modulated by the activity of the predictors and hence do not change throughout training. With this condition we want to test if the temporary nature of intrinsic reinforcements is necessary to facilitate learning.

2.3 Input Coding

All the inputs are encoded with population coding [44] through Gaussian radial basis functions (RBF) [45]:

$$a_i = e^{-\sum_d \left(\frac{c_d - c_{id}}{2\sigma_d^2}\right)^2}$$

where a_i is the activation of unit i, c_d is the input value on dimension d, c_{id} is the preferred value of unit i with respect to dimension d, and σ_d^2 is the width of the Gaussian along dimension d (widths are parametrised so that when the input is equidistant, along a given dimension, to two contiguous neurons, their activation is 0.5).

The dimensions of the input to the two "retinas" of the eye controller are the position of the respective object (in x and y) with respect to the centre of the visual field and the activation of the touch sensor. We add the status of the touch sensor because for computational limits the eye is not able to follow the food when it is moved by the hand: providing this information we can separate the two situations (object not grasped from object grasped) and prevent the controller of the eye from losing the ability of looking at the objects. The preferred object positions of input units are uniformly distributed on a 7×7 grid with ranges $[-7; 7]$, which, multiplied by the binary activation of the touch sensor, form a total $7 \times 7 \times 2$ grid. In total, the eye has an input formed by two $7 \times 7 \times 2$ grids, one for each of the two objects.

The dimensions of the input to the arm controller are the angles of the two joints (α and β), the position of the hand (x and y) with respect to the fovea, and the activation of the touch sensor. The preferred joint angles of input units are uniformly distributed on two dimensions (7×7) ranging in [0; 180] whereas the preferred positions of the hand with respect to the fovea are uniformly distributed on other two dimensions (7×7) with ranges $[-7; 7]$. Hence, considering the binary activation of the touch sensor, the input is formed by a total $7 \times 7 \times 7 \times 7 \times 2$ grid.

The input units of the eye controller are fully connected to two output units with sigmoidal activation:

$$o_j = \Phi(\sum_i^M a_i w_{ji} + b_j) \qquad \Phi(x) = \frac{1}{1 + e^{-x}}$$

where M is the total number of input units, w_{ji} is the weight of the connection linking input unit i to output unit j and b_j is the bias of output unit j. Each actual motor command o_j^n is generated by adding some noise to the activation of the relative output unit:

$$o_j^n = o_j + n$$

where n is a random value uniformly drawn in $[-0.02; 0.02]$. The resulting commands (in [0; 1]) are remapped in $[-8, 8]$ and control the displacement of the eye along the two dimensions.

The arm controller has three output units. Two have sigmoidal activation, as those of the eye, with noise uniformly distributed in $[-0.2; 0.2]$. Each resulting motor command, remapped in $[-25; 25]$ degrees, determines the change of one joint angle. The third output unit has binary activation $\{0; 1\}$ and controls the grasping action (the activation is determined by the sigmoidal activation of the output unit plus a random noise uniformly drawn in $[-0.2; 0.2]$, with a threshold set to 0.5). The

activation of the grasping output is slightly punished with a negative reinforcement of 0.0001 to avoid that the system performs grasping also when it is not on the target.

The evaluation of the critic of each sub-controller k (V_k) is a linear combination of the weighted sum of the respective input units.

The input units of the predictors of fovea activation are formed by two 35×35 grids, each one encoding the position of the respective object with respect to the fovea along one axis and the programmed displacement of the eye along the same axis. Similarly, the input of the predictor of the touch sensor is formed by two 35×35 grids, each one encoding the position of the hand with respect to the food along one axis and the programmed displacement of the hand along the same axis. Preferred inputs are uniformly distributed in the range $[-7; 7]$ for objects positions and $[-25; 25]$ for displacements. The output of each predictor is a single sigmoidal unit receiving connections from all the units of the predictor.

2.4 Learning

Learning depends on the TD reinforcement learning algorithm [34] that was introduced to solve the temporal credit assignment problem, i.e. the problem of learning which of many actions contributed to the achievement of reward: the TD learning solves the problem with the use of predictions and in particular with the use of the TD-error as the learning signal reinforcing all those actions that lead the system closer to rewards. The TD-error δ_k of each sub-controller k is computed as:

$$\delta_k = (R^t + \gamma_k V_k^t) - V_k^{t-1}$$

where R^t is the reinforcement at time step t, V_k^t is the evaluation of the critic of controller k at time step t, and γ_k is the discount factor, set to 0.9 for both the eye and the arm controllers.

The weight w_{ki} of input unit i of critic k is updated in the standard way:

$$\Delta w_{ki} = \eta_k^c \delta_k a_i$$

where η_k^c is the learning rate, set to 0.02 for both the eye and the arm controllers.

The weights of actor k are updated as follows:

$$\Delta w_{kji} = \eta_k^a \delta_k (o_{kj}^n - o_{kj})(o_{kj}(1 - o_{kj}))a_{ki}$$

where η_k^a is the learning rate (set to 0.2 for both the eye and the arm controller), and $o_{kj}(1 - o_{kj})$ is the derivative of the sigmoid function.

Predictors are trained through a TD-learning algorithm (for a generalization of TD-learning to general predictions, see [46]). We decided to use TD-learning neural networks to implement the predictors because it is difficult to built predictors able

to perfectly anticipate the activations of the sensors: a TD neural network solves the problem because it starts to anticipate the activations earlier than a one-step predictor.

For each predictor p, the TD-error δ_p is calculated as follows:

$$\delta_p = (A_p^t + \gamma_p O_p^t) - O_p^{t-1}$$

where A_p^t is the activation of the sensor related to predictor p at time step t, O_p^t is the output of predictor p at time step t, and γ_p is the discount factor, set to 0.7 for each predictor. Finally, the weights of the predictors are updated as those of the critics of the two sub-controllers, with a learning rate set to 0.00008 for each predictor.

3 Results

We tested each condition on the experimental task for 500,000 trials, each trial terminating when food was eaten or when it "fell off" the table (i.e., if the food is positioned outside the table and not "grasped"), or after a time out of 40 steps. At the end of every trial the food, the eye centre and the hand were repositioned randomly without overlaps, with the first two always inside the table. Every 500 trials we performed 50 test trials (where learning was switched off). For each condition we ran ten replications of the experiment and here we present the average results of those replications.

Figure 3 shows the performance in the task of the three experimental conditions. In the *extrinsic* condition the robot is not able to learn to eat reliably. Adding permanent reinforcements for every possible interaction with the environment, as in the *sub-tasks* condition, does not improve the performance of the system in the final task. Differently, in the *intrinsic* condition, where the activations of the sensors are reinforcing only when unpredicted, the system is able to reach high performance in the eating task (about 85 %).

It is quite easy to understand why in the *extrinsic* condition the system is not able to achieve the final goal: the only reinforcement provided by the final reward is too distant and infrequent to drive the learning of the sub-tasks needed for bringing the food into the mouth. Although the TD algorythm is built to solve the credit assignment problem, it is difficult to trace back few rewards provided by a complex sequence of different actions.

It is more interesting to analyse the results of the other two conditions where further reinforcements are given in addition to the final one. To understand the reason of these results we have to look at the behaviour of the eye. In the *sub-tasks* condition (Fig. 4), the robot starts to look at the distractor, which is simpler to find within the table. Because of the permanent reinforcements provided by the activation of the fovea sensor the system is stuck on this activity, but looking

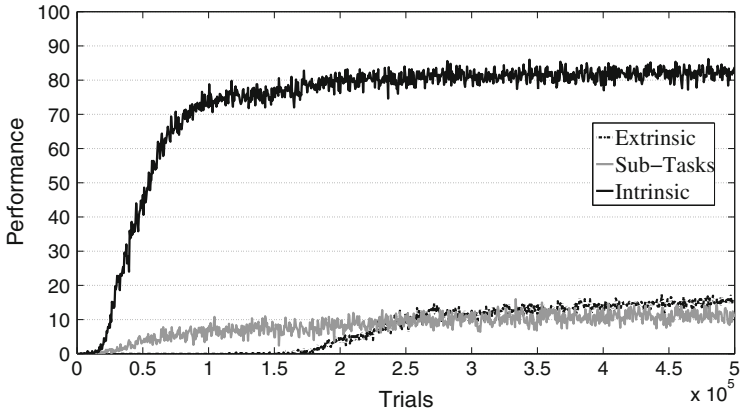

Fig. 3 Performance (percentage of test trials in which the robot eats the food) in the three experimental conditions. Figure reprinted from [37], copyright (2013), with permission from Elsevier

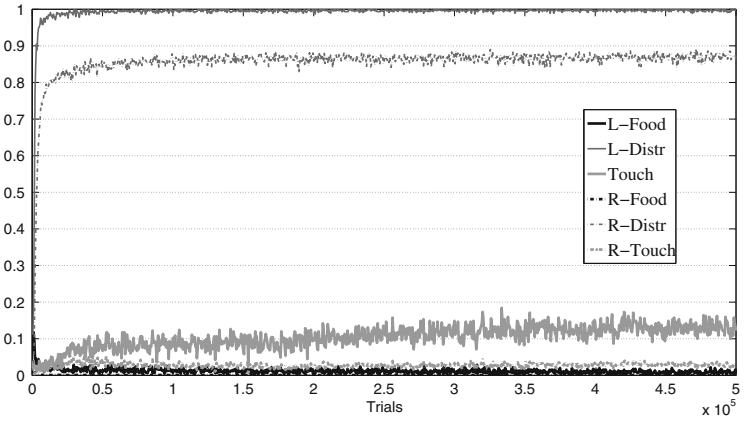

Fig. 4 Behaviour of the eye and of the arm in the *sub-tasks* condition. Average percentage of test trials in which the eye foveates the food (*L-Food*) and the distractor (*L-Distr*) and in which the hand touches the food (*Touch*); average reinforcements per step generated by the unpredicted activations of the sensors (*R-Food*, *R-Distr* and *R-Touch*). Figure modified from [37], copyright (2013), with permission from Elsevier

at the distractor is not related to the other skills so the agent is not able to develop the capacity to look at the food, which is a prerequisite for the other abilities (reaching and grasping the food) and for the achievement of the final goal.

On the contrary, in the *intrinsic* condition (Fig. 5) the robot is able to learn the correct sequence of actions. Also in this case the system starts with looking at the fixed target, but after the predictor of the fovea sensor for the blue colour starts to predict the perception of the distractor, that sensory event is no more reinforcing.

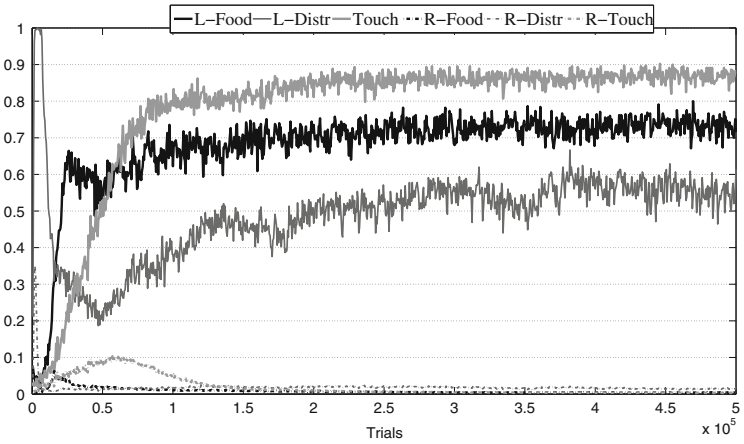

Fig. 5 Behaviour of the eye and of the arm in the *intrinsic* condition. Same data as in Fig. 4. Figure modified from [37], copyright (2013), with permission from Elsevier

As a result, the robot can discover that also foveating the food can be reinforcing and so starts acquiring this second ability, that is the prerequisite for the arm to learn to touch and eventually grasp the food and then to bring it to the mouth.

In the *intrinsic* condition the activations of the sensors determined by the interactions with the objects are reinforcing only when they are unexpected. If we look at Fig. 5, we can see that the reinforcements provided by the fovea and the touch sensors are not continuous as in the *sub-tasks* condition: they rapidly grow when the related ability is encountered and repeated, and they fade away when the motor skills are learned and their consequences become predictable. Although those skills do not directly generate more reinforcements, they are still performed when they constitute the prerequisites for successive actions that can provide new reinforcements and for the maximization of extrinsic rewards.

Note (Fig. 5) that as the robot learns to eat the food, the number of times it looks at the distractor increases again. Due to architectural limits, the eye is not able to track the food while the hand is moving it (the eye controller is not informed about the movements of the arm). As a result, the eye resorts to the behaviour that it has previously learnt, i.e. foveating the distractor. Moreover, the performance of the arm in touching the food is higher than the one of the eye in looking at it: when skills are learnt it is sufficient that the eye looks close to food to allow the arm to reach it.

We wondered if the results of the experiments are dependent on the values that we assigned to the different reinforcements: to verify this possibility, we tested the three conditions varying the value assigned to eating the food. The results (Fig. 6) show that changing the value of the extrinsic reward in the learning signal does not modify the comparison between the different conditions: lowering or rising the reward for eating the food maintains the *intrinsic* condition as the best performer.

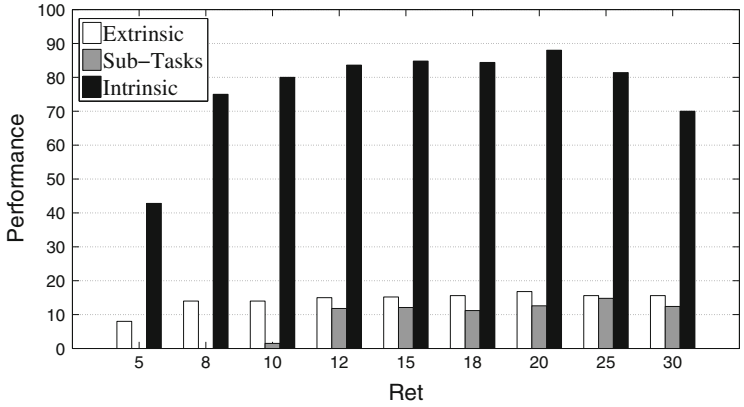

Fig. 6 Average final performance of the three conditions as a function of the value of the extrinsic reinforcement (Re) provided by eating the food. See text for details. Figure reprinted from [37], copyright (2013), with permission from Elsevier

4 Discussion

This paper validates our hypothesis that implementing artificial agents with a learning signal that resembles the phasic activations of DA neurons of biological organisms can support cumulative learning. We tested a simulated robotic agent in a simulated environment where not all the possible interactions with the world are useful for the achievement of the final goal. We varied the composition of the learning signal and we verified that only the one implementing our hypothesis was able to guide the simulated robot in the achievement of the task.

Extrinsic reinforcements by themselves are not sufficient to drive the acquisition of complex sequences of actions. Simply adding a further reinforcement for every interaction with the environment will lead the agents to get stuck in useless activities. Differently, a learning signal based both on the temporary reinforcements provided by unexpected events and by the permanent reinforcements of extrinsic rewards is able to guide the discovery of novel actions and the deployment of the acquired skills for the achievement of goals.

The nature of IMs fits particularly well with the complexity of real environments and cumulative learning. Intrinsic reinforcements are present only when they are needed: when the system discovers a new possible way to interact with the environment, the consequences of its actions provide high reinforcement; once the system has learnt to systematically generate an effect (after some repetitions of the same actions), that effect can be predicted and for this reason it is no more reinforcing; the system then is not stuck on the repetition of the same actions and can move to different activities. In this way intrinsic reinforcements are able to guide agents in the discovery of novel interactions with the environment, increasing their

repertoire of skills. Moreover, such a learning signal can be useful to develop more autonomous agents: IMs are able to push systems to learn every possible interaction with the environment just because of the novelty of those interactions, also if those new skills are not immediately related to the fitness of the system [38, 47]. These skills can then be deployed in the appropriate situations exploiting the reinforcing value of extrinsic reinforcements.

Looking at the implementation of our hypothesis, the system still has some limits. Schmidhuber [12] underlined how using the prediction error as an intrinsic reinforcement can generate problems if the environment is unpredictable or the system has limited learning capabilities: in such cases, the reinforcement would never decrease and the system would get stuck, trying to reproduce outcomes with unpredictable consequences. To avoid this problem, he proposed the progress in predictions error as a better intrinsic reinforcement. However, we believe that this hypothesis does not reflect the biology underlying IMs and we built our system using the simple prediction error to implement intrinsic reinforcements.

Another limit is connected to the second problem related to the implementation of cumulative learning (that we decided not to tackle in this work), the architectural problem: building a complex repertoire of actions needs an architecture that is able to discover and retain different abilities. In fact, another problem related to cumulative learning is *catastrophic forgetting*, the phenomenon by which neural networks forget past experiences when exposed to new ones. A good solution to this problem is to develop hierarchical architectures (e.g. [48, 49]. See [16] for a review) that are able to store new skills without impairing the old ones. We designed our system in order to bypass some of the problems related to catastrophic forgetting, but we will certainly need to move towards hierarchical structures in order to fully support cumulative learning processes. Moreover, we believe that within the framework of hierarchical organization of actions, we can provide a reinforcement signal that, without losing the inspiration provided by biological organisms, can cope with the problem raised by Schmidhuber: intrinsic reinforcements can be determined by the learning progress in skills acquisition [50]. If nothing can be learnt, there will be no learning progress and the system will move away looking for new skills to acquire.

Acknowledgements This research was supported by the European Community 7th Framework Programme (FP7/2007-2013), "Challenge 2 - Cognitive Systems, Interaction, Robotics", grant agreement No. ICT-IP-231722, project "IM-CLeVeR - Intrinsically Motivated Cumulative Learning Versatile Robots".

References

1. Weng, J., McClelland, J., Pentland, A., Sporns, O., Stockman, I., Sur, M., Thelen, E.: Artificial intelligence. Autonomous mental development by robots and animals. Science **291**(5504), 599–600 (2001)

2. Baldassarre, G., Mirolli, M.: What are the key open challenges for understanding autonomous cumulative learning of skills? Auton. Mental Develop. Newsl. **7**(2), 2–9 (2010)
3. White, R.: Motivation reconsidered: The concept of competence. Psychol. Rev. **66**, 297–333 (1959)
4. Berlyne, D.: Conflict, Arousal and Curiosity. McGraw Hill, New York (1960)
5. Ryan, R.M., Deci, E.L.: Intrinsic and extrinsic motivations: Classic definitions and new directions. Contemp. Educ. Psychol. **25**(1), 54–67 (2000)
6. Montgomery, K.: The role of the exploratory drive in learning. J. Comp. Psychol. **47**(1), 60–64 (1954)
7. Butler, R.A., Harlow, H.F.: Discrimination learning and learning sets to visual exploration incentives. J. Gen. Psychol. **57**(2), 257–264 (1957)
8. Hull, C.L.: Principles of behavior. Appleton-century-crofts, New York (1943)
9. Kish, G.B.: Learning when the onset of illumination is used as reinforcing stimulus. J. Comp. Physiol. Psychol. **48**(4), 261–264 (1955)
10. Glow, P., Winefield, A.: Response-contingent sensory change in a causally structured environment. Learn. Behav. **6**, 1–18 (1978)
11. Reed, P., Mitchell, C., Nokes, T.: Intrinsic reinforcing properties of putatively neutral stimuli in an instrumental two-lever discrimination task. Anim. Lear. Behav. **24**, 38–45 (1996)
12. Schmidhuber, J.: Curious model-building control system. In: Proceedings of International Joint Conference on Neural Networks, vol. 2, pp. 1458–1463. IEEE, Singapore (1991)
13. Huang, X., Weng, J.: Novelty and reinforcement learning in the value system of developmental robots. In: Prince, C., Demiris, Y., Marom, Y., Kozima, H., Balkenius, C. (eds.) Proceedings of the Second International Workshop on Epigenetic Robotics, vol. 94, pp. 47–55. Lund University (2002)
14. Oudeyer, P., Kaplan, F., Hafner, V.: Intrinsic motivation system for autonomous mental development. IEEE T. Evolut. Comput. **11**, 703–713 (2007)
15. Baranes, A., Oudeyer, P.Y.: Intrinsically motivated goal exloration for active motor learning in robots: A case study. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan (2010)
16. Barto, A., Singh, S., Chantanez, N.: Intrinsically motivated learning of hierarchical collections of skills. In: Proceedings of the Third International Conference on Developmental Learning (ICDL), pp. 112–119 (2004)
17. Schembri, M., Mirolli, M., Baldassarre, G.: Evolving internal reinforcers for an intrinsically motivated reinforcement-learning robot. In: Demiris, Y., Mareschal, D., Scassellati, B., Weng, J. (eds.) Proceedings of the 6th International Conference on Development and Learning, pp. E1–6. Imperial College, London (2007)
18. Oudeyer, P.Y., Kaplan, F.: What is intrinsic motivation? A typology of computational approaches. Front. Neurorobot. **1**(1), 1–14 (2007)
19. Baldassarre, G., Mirolli, M. (eds.): Intrinsically Motivated Learning in Natural and Artificial Systems. Springer, Berlin (2013)
20. Santucci, V.G., Baldassarre, G., Mirolli, M.: Biological cumulative learning through intrinsic motivation: A simulated robotic study on the development of visually-guided reaching. In: Johansson, B., Sahin, E., Balkenius, C. (eds.) Proceedings of the Tenth International Conference on Epigenetic Robotics, pp. 121–128. Lund University Cognitive Studies, Lund (2010)
21. Wise, R.: Dopamine, learning and motivation. Nat. Rev. Neurosci. **5**(6), 483–494 (2004)
22. Schultz, W.: Behavioral theories and the neurophysiology of reward. Annu. Rev. Psychol. **57**, 87–115 (2006)
23. Berridge, K.: The debate over dopamine's role in reward: The case for incentive salience. Psychopharmacology. **191**(3), 391–431 (2007)
24. Romo, R., Schultz, W.: Dopamine neurons of the monkey midbrain: Contingencies of responses to active touch during self-initiated arm movements. J. Neurophysiol. **63**(3), 592–606 (1990)

25. Ljungberg, T., Apicella, P., Schultz, W.: Responses of monkey midbrain dopamine neurons during delayed alternation performance. Brain Res. **567**(2), 337–341 (1991)
26. Schultz, W., Apicella, P., Ljumberg, T.: Responses of monkey dopamine neurons to reaward and conditioned stimuli during successive steps of learning a delayed response task. J. Neurosci. **13**, 900–913 (1993)
27. Mirenowicz, J., Schultz, W.: Importance of unpredictability for reward responses in primate dopamine neurons. J. Neurophysiol. **72**(2), 1024–1027 (1994)
28. Ljungberg, T., Apicella, P., Schultz, W.: Responses of monkey dopamine neurons during learning of behavioral reactions. J. Neurophysiol. **67**(1), 145–163 (1992)
29. Schultz, W.: Predictive reward signal of dopamine neurons. J. Neurophysiol. **80**(1), 1–27 (1998)
30. Horvitz, J.C.: Mesolimbocortical and nigrostriatal dopamine responses to salient non-reward events. Neuroscience **96**(4), 651–656 (2000)
31. Dommett, E., Coizet, V., Blaha, C.D., Martindale, J., Lefebvre, V., Walton, N., Mayhew, J.E.W., Overton, P.G., Redgrave, P.: How visual stimuli activate dopaminergic neurons at short latency. Science **307**(5714), 1476–1479 (2005)
32. Houk, J., Adams, J., Barto, A.: A model of how the basal ganglia generate and use neural signals that predict reinforcement. In: Houk, J.C., Davis, J.L., Beiser, D.G. (eds.) Models of Information Processing in the Basal Ganglia, pp. 249–270. MIT Press, Cambridge (1995)
33. Schultz, W., Dayan, P., Montague, P.R.: A neural substrate of prediction and reward. Science **275**(5306), 1593–1599 (1997)
34. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
35. Redgrave, P., Gurney, K.: The short-latency dopamine signal: A role in discovering novel actions? Nat. Rev. Neurosci. **7**(12), 967–975 (2006)
36. Redgrave, P., Vautrelle, N., Reynolds, J.N.J.: Functional properties of the basal ganglia's re-entrant loop architecture: Selection and reinforcement. Neuroscience. **198**, 138–151 (2011)
37. Mirolli, M., Santucci, V.G., Baldassarre, G.: Phasic dopamine as a prediction error of intrinsic and extrinsic reinforcements driving both action acquisition and reward maximization: a simulated robotic study. Neural Networks **39**, 40–51 (2013)
38. Baldassarre, G.: What are intrinsic motivations? A biological perspective. In: Cangelosi, A., Triesch, J., Fasel, I., Rohlfing, K., Nori, F., Oudeyer, P.Y., Schlesinger, M., Nagai, Y. (eds.) Proceedings of the International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob-2011), pp. E1–8. IEEE, Piscataway (2011)
39. Romanelli, P., Esposito, V., Schaal, D.W., Heit, G.: Somatotopy in the basal ganglia: Experimental and clinical evidence for segregated sensorimotor channels. Brain Res. Rev. **48**(1), 112–128 (2005)
40. Graybiel, A.M.: The basal ganglia: Learning new tricks and loving it. Curr. Opin. Neurobiol. **15**(6), 638–644 (2005)
41. Joel, D., Niv, Y., Ruppin, E.: Actor-critic models of the basal ganglia: New anatomical and computational perspectives. Neural Network **15**(4–6), 535–547 (2002)
42. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: NIPS, vol. 99, pp. 1057–1063 (1999)
43. Schultz, W.: Getting formal with dopamine and reward. Neuron **36**(2), 241–263 (2002)
44. Pouget, A., Snyder, L.H.: Computational approaches to sensorimotor transformations. Nat. Neurosci. **3**(Suppl), 1192–1198 (2000)
45. Buhmann, M.: Radial Basis Functions. Cambridge University Press, New York (2003)
46. Sutton, R., Tanner, B.: Temporal-difference networks. Adv. Neural Inf. Process. Syst. **17**, 1377–1348 (2005)
47. Singh, S., Lewis, R., Barto, A., Sorg, J.: Intrinsically motivated reinforcement learning: An evolutionary perspective. IEEE T. Auton. Mental Dev. **2**(2), 70–82 (2010)
48. Doya, K., Samejima, K., Katagiri, K., Kawato, M.: Multiple model-based reinforcement learning. Neural Comput. **14**(6), 1347–1369 (2002)

49. Caligiore, D., Mirolli, M., Parisi, D., Baldassarre, G.: A bioinspired hierarchical reinforcement learning architecture for modeling learning of multiple skills with continuous states and actions. In: Johansson, B., Sahin, E., Balkenius, C. (eds.) Proceedings of the Tenth International Conference on Epigenetic Robotics, pp. 27–34 (2010)
50. Schembri, M., Mirolli, M., Baldassarre, G.: Evolving childhood's length and learning parameters in an intrinsically motivated reinforcement learning robot. In: Berthouze, L., Dhristiopher, G., Littman, M., Kozima, H., Balkenius, C. (eds.) Proceedings of the Seventh International Conference on Epigenetic Robotics, pp. 141–148. Lund University Cognitive Studies, Lund (2007)

Development of Categorisation Abilities in Evolving Embodied Agents: A Study of Internal Representations with External Social Inputs

Francesco Pugliese

Abstract This paper investigates the behaviour of embodied and situated agents, which perform tasks requiring categorisation skills. These agents are simulated robots selected by an artificial adaptation process. Their task is to categorise objects with different shapes. To achieve this goal, the robots can use sensory information from the environment and external "linguistic" inputs. The results show that the agents are able to solve the categorisation task by conveniently integrating the experienced sensory-motor states and linguistic inputs. The aim of this work is to demonstrate that autonomous agents are able to develop some high-level cognitive abilities. Interestingly, the behavioural pattern seems to be in agreement with the theoretical hypothesis "social" information (external inputs) facilitates individual capacity to categorise, by producing good internal representations.

1 Introduction

Categorisation is the ability to discriminate different environmental situations (and different states of interaction between the agent and the environment) by producing different behaviours [1]. The way the categorisation process may occur depends on the interaction between the agent control system, its body and the environment where it is situated. The categorisation may emerge by actively exploiting sensory-motor experience and by recording it in some internal states of the agent control system [2].

Generally, the categorisation process is performed individually by the agents but, in some cases, it may be obtained in a social context through some form of

F. Pugliese (✉)
Natural and Artificial Cognition Laboratory, University of Naples, Via Porta di Massa 1, Naples, Italy
e-mail: francesco.pugliese@unina.it

S. Cagnoni et al. (eds.), *Evolution, Complexity and Artificial Life*,
DOI 10.1007/978-3-642-37577-4_8,

communication that can improve the categorisation process itself, by providing the agent with further information.

Two kinds of categorisation can be distinguished:

(a) *Active categorisation*, which is the ability of an agent to categorise using actions that affect its own sensory state. For example, a robot with a camera equipped with zoom, mounted on a moving head, is able to "manipulate" what it can see simply by moving its head;
(b) *Passive categorisation*, the opposite of active categorisation, when a robot is stationary or has few degrees of freedom, in a way that it cannot move in directions enabling it to reach an optimum perception. An example of passive categorisation is a robot with a fixed camera, which must solve the task of recognising the shape of some geometric figures. In this case, the robot is not able to alter the image perceived by the optical sensors of the camera.

Another possible classification of the categorisation ability is based on the emergence mechanisms:

(a) *Behavioural categorisation*, that is the ability to produce different behaviours in order to discriminate different environmental situations. Usually these behaviours are the result of a sequence of interactions between the agent control system (with its body) and the environment. For example, in an environment with some edible and non-edible food, an agent should be able to reach the edible, but stay away from the non-edible food;
(b) *Categorical perception*, that is the agent's ability to produce different behaviours as a consequence of different environmental conditions, by using its internal states' "memory". In this type of categorisation, the control system of the agent maps groups of sensory stimuli, belonging to a same category, into the same internal states.

The behavioural categorisation can arise from a careful and parsimonious exploitation of local environmental properties during the interaction between the agent's body and the environment, for this reason it may occur with purely reactive agents. Reactive behaviour is defined as the behaviour resulting from a direct association between the sensory input of the robot and its actuators, namely without making use of representations or internal states. Some researchers have designed an experimental set-up to study the behavioural categorisation, where a robot is placed in a circular room, whose perimeter is marked by separating it into 40 cells [1]. The purpose of the robot is to reach the left half of the environment, starting from any point on the circumference, and remaining on it. The control system consists of a neural network with 20 sensory neurons, which encode the number of the environmental cell where the robot is placed at a given time-step. The neural network has no hidden neurons, that makes it a perceptron. Using a genetic algorithm to evolve the robot, by a fitness function which computes the number of life cycles in which the robot remains in the left part of the environment, the robot can always find a solution that exploits only the sensory-motor coordination. In fact, the robot reacts consistently to the same sensory pattern, as the evolutionary

process selectively define some "behavioural attractors". In this case the attractors are represented by pairs of adjacent cells belonging to the left semi-circle. The robot continuously wanders clockwise and counterclockwise, rapidly reacting to these attractors. Finally, the robot's gets stationary on the left side of the environment, at some point.

Categorical perception is required when the autonomous agent needs internal states for mapping a sequence of stimuli received to the relevant category: for this reason, categorical perception requires the ability to discover the categories and the relation among the different categories, as well as the behaviours that should be exhibited when the event, related to a category, occurs. The opportunity to develop skills in perceptual categorisation has been investigated in a series of experiments simulating two e-puck robots. E-pucks are placed in a square environment, bounded by walls. Two target areas are located inside the square and each one has a different colour. The robot's aim is to stay in the same target area [3]. The control system of the robot consists of a neural network with 14 sensory neurons, which encode the state of activation of eight infrared sensors, one ground sensor (which is activated when the robot is on the top of a target area), four directional communication sensors and an input–output communication connection. Moreover, there are two hidden neurons and two output neurons controlling each engine, and one output neuron controlling the emitter of the communication signal. During the evolutionary process, the robots' reward is to stand in the same target. At the end of the evolutionary process, robots can solve the task by developing some strategies based on a rudimentary form of communication. During the first generation, the robots develop some remarkable exploratory skills consisting of attempts to avoid the walls of the environment. Then, each robot develops more purely social skills such as emitting signals that affect the trajectory of other robot. For example, if both robots stay in the same target area, they begin to produce a different sequence of signals merging both the information from the infrared sensors and from communications, with the ultimate effect of causing the two robots to stay in the same target area. So, communicative behaviours allow the two robots to "feel" when they are outside the target area, and to approach and remain relatively close.

A special case of categorical perception is the active categorical perception which may arise when internal states are not sufficient to discover the regularities necessary to discriminate between different categories of stimuli, because maybe regularities are hidden or absent. In fact, in these situations, the sensory stimuli experienced by an agent are co-determined by actions accomplished by the agent itself. Active categorical perception has been recently analysed in an experiment where a simulated anthropomorphic robotic arm, equipped with tactile sensors, was required to perceptually categorise spherical and ellipsoidal objects [4]. The anthropomorphic robotic arm was equipped with proprioceptive and tactile sensors distributed on the external part of the arm. The whole arm mainly consisted of three elements: the arm, the forearm, and the wrist. The arm's controller consisted of a continuous-time recurrent nonlinear network (CTRNN) with 22 sensory neurons, 8 internal neurons, and 18 motor neurons. The neural controller was trained by an evolutionary process in which the free parameters of dynamical neural networks

were varied randomly and in which variations were retained or discarded on the basis of their effects on the overall ability of the robots to carry out their task. Results indicate that robots were capable of developing an ability to effectively categorise the shape of the objects despite the close similarities between the two types of objects, the difficulty of effectively controlling the arm, and the need to reduce the effects produced by gravity, inertia, collisions, etc. More specifically, the best individuals were able to correctly categorise the objects located in different positions and orientations, as well as to generalise their skill to objects positions and orientations never experienced during evolution.

2 Materials and Methods

The purpose of this study is to evolve an embodied and situated agent in order to solve the non-trivial problem of how to discriminate objects of different shape and size. The agent is provided with a simple sensory-motor system and the capability of integrating information, over the time. In particular, the aim of the experiment is to understand how external "linguistic" inputs (labels) can facilitate categorisation abilities.

2.1 Experimental Set-up n.1

A simulated robot lives in an environment whose floor displays one geometric figure. Position and size of the figure are selected randomly. Conventionally, only two kinds of figures can be displayed on the environment (even though, theoretically, there might be more than two); in particular, these shapes are an equilateral triangle and a square. The environment is essentially a square arena of 80×80 cm, surrounded by walls. In order to make the categorisation process more complex with respect to the classical problems described in literature, noise has been added to the sensors' inputs. Noise is randomly spread on the whole arena floor, with a probability of 30 %. The noise granularity is set to have a ratio of 4:1, which means that each noise pixel is composed of four screen pixels.

Every robot has a circular body with a radius of 11 cm and is equipped with two motors controlling the movements of two wheels. Moreover, robots are provided with eight infrared sensors placed all around the body perimeter. The infrared sensors are able to detect the presence of obstacles up to a distance of 15 cm. On the lower surface of the robot's chassis (body) there are eight ground sensors, which can detect the colours of the floor. The ground sensors return a value (normalised between 0 and 1) depending on the detected colour tone: if one of the ground sensors is on the internal area of the geometric figure (dark grey), then the sensor will return a value close to 1; otherwise, if a ground sensor is external to the figure (i.e. the white arena floor), then it will return 0. Finally, the robot is equipped

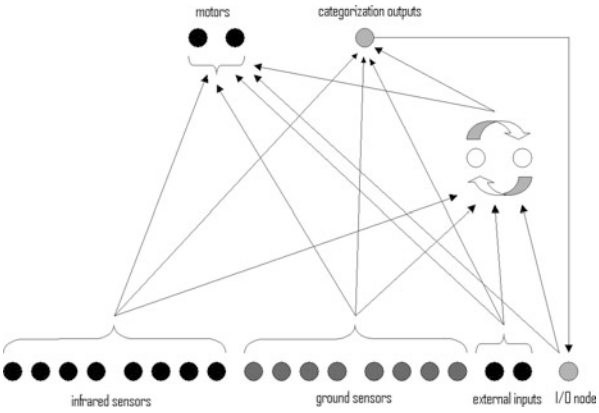

Fig. 1 Architecture of the robot neural controller

with two more sensors (radio receivers) able to receive "linguistic inputs" from outside of the environment (a hypothetical trainer or another robot). The robot returns a categorisation pattern (by a specific output), corresponding to the category recognised by the robot for that object.

The robot control system consists of a neural network with recurrent hidden neurons. The neural network is made of 24 neurons, which are organised into three different layers: input, output and hidden. The input layer contains 19 neurons encoding the activation state of the corresponding input sensors; there are eight neurons receiving the signal from the infrared sensors, eight neurons receiving the signal from the ground sensors, one neuron receiving the recurrent activation of the categorisation output and two neurons receiving the external linguistic input. The inner layer consists of two hidden neurons, which are connected with each other by two recurrent connections. They are encoded by the "leaky" mode. The output layer consists of two neurons, which, respectively, control the speed of two motors, and one neuron controlling the categorisation output. The categorisation output is binary-encoded: if the neuron returns a value less than 0.5 (and greater than 0.0) it means the "square" category, otherwise, if it returns a value greater than 0.5 (and less than 1.0) it represents a "triangle". The neural network topology (see Fig. 1) is fully connected: there are full direct connections between the input and the hidden layer and between the hidden and the output layer.

The free parameters of the robot neural controller, i.e., the synaptic weights of the connections, the biases, and the time constants, are encoded in the individual genotypes of the genetic algorithm. Connection weights are encoded into eight bit strings, and normalised into the $[-5.0, +5.0]$ interval. Instead, time constants are normalised into the range $[-1.0, 1.0]$. The initial population consists of 100 random genotypes (individuals), which represent synaptic weights, biases and time constants of the 100 corresponding neural networks. At the start of each generation, every genotype is converted into the numeric description of the corresponding neural network. Then, the numeric string associated with a neural network is

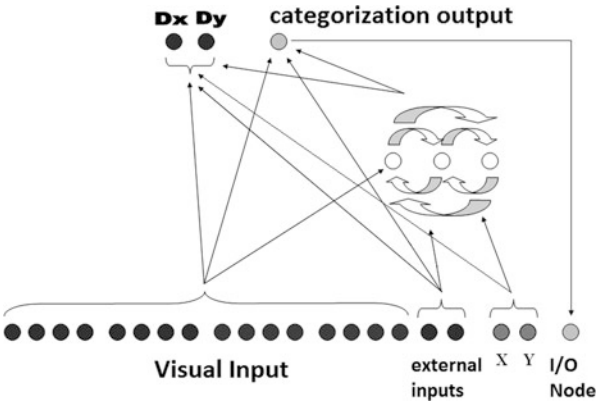

Fig. 2 Architecture of the fovea neural controller

loaded into the robot control system, which performs actions (using the current genotype) corresponding to the robot's entire life (2,000 cycles). The 20 best genotypes (with respect to the fitness) are allowed to reproduce themselves by generating an offspring of five individuals ($20 \times 5 = 100$ individuals populating the next generation) with a mutation probability of 2 % (2 % of the bits are randomly replaced by a new value). The whole evolutionary process needs 300 generations.

During each generation, every individual is evaluated for 20 trials lasting 2,000 cycles, with each cycle lasting 100 ms. At the begin of each epoch (trial) the robot wanders from a random position and orientation around the environment, but it always starts from outside the target area. The experiment was replicated ten times by using different initial populations.

The fitness function of the genetic algorithm is computed by adding $+1.0$ to the fitness score if the robot is located on the target area and its categorisation output is consistent with the figure shape; and -1.0, if the robot is placed on the target area and its categorisation output is not consistent with the figure shape; in all the other cases, the fitness does not change.

2.2 Experimental Set-up n.2

In order to verify the results of the first experimental set-up (comparing it with another experimental paradigm), a second set-up has been designed, being inspired by some previous experiments [5, 6]. The set-up n.2 is n.1-like, except that the second uses an evolutionary active vision system, consisting of a simulated "fovea" (a square visual area) instead of the robot. The fovea is able to move only horizontally or vertically with respect to the environment and cannot shift more than three pixels per cycle.

The fovea control system consists of a neural network with recurrent hidden neurons. The neural network (see Fig. 2) is made of 27 neurons. The input layer

contains 21 neurons encoding the activation state of the corresponding input sensors; there are 16 image neurons, two external inputs, two receiving the fovea position, and one the I/O node. The internal layer consists of three leaky recurrent hidden neurons. Finally, the output layer consists of two neurons, defining the fovea movements, and one neuron for the categorisation output.

3 Results

3.1 Results of Experimental Set-up n.1

By evolving experiment n.1 over 20 replications, it can be observed that the robot seems to solve the categorisation task correctly, by exploiting the external linguistic input. Initially, the categorisation output (returned by the robot) has not a constant trend, but it keeps being variable over time. The output takes on the right configuration only when the robot is placed on the top of the figure and receives the correct social external input (from the 1,000th cycle on). At the end of the evolution, the emerging strategy to solve the task is the same in all the test replications. Originally, the evolved robot exhibits a purely exploratory behaviour, by jumping from one wall to another and drawing curvilinear trajectories in the environment. This first strategy allows the robot to improve the chances of identifying the target area position, since the food zone is more likely to be located in the middle of the environment.

When the target area is detected, the robot exhibits a "line-following" behaviour, since it moves along the picture perimeter, in order to discover some peculiar features of the geometric shape (see Fig. 3). Then, when the robot appears to be located on a feature of the picture, such as an angle, it takes advantage of the "active perceptual categorisation" in order to classify the angles and thus identify the object shape.

Essentially, when the robot reaches the corner, it performs a "measurement" of the angle by integrating the information from its ground sensors with the representations evolved in its internal states. Depending on the category that the robot identifies, it produces a categorisation output that is closer to the value 0.0 for a square, while it is closer to 1.0 for a triangle. When the external inputs arrive (from the 1,000th cycle on), the correct categorisation is achieved. This fact shows that, from the 1,000th time-step on, the robot mainly makes use of external inputs in order to determine the true category.

The experimental set-up n.1 has been evolved again in conditions of total absence of external labels. It has been observed that in this conditions, the fitness values are lower than the values of the experiment evolved with external inputs.

In order to underline the effect of the external inputs on the individual's categorisation ability, a performance measure is computed by testing the robots in three different conditions: (a) robot evolved with external inputs and tested with

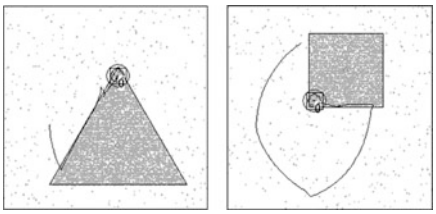

Fig. 3 The robot follows the figure perimeter looking for discrimination parts such as angles

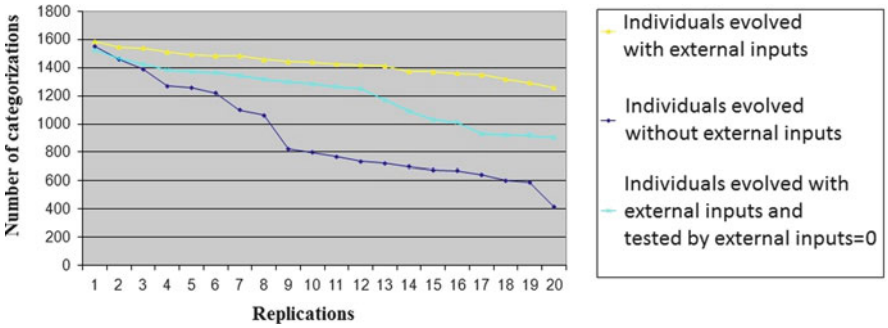

Fig. 4 Performance curves, sorted by number of categorisations, in the three conditions

external inputs; (b) robot evolved with external inputs and tested without external inputs; (c) robot evolved without external inputs and tested without external inputs.

Performance curves have been produced, by summing the total number of correct categorisations throughout the robot's lifetime. The performance measure is computed over 100 trials for each of the three conditions, sorting and plotting the values as depicted in Fig. 4. A statistical test (heteroscedastic Student's *t*-test, two-tailed) is performed on the distribution of 2,000 performance measures, for each condition (see Fig. 5). The *t*-test *p*-value is smaller than 1 % in all the pairs of comparisons.

The performance curves and statistics are in excellent agreement with the fitness results, i.e. the robots evolved with external inputs achieve better performances with respect to the robots evolved in total absence of external labels. Moreover, the performance curves and the statistics provide new information: the robot evolved with labels, and tested without such external inputs, can categorise better than the robot evolved without external inputs (Fig. 4). The robot seems to be able to "internalise" the social information (labels) available during evolution. Then the robot exploits the previous social information to take an advantage, in the test phase. Hidden nodes could play a key role in this process of internalising the social information. To prove this, the previous experiment is replicated after removing the hidden neurons. The performance curves of the new experiment (see Fig. 6) show that the robot evolved with external inputs and tested without them no longer achieves better performance in respect to the robot evolved without external inputs.

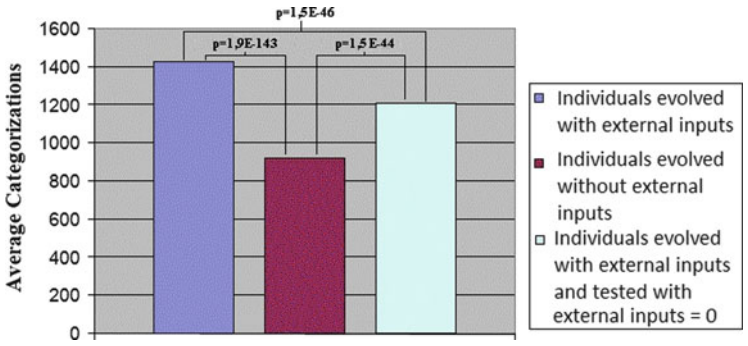

Fig. 5 Average number of categorisations calculated in 20 seeds × 100 epochs, for each condition. Student's *t*-test two-tailed heteroscedastic *p*-values are reported

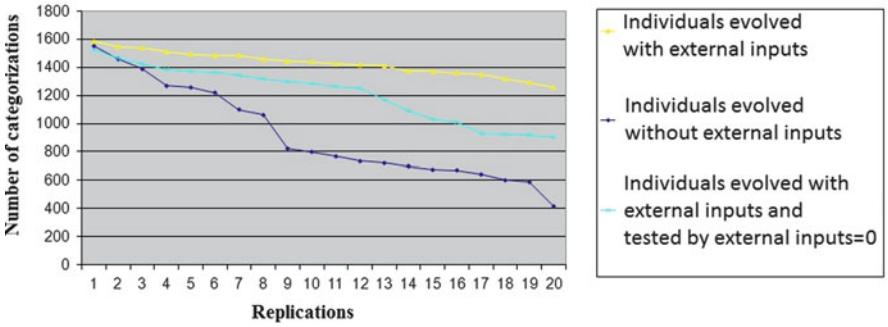

Fig. 6 Performance curves, sorted by number of categorisations, in the three conditions, without hidden neurons

This proves, without any doubt, the importance of internal hidden units for the formation of internal representations.

The hidden neurons also play an essential role in the process of integration process of sensory and social information.

In order to better illustrate how the hidden neurons affect the internal representations, a plot of the two hidden neurons' activations has been reported in Fig. 7. Sampled data (for each replication) consist of 20,000 values (2,000 activation values relating to the 2,000 life cycles × 10) belonging to the interval [0.0, 1.0].

Activation data have been captured in the three conditions: (a) individual evolved and tested with external inputs, (b) individual evolved without external inputs and tested without external inputs, (c) individual evolved with external inputs and tested without external inputs (set to 0). The *x*-axis shows the values of the first hidden node, H1, while the *y*-axis shows the values of the second hidden node, i.e., H2. Looking at the charts it is possible to notice that data in (a) and (c) are grouped in two distinct clusters representing two different internal states associated with two different categories. These are the internal representations. In condition (b),

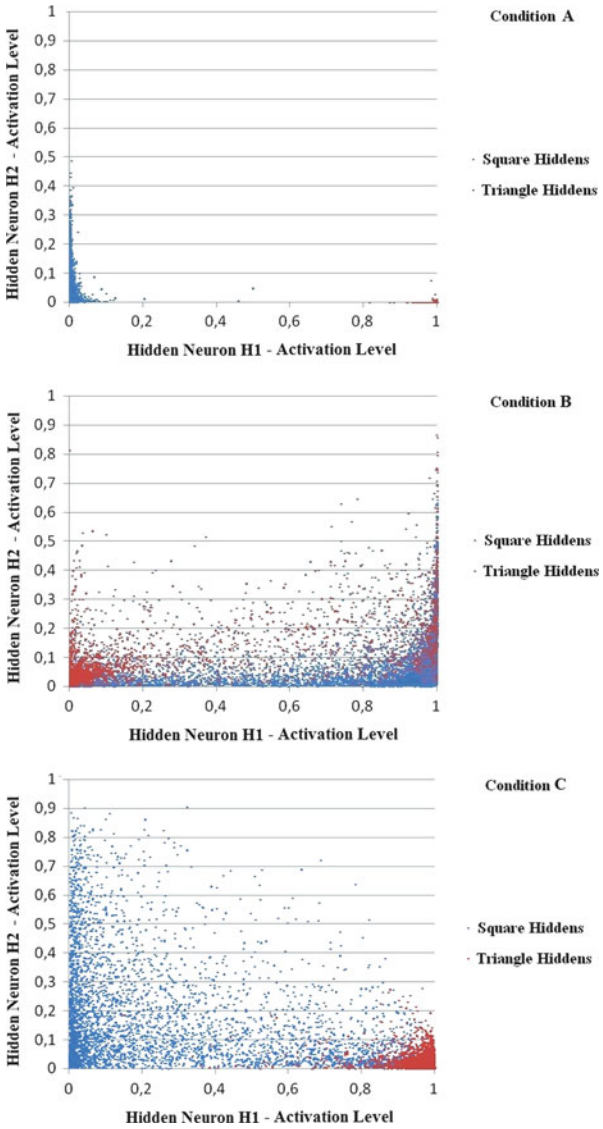

Fig. 7 Plot of the activation of the two internal neurons in the three conditions

the differentiation among the internal representations is less clear. In conclusion, the plots of the internal neurons activities confirm the assumptions: the external labels enable the neural network to create well-defined internal representations of the categories, which can be exploited later, even when these labels are no longer available.

Fig. 8 Different strategies of pattern recognition of the mobile retina

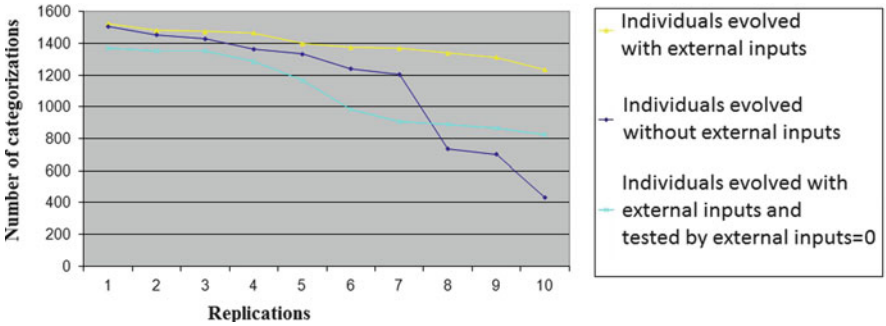

Fig. 9 Performance curves, sorted by number of categorisations, in the three conditions, of the active vision system

3.2 Results of Experimental Set-up n.2

The paradigm of the experimental set-up n.2 allows us to obtain the same results as in the experimental set-up n.1. However, the strategies used by the fovea for categorising are more sophisticated than in the set-up n.1. The mobile fovea, starting from a random position on the visual area (environment), initially exhibits a purely exploratory behaviour, since it is able to perceive only a limited portion of the displayed image. In general, the search strategy within the picture is to move along a spiral path from the initial position, thus the fovea can maximise the probability of detecting the position of the geometric shape. Whenever the fovea is positioned on the geometric figure, it can use two different strategies to categorise the object: (a) measuring the inclination of the geometric shape side (see Fig. 8a), which, for the square, is parallel to the horizontal or vertical axes of the visual field, while for the triangle, it is inclined by an angle of 60°; (b) measuring the angle at the corners of the geometric shape (see Fig. 8b), which for the square are right angles, while in the case of the triangle are acute angles (<90°) (see Fig. 9).

4 Conclusions

This work shows how a robot, evolved by genetic algorithms, is able to solve tasks that require complex cognitive skills such as the categorisation. The robot has been able to develop some emerging skills, such as the ability to exploit the information from the environment, for recognising objects of different shapes and sizes. Moreover, by using social information, the robot exhibits a "social" behaviour, learning to categorise the objects belonging to the environment and surrounding it, even though the social information is no longer available. This fact proves an important theoretical result: the social information, which is communicated by external inputs (another individual or a trainer), allows an agent to solve the task of understanding the surrounding environment more easily. This is true, since the external information enables the robot to create internal representations much better than when the external inputs are not available during the evolution phase.

Once these good internal representations are formed, during the evolution process, they enable the evolved agent to achieve an advantage in the categorisation task, even when, during the testing phase, the agent is deprived of the external information.

Acknowledgments I gratefully thank Stefano Nolfi, Davide Marocco, and Marco Mirolli for their helpful support.

References

1. Nolfi, S.: Categories formation in self-organizing embodied agents. In: Cohen, H., Lefebvre, C. (eds.) Handbook of Categorisation in Cognitive Science, pp. 869–889. Elsevier (2005)
2. Nolfi, S.: Behaviour as a complex adaptive system: on the role of self-organization in the development of individual and collective behaviour. ComplexUs **2**(3–4), 195–203 (2006)
3. Marocco, D., Nolfi, S., Ikegami, T.: Emergence of protosentences in artificial communicating systems. IEEE Trans. Auton. Mental Dev. **3**(2), 146–153 (2011)
4. Tuci, E., Massera, G., Nolfi, S.: Active categorical perception in an evolved anthropomorphic robotic arm. In: IEEE Congress on Evolutionary Computation (CEC'09), pp. 31–38, IEEE, Trondheim (2009)
5. Kato, T., Floreano, D.: An evolutionary active-vision system. In: Proceedings of the Congress on Evolutionary Computation, vol. 1, pp. 107–114, IEEE, Seoul (2001)
6. Mirolli, M., Ferrauto, T., Nolfi, S.: Categorisation through evidence accumulation in an active vision system. Connect. Sci. **22**(4), 331–354 (2010)

Regulatory Traits: Cultural Influences on Cultural Evolution

Alberto Acerbi, Stefano Ghirlanda, and Magnus Enquist

Abstract We use the term *regulatory traits* to indicate traits that both regulate cultural transmission (e.g., from whom to learn) and are themselves culturally transmitted. In the first part of this contribution we study the dynamics of some of these traits through simple mathematical models. In particular, we consider the cultural evolution of traits that determine the propensity to copy others, the ability to influence others, the number of individuals from whom one may copy, and the number of individuals one tries to influence. We then show how to extend these simple models to address more complex human cultural phenomena, such as in-group biases, the emergence of open or conservative societies, and of cyclical, fashion-like, increases and decreases of popularity of cultural traits. We finally discuss how the ubiquity of regulatory traits in cultural evolution impacts on the analogy between genetic and cultural evolution and therefore on the possibility of using models inspired by evolutionary biology to study human cultural dynamics.

A. Acerbi (✉)
Centre for the study of cultural evolution, Stockholm University, Stockholm, Sweden
e-mail: alberto.acerbi@gmail.com

S. Ghirlanda
Department of Psychology, Brooklyn College, New York, NY, USA

Centre for the study of cultural evolution, Stockholm University, Stockholm, Sweden
e-mail: drghirlanda@gmail.com

M. Enquist
Centre for the study of cultural evolution and Department of Zoology, Stockholm University, Stockholm, Sweden
e-mail: magnus.enquist@intercult.su.se

S. Cagnoni et al. (eds.), *Evolution, Complexity and Artificial Life*,
DOI 10.1007/978-3-642-37577-4_9, © Springer-Verlag Berlin Heidelberg 2014

1 Introduction

We use the term *regulatory traits* to indicate cultural traits that both regulate cultural transmission (e.g., from whom and when to learn) and are themselves subject of cultural transmission. In modern western societies, for example, parents actively transmit the idea that children should learn from schoolteachers, or teenagers attempt to persuade their peers not to listen to adults. Depending on our experiences, we can learn to be conformist or anti-conformist, which, in turn, can modify the outcome of future social interactions. Many other examples are possible.

Regulatory traits constitute, in our view, an important difference between cultural and genetic evolution. Tools from evolutionary biology have been extensively used to develop cultural evolutionary theory [1, 2], based on the assumption that the process of cultural change shares some fundamental properties with the process of genetic change: namely, variation, inheritance, and competition [3, 4]. While most researchers agree that culture has these properties, many have questioned whether they are sufficient to consider genetic and cultural evolution as essentially similar [5–7]. Some differences are obvious: for example, whereas genetic transmission is necessarily from parents to offspring, cultural transmission can, in principle, occur between any two individuals [8]. Thus models of genetic evolution certainly need to be modified to apply them to cultural evolution. The main question, however, is whether cultural and genetic evolution are different enough that it is misleading to study them using essentially the same models.

Population genetics models generally assume that the rules of transmission are stable. Consequently the routine assumption of cultural evolution models inspired by evolutionary biology is also that the rules of transmission are stable, or change slowly under genetic influence [1, 9]. However, this is not necessarily true: regulatory traits seem ubiquitous in cultural evolution—and not in genetic evolution (see Sect. 4). Here we explore the consequences of this difference on cultural dynamics.

We first use simple, yet very general, mathematical models to show that regulatory traits may have a profound impact on cultural evolution. Then we show, using a mix of mathematical models and computer simulations (see also [10–12]), how to extend these models to investigate real-world cultural phenomena such as in-group biases, conservatism, and fashion dynamics.

2 Models

Our basic assumption is that, in social interactions, individuals tend to copy each other's cultural traits. This tendency, however, is modulated by the traits themselves. For example, someone with the idea that people are trustworthy is expected to copy others more often than someone who mistrust people. In our models, we either calculate the expected effect of an interaction between two

randomly chosen individuals, or the expected change, per time step, in the number or proportion of individuals carrying a particular trait. The key insight exploited in such calculations is that a regulatory trait also regulates its own transmission. Using the same example above, acquiring the idea that people are trustworthy will increase copying and thereby foster opportunities for modifying the same idea. Our calculations show clearly the dynamical forces generated by regulatory traits and enable us to predict the end-state of the population when only one trait is considered. When many regulatory traits influence cultural evolution, in-depth analysis of the dynamical equations and computer simulations are generally required and the resulting dynamics can be highly nontrivial, as we will show in Sect. 3.

2.1 Openness

We define openness, p, as the probability that an individual changes in a social interaction ($1 - p$ is thus a measure of conservatism). Let p_i be the openness of individual i, $p_i \in [0, 1]$, and let us consider a social interaction between individuals i and j such that i is less open j, $p_i < p_j$. Let $E[\Delta p_i]$ be the expected change in i's openness caused by the interaction. Because individuals tend to copy each other, the interaction is expected to increase p_i and decrease p_j:

$$E[\Delta p_i] > 0 \qquad E[\Delta p_j] < 0 \tag{1}$$

Since i is less open, she will on average change less than j, by hypothesis:

$$|E[\Delta p_i]| < |E[\Delta p_j]| \tag{2}$$

It follows that the average openness of the two interacting individuals, $\frac{1}{2}(p_i + p_j)$, is expected to decrease as a consequence of the interaction:

$$E\left[\Delta \frac{1}{2}(p_i + p_j)\right] = \frac{1}{2}E[\Delta p_i] + \frac{1}{2}E[\Delta p_j] < 0 \tag{3}$$

where the conclusion that the change is negative follows directly from Eqs. (1) and (2). Thus social interactions tend to decrease population openness, until all variation in openness is eliminated. This results in very conservative populations, in which the outcome of social transmission is, paradoxically, to eliminate almost all social transmission [11, 13, 14]. An intuitive justification for this result is that conservative individuals, for the very reason they are conservative, change more rarely than open individuals. Transitions from open to conservative thus occur more often than transitions from conservative to open. We will see in Sect. 3 that the drive toward conservatism can be overcome when the interactions of multiple cultural traits are considered, although it remains a powerful influence on cultural dynamics [10].

2.2 Persuasiveness

We define persuasiveness, q, as the probability that an individual causes another to change in a social interaction. We can show that cultural transmission favors high persuasiveness with the same reasoning leading to the conclusion that it favors low openness. In a social interaction between individuals i and j such that i is more persuasive than j, $q_i > q_j$, we expect, on average, that q_i decreases and q_j increases:

$$E[\Delta q_i] < 0 \qquad E[\Delta q_j] > 0 \tag{4}$$

Since i is more persuasive, j is expected to change more:

$$|E[\Delta q_i]| < |E[\Delta q_j]| \tag{5}$$

It follows that the average persuasiveness is expected to increase:

$$E\left[\frac{1}{2}(q_i + q_j)\right] = \frac{1}{2}\left(E[\Delta q_i] + E[\Delta q_j]\right) > 0 \tag{6}$$

This dynamics has been studied in detail in previous work [11, 13, 14] and will be discussed further in Sect. 3.

2.3 Social Networks: Whom to Listen to

Social networks are widely recognized as important determinants of cultural dynamics, but it is equally important to study how cultural dynamics determines social networks (see, e.g., [15]). An individual's social network influences the transmission of other cultural traits, and thus cultural traits that modify individual's social networks are regulatory traits according to our definition. By analogy with the reasoning we made for openness, we expect cultural evolution to favor individuals who accept only a few others as cultural models. The reason is that such individuals have a smaller chance of changing than individuals who are willing to copy from many. We thus expect transitions from large to small social networks to occur more often than transitions in the reverse direction.

Let us assume that individual i is susceptible of acquiring cultural traits only from n_i cultural models, chosen at random or according to some rule. If i copies one of her models, we assume for mathematical simplicity that she replaces her set of models with the model's set.[1] We ask how the average of n_i over the

[1] We have verified in computer simulations that this assumption is not crucial for our general argument, as long as i's set of models becomes more similar to that of her model as a result of the interaction.

whole population changes during cultural evolution. In an interaction between two individuals i and j the probability that i adopts j's set of models is the probability that j is in i's set of models, i.e., n_i/n, where n is the population size. If i adopts j's models, n_i becomes equal to n_j, hence the expected change in n_i is

$$E[\Delta n_i] = a\,\frac{n_i}{n}(n_j - n_i) \tag{7}$$

where a is the probability of learning from a cultural model in an interaction. Similarly, the expected change in n_j is:

$$E[\Delta n_j] = a\,\frac{n_j}{n}(n_i - n_j) \tag{8}$$

The average expected change is thus

$$E\left[\frac{1}{2}(\Delta n_i + \Delta n_j)\right] = \frac{1}{2}E[\Delta n_i] + \frac{1}{2}E[\Delta n_j] = -\frac{a}{2n}\left(n_j - n_i\right)^2 < 0 \tag{9}$$

where we have used Eqs. (7) and (8) and noted that the term $n_j - n_i$ appears with opposite signs in the two equations. Hence the result of the interaction is to reduce the average number of cultural models, favoring small social networks.

2.4 Social Networks: Whom to Talk to

The results above show that cultural evolution tends to limit individuals' opportunities to be influenced by others. Conversely, it tends to increase individuals' opportunities to influence others. In addition to favoring high individual persuasiveness (Sect. 2.2), cultural evolution is also expected to favor individuals who try to influence a large number of others. The reasoning is analogous to the one in the previous section.

Let us assume that individual i tries to convince of her own ideas only m_i other individuals, which we may call i's "target set." If i is copied by one of these individuals, the latter adopts i's target set as her own. In an interaction between individuals i and j the probability that i adopts j's target set is the probability that i is in j's target set, i.e., m_j/n, and the expected change in m_i is

$$E[\Delta m_i] = \frac{m_j}{n}(m_j - m_i) \tag{10}$$

Similarly, the expected change in m_j is:

$$E[\Delta m_j] = \frac{m_i}{n}(m_i - m_j) \tag{11}$$

The average expected change is thus

$$E\left[\frac{1}{2}(\Delta m_i + \Delta m_j)\right] = \frac{1}{2n}\left(m_j - m_i\right)^2 > 0 \tag{12}$$

which is obtained similarly to Eq. (9). This result shows that cultural evolution tends to produce individuals (or, generalizing, organizations such as religions or political parties) who, everything else being equal, try to influence as many others as possible, rather than limiting the set of potential cultural targets.

3 The Far-Reaching Consequences of Regulatory Traits

The models above show that cultural evolution can be profoundly shaped by the additive effects of repeated social interactions and that very simple assumptions can generate surprising results. We show below that, although we have so far considered rather idealized models, our results may shed light on actual cultural phenomena.

3.1 In-Group Bias

We saw in Sect. 2.3 that we expect cultural evolution to reduce the number of individuals' cultural models (irrespective of whether such reduction improves, say, the individuals' well-being or genetic fitness). This result directly bears on actual social networks once we take into account that, in reality, the set of one's cultural models is not an arbitrary list of individuals, but is itself based on various cues such as social class, ethnicity, and gender.

Consider, for example, a population subdivided into a number of recognizable groups, and in which two rules for determining whom to copy exist: "copy everyone," and "copy only individuals from my own group." We expect the second rule to spread because it results in a smaller set of cultural models. Indeed, let b_i be the number of individuals with in-group bias in group i, g_i the size of group i, and a the probability of learning from a cultural model. The expected number of individuals in group i that acquire the in-group bias in each time step is

$$a(g_i - b_i)\sum_i b_i$$

where the last two factors are the probability that an unbiased individual in group i (of which there are $g_i - b_i$) meets a biased individual from any group (of which there are $\sum_i b_i$). The expected number of group i individuals who lose the in-group bias is

$$ab_i(g_i - b_i)$$

because the bias can be lost only copying an unbiased individual from one's own group, as individuals with the bias do not copy those from other groups. The expected change in the number of biased individuals in group i is the difference of the two expressions above:

$$E[\Delta b_i] = a(g_i - b_i)\sum_i b_i - ab_i(g_i - b_i) = a(g_i - b_i)\sum_{j \neq i} b_j \qquad (13)$$

which is always positive as long as $b_i < g_i$ (not everyone in group i is biased) and there is at least one biased individual outside of group i. Thus the number of biased individuals is expected to increase until everyone is biased. While many mechanisms have been suggested to contribute to in-group bias [16] and other model-based biases [17, 18], it seems plausible that regulatory traits play a role in the origin of maintenance of such biases.

3.2 Openness and Conservatism

People, in real life, do not indiscriminately reject cultural information, and human populations do not become completely conservative, as predicted by the model described in Sect. 2.1. To investigate the circumstances under which a population can remain open to cultural influences, despite a tendency of cultural evolution to favor conservatism, we studied a more realistic model in which many cultural traits coevolve [10].

We modeled individuals as having both multiple cultural traits and preferences, i.e. positive or negative attitudes towards those traits. Preferences are themselves cultural traits that can be copied in social interactions. We assumed that the probability that an individual copies a potential cultural model is an increasing function of the individual's preference for the model's traits. Thus the probability to copy is highest when a model possesses many traits for which the observer has high preference. Low preferences make an individual conservative and should thus be favored by cultural evolution. Another force, however, promotes openness rather than conservatism. In fact, individuals who are too conservative fail to acquire cultural traits from others, and therefore cannot be copied since they don't display anything that observers can evaluate. Such an incentive to acquire traits is a form of the general incentive to persuade others described in Sect. 2.2. Our simulations elucidated the interplay between these forces, showing that an open population can be maintained when there are many cultural traits and/or when the efficiency of cultural transmission is low (the latter regulates how many traits and preferences can be acquired in a single social interaction). Figure 1a, as an example of the latter effect, shows how average openness varies in time in two populations in which ten cultural traits are present, but that differ in p_{adopt}, i.e. the efficiency of transmission (for the effect related to the increase of the number of cultural traits, see [10]).

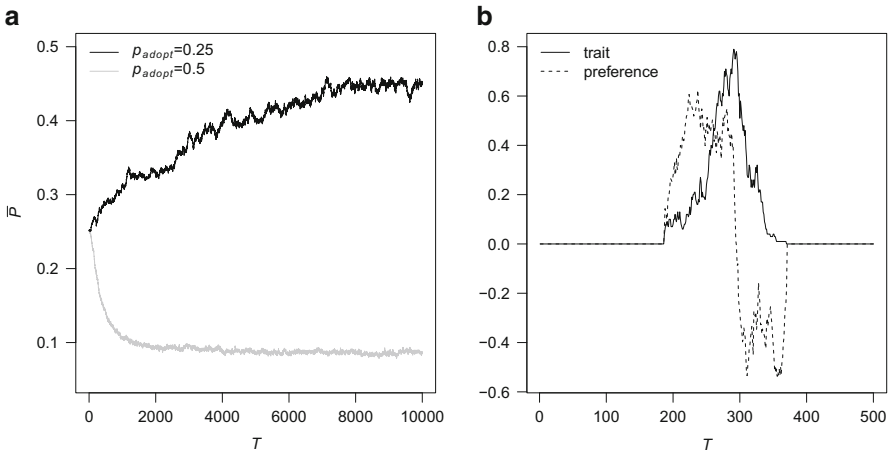

Fig. 1 (**a**) Average value of P (openness, X axis) through time (Y axis) in a population in which ten cultural traits are present, for an intermediate rate of transmission (*black line*: $p_{\text{adopt}} = 0.25$) and for a higher rate of transmission (*gray line*: $p_{\text{adopt}} = 0.5$). (**b**) An example of "fashion cycle." The *solid line* shows the frequency of the trait in the population through time, and the *dashed line* the average value of the preference associated with the trait. The graphs are redrawn from simulations detailed in [10] and [12], respectively

The increase of the number of cultural traits and the decrease of the efficiency of transmission have indeed an analogous effect on individual development: they both increase the number of interactions that an individual needs in order to acquire a substantial part of her culture. Successful cultural models are individuals who have many traits (so that they can influence others) and low preferences (so that they change rarely and thus repeatedly expose others to a stable set of traits). This combination can only be achieved by balancing openness and conservatism, remaining open during the first part of one's life, in order to learn as much culture as possible, and then become conservative to promote the spread of such culture [10, 12]. If the number of cultural traits is large and/or the efficiency of cultural transmission is low, an individual needs to spend a good part of her life acquiring culture, and thus must remain open for a relatively long time. Such relatively open individuals will be better cultural models than conservative individuals who have not acquired much culture. Their traits will thus spread in the population, including the relatively high preferences that make them open. Thus a population in which individuals need a good part of their life to acquire enough culture to be good models is predicted to remain relatively open. This picture fits well with empirical data showing that individuals become more conservative with age [19], but that individuals with higher education—in our model individuals with many cultural traits—remain open to new information into adulthood and old age [20].

3.3 Fashion Cycles

In a later study [12], we used the trait-preference model to study the fate of cultural innovations. Rather than limiting culture to a fixed set of traits, we allowed individuals to occasionally introduce new cultural traits. Individual preferences for new traits were randomly set at the moment of trait introduction. Thus it may happen, by chance, that a currently influential individual (someone possessing many traits others prefer) has a high preference for a new trait. Such a high preference can then spread as the individual is copied often by others, which in turn drives the spread of the new trait because, when the preference is common, individuals with the trait are better cultural models than individuals without the trait. As the trait becomes common, however the situation changes. Individuals with a low preference for a common trait gain an advantage in transmitting their traits, because they are more conservative than others, as explained in Sect. 2.1. This causes the low preference to spread, which in turn leads the population to abandon the trait, as possessing it is now a disadvantage. Figure 1b shows the complete cycle just described.

This dynamics offers a plausible and parsimonious explanation of fashions and fads, according to which rises and fall in trait popularity are a universal emergent property of cultural evolution, hinging on regulatory traits dynamics. Well-known examples of fashions and fads include clothing styles [21] and the popularity of pop records [22]. Further data indicate that fashion phenomena are present in all times and cultures. Symbolic features of Polynesian canoes such as paintings on paddles, for example, change more rapidly than functional features such as the shape of the hull [23], and analysis of decorative motifs in Neolithic pottery is consistent with the idea that individuals, or households, copied each other's motifs through time [24].

Our model explains two quantitative features of empirical data (compare Figs. 2 and 3). The first (Fig. 2a) is the power-law, or log-normal, distribution of frequency of cultural traits [25], meaning that only very few cultural traits become very common while the vast majority remains rare. This effect has been observed in several cultural domains, including, among others, first names, scientific citations, books translations, popularity of dog breeds (see [22, 25]). The second (Fig. 2b) is the finding that cultural traits that increase rapidly in popularity are also abandoned quickly, while slow increases in popularity correlate with slow decreases (shown for first names in the USA and France [26], and for dog breeds in the USA [12]).

The model accounts for these findings better than two common alternative views of fashion. The first is generally called neutral model of cultural evolution and simply assumes that individuals copy each other randomly [25]. This model has convincingly been shown to reproduce the long-tailed distributions of frequency of cultural traits, but fails in reproducing the correlation between the rates of increase and the rates of decrease of popularity. If individuals copy each other randomly, in fact, increases and decreases of popularity of a trait depend exclusively on the frequency of the trait at each time step, while the correlation requires some sort of "memory" (in our model provided by the coevolution of preferences and traits).

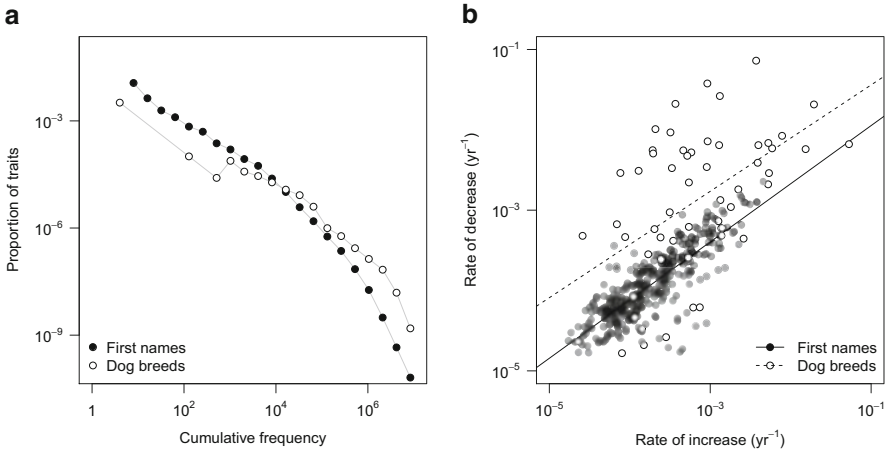

Fig. 2 Empirical findings on fashion cycles. (**a**) Distribution of frequency of cultural traits. (**b**) Rates of increase and rates of decrease of popularity of cultural traits. For both panels, *closed circles* are first names in USA 1880–2006, and *open circles* are dog breeds registered with the American Kennel Club, 1926–2005 (courtesy of Herzog)

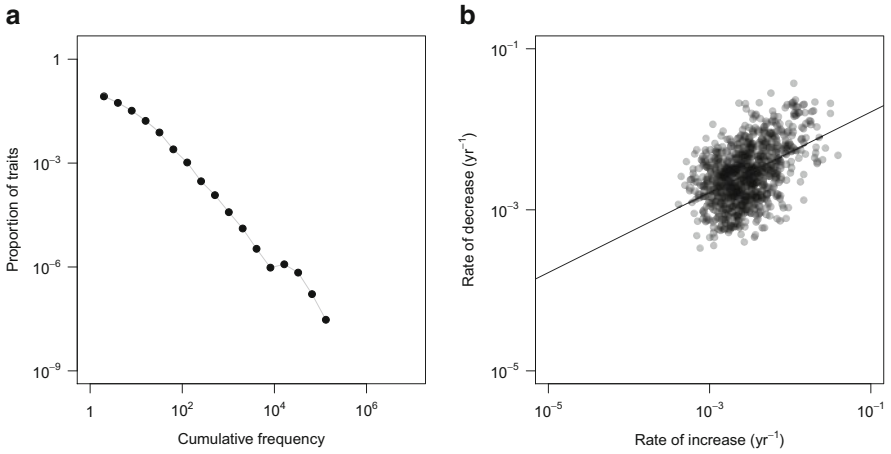

Fig. 3 Simulations results on fashion cycles. (**a**) Distribution of frequency of cultural traits. (**b**) Rates of increase and rates of decrease of popularity of cultural traits. Simulated time steps have been converted to years assuming an average lifetime of 70 years. The graphs are redrawn from simulations detailed in [12]

The second common view considers fashion cycles a direct result of social stratification [27]. According to this view, a cycle starts when individuals of low social status copy individuals of perceived high status. When a trait become common, however, high-status individuals abandon it to differentiate themselves

from low-status individuals, and, as a consequence, low-status individuals abandon it too. This *status* model, contrarily to the neutral model, generates correlations between rates of increase and rates of decrease of popularity, but not long-tailed frequency distributions. Indeed, as soon as a cultural trait become common, high-status individuals abandon it, triggering abandonment from low-status individuals. While the status model may describe brief fads, it seems unable to account for cultural traits that exhibit long-lasting popularity, for example English names such as Mary and John.

4 Discussion

In this chapter we extended our previous works to present a general argument about the importance of regulatory traits in cultural evolution. Our results show that regulatory traits may have a potent, and perhaps surprising, impact on cultural dynamics. Cultural evolution, in other words, can generate its own rules [13] in absence of extra-cultural driving forces, such as natural selection or memory mechanisms. This does not mean that these factors are unimportant, but studying pure cultural forces with mathematical and simulation models may help us to isolate and elucidate the effects of such forces.

A more general question is how the existence of regulatory traits impacts on the analogy between cultural and genetic evolution and, specifically, on the possibility to model cultural dynamics using models inspired by evolutionary biology. Note that "regulatory genes" are not analogous to cultural regulatory traits. Regulatory genes activate or inhibit other genes [28]; they do not alter how genetic material is transmitted. A genuine example of genetic regulatory trait, in the sense we use the term, is genes that determine the mode of reproduction (sexual vs. asexual), which are found in some species [29, 30].

Genetic regulatory traits, however, appear rare in genetic evolution, as witnessed by the remarkable success of population genetic models that employ immutable rules of genetic transmission [31–33]. By contrast, regulatory traits seem to be widespread in cultural evolution. Why is it so? Information transfer (or *modes of transmission* [7]) is highly constrained in genetic evolution. Although more flexible modalities of gene transfer exist [34, 35], genes typically propagate to offspring from just two (sexual reproduction of chromosomal DNA) or one parent (asexual reproduction or sexual reproduction of mitochondrial DNA). Cultural information instead can be transmitted in many different ways and, potentially, from any individual to any other individual, which creates the opportunity to regulate the flow of information in a more fine-grained and context-dependent way. Claidière and André [7] recently proposed that the instability of modes of transmission in culture is one of the major differences between cultural and genetic evolution. Our results suggest, indeed, that cultural forces can favor different kinds of information flow. We believe that the concept of regulatory traits can be important to identify and understand when and how this may happen.

Models of cultural evolution inspired by evolutionary biology have been criticized for their excessive simplicity, which has sometimes lead to a rejection of modeling altogether, especially within anthropology [36]. This is certainly not our conclusion. The possibility of applying models already developed in other disciplines to the study of culture is certainly positive. Modeling efforts, however, should progress toward a richer characterization of cultural dynamics. We hope to have shown that a complete theory of cultural evolution cannot ignore that cultural transmission can be more flexible than genetic transmission.

Acknowledgments Work supported by the Uniquely Human project funded by the Swedish Research Council.

References

1. Boyd, R., Richerson, P.J.: Culture and the Evolutionary Process. University of Chicago Press, Chicago (1985)
2. Cavalli-Sforza, L.L., Feldman, M.W.: Cultural Transmission and Evolution: A Quantitative Approach. Princeton University Press, Princeton (1981)
3. Lewontin, R.C.: The units of selection. Annu. Rev. Ecolo. Syst. **1**, 1–18 (1970)
4. Mesoudi, A.: Cultural Evolution. How Darwininan Evolutionary Theory Can Explain Human Culture and Synthesize the Social Sciences. University of Chicago Press, Chicago (2011)
5. Sperber, D., Claidière, N.: Why modeling cultural evolution is still such a challenge. Biol. Theor. **1**(1), 20–22 (2006)
6. Strimling, P., Enquist, M., Eriksson, K.: Repeated learning makes cultural evolution unique. Proc. Natl. Acad. Sci. **106**(33), 13870–13874 (2009)
7. Claidière, N., André, J.-B.: The transmission of genes and culture: a questionable analogy. Evol. Biol. **39**(4), 12–24 (2012)
8. Acerbi, A., Parisi, D.: Cultural transmission between and within generations. J. Artif. Soc. Soc. Simulat. **9**(1), 9 (2006)
9. Henrich, J., McElreath, R.: The evolution of cultural evolution. Evol. Anthropol. **12**, 123–135 (2003)
10. Acerbi, A., Enquist, M., Ghirlanda, S.: Cultural evolution and individual development of openness and conservatism. Proc. Natl. Acad. Sci. **106**(45), 18931–18935 (2009)
11. Ghirlanda, S., Acerbi, A., Enquist, M., Nakamaru, M.: The sometimes evitable route to conservatism and persuasiveness. Curr. Anthropol. **51**(2), 271–272 (2010)
12. Acerbi, A., Ghirlanda, S., Enquist, M.: The logic of fashion cycles. PLoS One **7**(3), e32541 (2012)
13. Ghirlanda, S., Enquist, M., Nakamaru, M.: Cultural evolution develops its own rules. The rise of conservatism and persuasion. Curr. Anthropol. **47**(6), 1027–1034 (2006)
14. Xue, J.Z., Costopolous, A.: The evitable route to zelaotry. Curr. Anthropol. **51**(2), 269–270 (2010)
15. Centola, D., Gonzalez Avella, J.C., Eguiluz, V.M., San Miguel, M.: Homophily, cultural drift, and the co-evolution of cultural groups. J. Conflict Resolut. **51**(6), 905–929 (2007)
16. Hewstone, M., Rubin, M., Willis, H.: Intergroup bias. Annu. Rev. Psychol. **53**, 575–604 (2002)
17. Henrich, J.: Cultural transmission and the diffusion of innovations: adoption dynamics indicate that biased cultural transmission is the predominate force in behavioral change. Am. Anthropol. **103**(4), 992–1013 (2001)
18. Richerson, P.J., Boyd, R.: Not by Genes Alone: How Culture Transformed Human Evolution. The University of Chicago Press, Chicago (2005)

19. Caspi, A., Roberts, B.R., Shiner, R.L.: Personality development: stability and change. Annu. Rev. Psychol. **56**, 453–484 (2005)
20. Costa, P.T.J., McCrae, R.R.: Comprehensive Handbook of Personality and Psychopathology. Wiley, Hoboken (2006)
21. Belleau, B.D.: Cyclical fashion movement: women's day dresses: 1860–1980. Cloth. Text. Res. J. **5**(2), 15–20 (1987)
22. Bentley, R.A., Lipo, C.P., Herzog, H.A., Hahn, M.: Regular rates of popular culture change reflect random copying. Evol. Hum. Behav. **28**, 151–158 (2007)
23. Rogers, D.S., Ehrlich, P.R.: Natural selection and cultural rates of change. Proc. Natl. Acad. Sci. **105**(9), 3416–3420 (2008)
24. Bentley, R.A., Shennan, S.J.: Cultural transmission and stochastic network growth. Am. Antiquity **68**(3), 458–485 (2003)
25. Bentley, R.A., Hahn, M., Shennan, S.J.: Random drift and culture change. Proc. Roy. Soc. Lond. B **271**, 1443–1450 (2004)
26. Berger, J., Le Mens, G.: How adoption spread affects the abandonment of cultural tastes. Proc. Natl. Acad. Sci. **106**(20), 8146–8150 (2009)
27. Simmel, G.: Fashion. Int. Q. **10**, 130–155 (1904)
28. West-Eberhard, M.J.: Developmental Plasticity and Evolution. Oxford University Press, New York (2003)
29. Stelzer, C.-P., Schmidt, J., Wiedlroither, A., Riss, S.: Loss of sexual reproduction and dwarfing in a small metazoan. PLoS One **5**(9), e12854 (2010)
30. Sandrock, C., Vorburger, C.: Single-locus recessive inheritance of asexual reproduction in a parasitoid wasp. Curr. Biol. **27**, 433–437 (2011)
31. Crow, J.F., Kimura, M.: Evolution in sexual and asexual populations. Am. Nat. **99**, 439–450 (1965)
32. Maynard Smith, J.: The Evolution of Sex. Cambridge University Press, Cambridge (1978)
33. Futuyma, D.J.: Evolutionary Biology. Sinauer, Saunderland (1998)
34. Gogarten, J.P., Townsend, J.P.: Horizontal gene transfer, genome innovation and evolution. Nat. Rev. Microbiol. **3**, 679–687 (2005)
35. Keeling, P.J., Palmer, J.D.: Horizontal gene transfer in eukaryotic evolution. Nat. Rev. Genet. **9**, 605–618 (2008)
36. Ingold, T.: The trobule with 'evolutionary biology'. Anthropol. Today **23**(2), 13–17 (2007)

Building Up Serious Games with an Artificial Life Approach: Two Case Studies

Onofrio Gigliotta, Orazio Miglino, Massimiliano Schembri, and Andrea Di Ferdinando

Abstract Artificial Life (AL) studies how to reproduce life-like phenomena exploring the life as could be in artificial systems (software, hardware, or hybrid). This challenging scientific perspective has produced a number of programming techniques often applied to solve concrete problems (Data Analysis, Process Optimization, Social Simulations, etc.). Computer Gaming is a field where AL techniques are applied. There are many successfully Alife products for pure entertainment (e.g., Tamagotchi and Creatures) and for educational objectives (e.g., Avida-Ed). However, we notice that all AL-Based games share a general flavor: they refer in someway to biological scenarios. In other terms, they represent often a sort of popularization of AL experiments designed for non-scientists. In this paper we argue that AL programming techniques (or more basically bio-inspired computational algorithms) could be used to develop generic games (e.g., sports, adventures, business games, etc.) without any relation with a biological perspective. We describe BreedBot and Learn2Lead, two Serious Games that we think could be paradigmatic examples about how to use AL techniques in different ways and fields that could be very different from their biological roots. BreedBot and its sequels (BestBot and BrianFarm) have been developed to disseminate the core-concepts of Autonomous Robotics and Learn2Lead has been developed to teach Psychological Theories of Teamwork in Small and Medium Enterprises. In BreedBot, AL techniques are used to develop the player–game interaction and they are explicitly visible by the user (he/she has to train/evolve a population of artificial agents). At the opposite side, Lear2lead has an *old style* appearance but it hides an AL engine. In this case AL techniques are used to model the game mechanics

O. Gigliotta · O. Miglino (✉)
University of Naples Federico II, via Porta di Massa 1, Naples, Italy
e-mail: onofrio.gigliotta@unina.it; orazio.miglino@unina.it

M. Schembri · A. di Ferdinando
ISTC-CNR, via San Martino della Battaglia 44, Rome, Italy
e-mail: massimiliano.schembri@istc.cnr.it; andrea.diferdinando@istc.cnr.it

S. Cagnoni et al. (eds.), *Evolution, Complexity and Artificial Life*,
DOI 10.1007/978-3-642-37577-4_10,

(e.g., artificial team dynamics and avatars' behavior). Both games are also able to be played online (www.nac.unina.it/bestbot2 and www.unina.l2l.it).

1 Introduction

The term Artificial Life (AL) was originally coined by Cristopher Langton with the aim to shed light on life as it could be possible in a vast number of settings and ways [1]. Without describing what is or is not Life (see [2] for this issue), that is beyond the scope of this paper, we can state that the milestone that made feasible the shift from biological to artificial life was the possibility to replicate, in modern computers, life-like phenomena through dynamical systems made up of interacting agents (the bigger the system, the better the result). One of the first attempts was the Conway's Game of Life, a simple game in which every cell of a lattice could switch from alive to dead following some basic rules (it is easy to trace the roots of this work back to von Neumann and his cellular automata), but only in the last two decades this field has flourished in a rich and interesting manner, covering many disciplines spanning from biology to social sciences. So far, within AL, a lot of tools have been fruitfully used: cellular automata, neural networks, agents simulation and the like, aiming to study quite different scientific phenomena that share a common trait in being the emergent outcome of dynamical interactions among simple agents at a lower scale [3]. Useful examples are those coming from social insects life: hoards of ants are able to find food in effective way by interacting with each other through stigmergy, bees can communicate the presence of food with a fancy dance, and fireflies can synchronize their firing rate using a simple visual feedback [4]. With tons of fascinating examples in nature, very soon AL scientific toolkit was borrowed to understand the life as it is along with the life as it could be [5–8]. AL has had an impressive impact on scientific investigations and has inspired several applications in different fields (robotics, data analysis, process optimization, etc.). Computer Gaming is a field where AL techniques are applied. There are several successfully Alife products for pure entertainment (e.g., Tamagotchi and Creatures) and for educational objectives (e.g., Avida-Ed). However, we notice that all AL-Based games share a general framework of application: usually they refer directly to biological topics. In other terms, they represent a sort of popularization of AL experiments designed for non-scientists. For example, a Tamagotchi player has to take care of an artificial pet or playing with Creatures we can determine the emergence of life in a digital universe. In this paper we argue that AL programming techniques (or more basically bio-inspired computational algorithms) could be useful to build up games for a wide range of genres (sports, adventure, business, etc.). In other words, the AL programming techniques could be used as engines to generate games not strictly involved in conveying biological related contents. We will provide concrete examples of this perspective describing two Edutainment games developed by our Lab. Usually Educational Games are produced by little developers team and they can be deployed as case-studies to test and validate innovative production

techniques. In particular, Edutainment is a neologism that puts together education and entertainment. In other words it is a field in which education and learning are linked with ludic and playful experiences [9]. Although historically not new, the idea has grown thanks to modern computers (provided with appealing graphics and sound quality) and Internet. Recently, a new strain of games called *Serious Games* (SG), stemmed from edutainment systems, has gained consensus among educators. At first glance, the term serious game seems an oxymoron, while a deeper examination renders a clear picture of a powerful learning tool. A useful definition of the term game has been provided by Michael and Chen [10]

> . . .games are voluntary activity, obviously separate from real life, creating an imaginary world that may or may not have any relation to real life and that absorbs the player's full attention.

The adjective serious refers to the fact that this activity must rely on a set of rules closely related to a particular phenomenon (e.g., a driving simulator). Generally, serious games are meant to be digital. They make use of modern digital technology to convey educational contents to a specific target audience. In particular, they have been used in many areas. There are military, political, religious, artistic, healthcare, and government serious games [11]. The rationale behind the introduction of digital games in education is manifold. First of all, serious games can be seen as complex laboratories in which users can experience situations that can be too costly or too risky to replicate in the real world (e.g., flying with a Boeing). Serious games can be easily deployed, an Internet connection and a computer or a mobile phone in many cases are quite sufficient. The United States government with the serious game America's Army has helped recruitment of new soldiers saving 75 % of the cost comparing to a normal recruitment program [11]. Nowadays digital experience is particularly prominent and widespread, hence Serious Games provide contents in a very familiar and usable way. Finally, serious games try to provide educational contents in a playful environment that can be highly motivating in a learning context. AL techniques are particularly effective to simulate and reproduce very complex systems, ranging from biological (neural cells) to social ones. A complex system here is meant as a system whose behavior emerges from the interactions of many elements which could be either simple cells, like neurons, or human beings. SG can exploit the power of AL at least in two ways. In the first one, games use an AL explicit interaction mechanism. In this case, the user interacts (plays) with a game using traditional AL methods (e.g., by evolving a population, training an organism, setting up an ecological system, etc.). In the second one AL is used as bio-inspired tool to model complex system that is placed in "game engine." In this case, the user interacts with the game using traditional method. In this paper we present two serious games based upon an artificial life core in two different ways. The first serious game, Breedbot (along with its sequels Bestbot and Brainfarm) has been used to teach Autonomous Robotics and it uses an interaction mechanism developed with an AL approach. The other Serious Game, Learn2Lead, has been designed to teach psychological theories of group dynamics. It appears as a traditional game but it hides an AL engine. Both games are also able to be played online (www.nac.unina.it/bestbot2 and www.unina.l2l.it).

2 AL Used to Build Up Autonomous Artificial Organisms Through an Evolutionary Guided Design: The Case of Breedbot, Bestbot, and Brainfarm

Breedbot, Bestbot, and Brainfarm are integrated software/hardware platforms. By using these software every user, without any particular computer skill, can train/breed, within customizable virtual worlds, artificial organisms that can be downloaded onto the real counterparts: real robots made up of Lego components. The rationale behind those edutainment systems relies on the possibility to give users a rich interactive system able to link virtual and real environments through intelligent hardware. Nowadays robotic hardware are very affordable, the market is full of robot toys and also our houses start to host little and effective cleaning robots, hence having house-wandering intelligent machines in the future will be very common. From an edutainment perspective, using robots in education implies the possibility to exploit users' manipulation skills to favor a better and a deeper learning process. In the next sections we describe with more details the three edutainment systems.

2.1 Breedbot

Breedbot has been developed as an edutainment hardware/software system aiming to introduce users into the world of autonomous robotics [12,13]. The software side of Breedbot allows users to breed a population of nine wheeled robots. Every robot is provided with three infrared sensors, to detect nearby obstacles, and two motors that control wheels' speed. The three infrared sensors are placed on the right, on the center, and on the left of the robot's body in order to maximize its visual field. A differential drive system allows robots to steer in any directions (see Fig. 1 Right).

Each robot is controlled by a simple feedforward neural network depicted in Fig. 1 Left. The input layer consists of three infrared sensors and two motor context units that are simply two relay units of the previous motor activation (all inputs are normalized between [0,1]). The five input neurons finally project to the output layer made up of two motor neurons that control the right and left motor of the robot. Motor activation is computed according to the standard logistic equation (Eq. 1).

$$O_j = \frac{1}{1 + e^{-A_j}} \tag{1}$$

where O_j and A_j are, respectively, the output and the net input of the j_{th} motor neuron. Robot's speed, finally, is updated according to the activation of the motor neurons ranged between [−14,14] mm/s. The neural network parameters are in turn encoded in a genetic string that will undergo an evolutionary process guided by either the users (artificial selection) or the machine (automatic selection). The real robot is made up of Lego Mindstorms components and custom infrared sensors (Fig. 1 Right). The software side communicates with the robot by an infrared link.

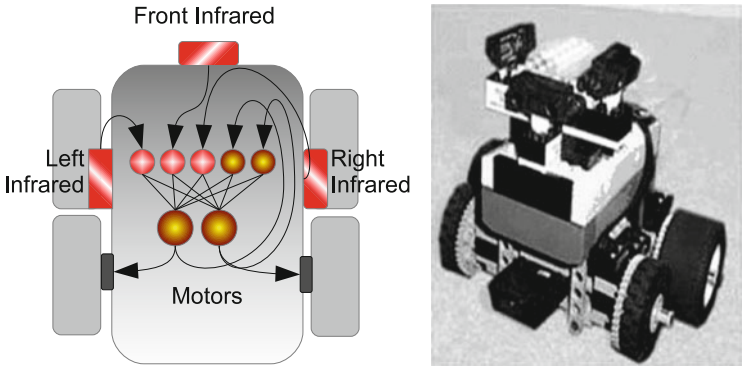

Fig. 1 *Left*: Schema of the robot used in Breedbot provided with a neural controller that links the input layer (three infrared sensors and two motor context units) to two motor neurons. *Right*: Real robot made up of Lego components

Breedbot simulates, in a fully customizable digital environment, a population of nine robots. At the beginning of each simulation, the computer screen shows the initial generation of robots, generated with random genetic strings, in action. Users can observe the behavior of the agents and decide whether to use artificial or automatic selection. In the first case users, whenever they want, can select up to three robots as parents to generate a whole new population. According to the evolutionary process, offspring are generated by cloning and mutating parents' genotypes [14, 15]. These steps (selection, cloning, and mutation) can be reiterated until users (breeders) find a satisfying solution. In the second case, the software selects each generation three robots by their ability to explore the surrounding environment. In both cases and at any time, users can download their neural controller onto the real robot, through an infrared protocol, in order to evaluate how it performs in real environments reacting to real obstacles.

2.2 *Bestbot*

Bestbot[1] is an online game stemmed from Breedbot. Nowadays social games are gaining consensus as well as thousands of players, so it was natural thinking about a simple porting of Breedbot. The AL engine is the same of Breedbot but Bestbot introduces a new graphical interface (3D physics engine powered by Unity) a new gameplay, two robots instead of nine, a more complex neural controller, and an improved sensory systems (three infrared sensors plus a camera). Users can train,

[1] http://eutopia.unina.it/bestbot.
http://eutopia.unina.it/bestbot2.

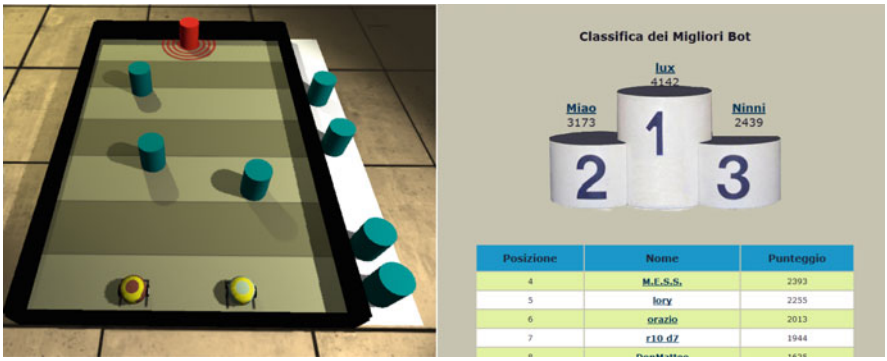

Fig. 2 *Left*: Bestbot arena with two robots. *Right*: Online ranking

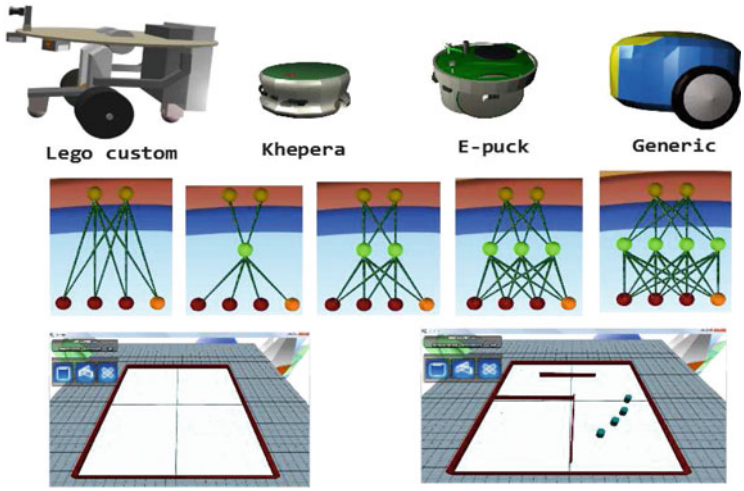

Fig. 3 *Top*: Robot supported in Brainfarm. *Center*: Custom neural networks. *Bottom*: Arenas

through a genetic algorithm, their robots to reach as soon as possible a colored target placed into a rectangular arena (Fig. 2 Left). Trained robots then can challenge other users' robots in an online contest. A final ranking, finally, shows robots' performance quality (Fig. 2 Right).

2.3 Brainfarm

Brainfarm represents a direct upgrade of Breedbot. This new software, as the name suggests, presents new features related to the possibility to design the architecture of the robots' brain (Fig. 3 Center). Users can use a simple feedforward network

or more complex architectures to control robots' behavior. Brainfarm, in fact, is a serious game designed to introduce students in what has been defined the century of the brain. Moreover, Brainfarm allows user to choose more robots than ever in order to get more fun and more hints about how brain and body work tightly coupled to solve survival problems [16]. Robots can be trained genetically or with an online learning algorithm such as Hebbian and the like. As in Breedbot and Bestbot, the robot's virtual controller can be evolved in a fully customizable environment (Fig. 3 Bottom) and downloaded onto several robots from Lego Next to Khepera, E-puck, and custom robots (Fig. 3 Top).

3 AL Used as a *Hidden* Game Engine: Learn2Lead

Learn to Lead (L2L) is a digital laboratory (2D web-based game) where an user (the leader) learns psychological leadership theories by governing a team of artificial agents (the followers). The game is based on the Full-Range Leadership Theory (FRL), a well-known scientific theory about leadership dynamics in small groups [17]. The game mechanics is developed by using AL techniques (agents-based modeling and artificial neural networks), in this software the AL engine is hidden to the users. In fact, it is only an effective technique to implement the FRL tenets, whereas in Breedbot and its sequels, the AL engine is an important part of the interactive process. The first version of L2L (Fig. 4) has been used in several European vocational courses about Leadership and Human Resources Management.

To become a good leader one has to study a lot of psychology, to attend very costly MBA courses or maybe to observe as much as possible human behaviors. This latter solution at first glance may seem very time- and resource demanding but implemented in the right way can be the best solution in leaders education. In fact, modeling human behavior in a proper agents simulator can offer learners the possibility to check how agents' behavior change varying a set of psychological variables. Although Learn2Lead could not substitute a long and professional training, as an implementation of a psychological theory in an agents simulator can be used as a powerful supporting tool in teaching how to successfully manage a group of followers [18].

Each follower has to accomplish a task, allocated by the leader, in different kinds of environments. The maximum workload is determined by two variables: ability and motivation. The player/leader can vary those variables through a series of possible actions (for example, by sending the follower to a training course, or by stressing them). Leaders get a score on the basis of the motivational and skill development of the followers. In the gameplay leaders are human players, whereas followers are artificial agents controlled by a connectionist network. The idea underlying this general framework is that in some given conditions, the leader agent (i.e., the player) has to take some decision about one or more followers. The game design described thus far requires a method for simulating the behavior of followers. The player must feel that his actions do have reliable and realistic consequences on

Fig. 4 An L2L environment

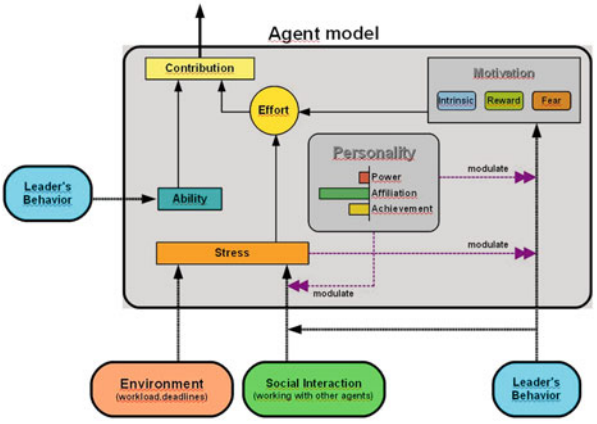

Fig. 5 L2L agent model. According to the FRL theory each agent behavior is guided by a set of external variables such as leader's behavior, workload, etc. and a set of internal variables related to personality, motivation, ability, and stress. This model in L2L has been implemented exploiting the mathematical power of neural nets

the behavior and development of their followers. In order to model the follower dynamics we used a bio-inspired computational approach: the teamwork was constituted by a set of artificial agents and each agent was controlled by a neural network implementing FRL theory (Fig. 5). In other words we used a combination of two techniques: agent-based modeling and artificial networks so as to combine social and cognitive factors [19]. Agent-based simulations are extensively used in many branches of natural and social sciences to study complex phenomena that are not safely reducible to a set of mathematical equations [20]. These phenomena

typically emerge from the interaction among individual entities. Examples are organisms living in an ecosystem or human beings acting in a society [21].

To build an agent-based simulation of the FRL theory we started modeling followers as artificial agents. Artificial Agents are essentially input–output systems with an internal state that changes over time depending on the external input and some internal variables. As shown in Fig. 5, every follower has some internal variables that affect the final contribution in getting through the jobs assigned. The most relevant to the FRL theory is the motivation. The motivation level is affected by three subcomponents: intrinsic, reward, and fear. The intrinsic component models the dynamics of intrinsic motivation and it is related to the transformational leadership style while the reward and fear components model extrinsic motivation and are related to the transactional leadership style. What differs among the three is the time dynamics, specifically their decay rate. For example, the intrinsic component has a slower decay rate than the reward and fear, but can be activated only by appropriate leader behaviors (typically pertaining to transformational style). The stress variable is linked to some external inputs like social interaction, workload, and deadlines. It affects the contribution and is an important aspect to keep under control during the game. Stress also has a modulator effect on the leader motivation-oriented behaviors. Personality and ability try to capture what the FRL theory says about individual consideration. Ability level is linked to follower performance. Personality is conceived as a modulator for the leader behavior so that the same leader action may have a different impact on followers with different personality. On the contrary, the leader that aim at raising the motivation of the team as high as possible needs to perform some individualized consideration.

4 Conclusions

In this paper we showed two ways to build serious game systems exploiting the power of AL techniques. In particular, we applied agent modeling, neural networks, genetic algorithms, and robotics. It is important to stress out the different meaning of the use of AL techniques in the two cases (Breedbot, Bestbot and Brainfarm vs L2L). In the first one, AL techniques such as neural networks and genetic algorithms are utilized as overt tools to guide artificial organisms design, so as to enable users to learn fundamental principles of autonomous robotics. The neural network, here, is just an artificial *brain* (resembling the functioning of living brains), and its structure is directly accessible and customizable by users as well as the genetic algorithm parameters that lead to the evolutionary process. Whereas in the second case, AL techniques are the mathematical bio-inspired implementation of a psychological theory (FRL): hence only a powerful way to implement a scientific theory. Our two cases show how AL techniques can boost serious game systems toward a new level of usability in the context of a bio-inspired evolutionary design process (evolutionary autonomous robotics) and in the field of management training.

The importance of such tools relies on the possibility they offer to train a target audience, in complex topics such as evolutionary robotics and group dynamics, with ease using multiple media in a playful context.

References

1. Langton, C. (ed.): Artificial Life. Addison-Wesley, Redwood City (1989)
2. Monod, J.: Chance and Necessity: An Essay on the Natural Philosophy of Modern Biology. Alfred A. Knopf, New York (1971)
3. Bedau, M.A.: Artificial life. In: Matthen, M., Stephens, C. (eds.) Handbook of the Philosophy of Biology, pp. 585–603. Elsevier, Amsterdam, (2007)
4. Trianni, V.: Evolutionary Swarm Robotics: Evolving Self-Organising Behaviours in Groups of Autonomous Robots. Springer, Germany (2008)
5. Webb, B.: Animals versus animats: or why not model the real iguana? Adapt. Behav. **17**, 269–286 (2009)
6. Gigliotta, O., Pezzulo, G., Nolfi, S.: Evolution of a predictive internal model in an embodied and situated agent. Theor. Biosci. **130**(4), 259–276 (2011)
7. Gigliotta, O., Miglino, O., Domenico, P.: Groups of agents with a leader. J. Artif. Soc. Soc. Simulat. **10**(4), 1 (2007)
8. Ponticorvo, M., Miglino, O.: Encoding geometric and non-geometric information: a study with evolved agents. Anim. Cogn. **13**(1), 157–174 (2010)
9. Veltman, K.H.: Civita Annual Report 2003, chapter Edutainment, Technotainment and Culture. Giunti, Florence (2004)
10. Michael, D., Chen, S.: Serious Games: Games that Educate, Train, and Inform. Thomson Course Technology, Boston (2006)
11. Susi, T., Johannesson, M., Backlund, P.: Serious games: an overview. Technical report, School of Humanities and Informatics, University of Skövde, Sweden (2007)
12. Miglino, O., Gigliotta, O., Ponticorvo, M., Nolfi, S.: Breedbot: an evolutionary robotics application in digital content. Electron. Libr. **26**(3), 363–373 (2008)
13. Miglino, O., Gigliotta, O., Ponticorvo, M., Nolfi, S.: Breedbot: an edutainment robotics system to link digital and real world. In: Apolloni, B., Howlett, R.J., Jain, L.C., (eds.) Knowledge-based Intelligent Information and Engineering Systems, vol. 4693, pp. 74–81. Springer, Berlin (2007)
14. Nolfi, S., Floreano, D.: Evolutionary Robotics. MIT Press, Cambridge, MA, USA (2000)
15. Nolfi, S., Gigliotta, O.: Evorobot*. In: Nolfi, S., Mirolli, M., (eds.) Evolution of Communication and Language in Embodied Agents. Springer, Berlin (2010)
16. Pfeifer, R., Bongard, J.C.: How the Body Shape the Way We Thinks. MIT Press, London (2007)
17. Bass, B.M.: Bass & Stogdill's Handbook of Leadership: Theory, Research and Managerial Applications. Free Press, New York (1990)
18. Di Ferdinando, A., Schembri, M., Nigrelli, M.L., Linehan, C., Miglino, O.: Learn to lead a web based game to teach leadership theories in vocational courses. In: Barajas, M., Trifonova, A., Veneri, A.D., Frossard, F., Mellini, B. (eds.) Games and Creativity in Education and Training. Fridericiana Editrice Universitaria (2011)
19. Sun, R.: Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation. Cambridge University Press, New York, NY (2006)
20. Borshev, A., Filippov, A.: From system dynamics and discrete event to practical agent based modeling: reasoning tecniques, tools. In: Proceedings of the 22nd International Conference of the System Dynamics Society (2004)
21. Haken, H.: Synergetics, An Introduction: Nonequilibrium Phase Transitions and SelfOrganization in Physics, Chemistry, and Biology. Springer, Berlin (1977)

Part IV
Applications

The Effects of Multivalency and Kinetics in Nanoscale Search by Molecular Spiders

Oleg Semenov, Darko Stefanovic, and Milan N. Stojanovic

Abstract Molecular spiders are nanoscale walkers made with catalytic DNA legs attached to a rigid body. They move over a surface of DNA substrates, cleaving them and leaving behind product DNA strands, which they are able to revisit. The cleavage and detachment from substrates together take more time than the detachment from products. This difference in residence time between substrates and products, in conjunction with the plurality of the legs, makes a spider move differently from an ordinary random walker. The number of legs, and their lengths, can be varied, and this defines how a spider moves on the surface, i.e., its gait. Here we define an abstract model of molecular spiders in two dimensions. Then, using Kinetic Monte Carlo simulation, we study how efficiently the spiders with various gaits are able to find specific targets on a finite two-dimensional lattice. Multi-legged spiders with certain gaits find their targets faster than regular random walkers. The search performance of spiders depends both on their gait and on the kinetic rate r, which describes the relative substrate/product "stickiness." Spiders with gaits that allow more freedom of leg movement find their targets faster than spiders with more restrictive gaits. For each gait, there is an optimal value of r that minimizes the time to find all target sites. Spiders influence each other's motion through stigmergy, and this also affects the search performance.

O. Semenov
Department of Computer Science, University of New Mexico, Albuquerque, NM 87131, USA

D. Stefanovic (✉)
Department of Computer Science, University of New Mexico, Albuquerque, NM 87131, USA

Center for Biomedical Engineering, University of New Mexico, Albuquerque, NM 87131, USA
e-mail: darko@cs.unm.edu

M.N. Stojanovic
Division of Experimental Therapeutics, Department of Medicine, Columbia University, New York, NY 10032, USA

Department of Biomedical Engineering, Columbia University, New York, NY 10032, USA

S. Cagnoni et al. (eds.), *Evolution, Complexity and Artificial Life*,
DOI 10.1007/978-3-642-37577-4_11,

1 Introduction

We are developing synthetic nanoscale walkers, called molecular spiders [1, 2], which are able to move across a surface, propelled by the multivalent chemical interactions of their multiple legs with the surface (Sect. 2). Molecular spiders may find use in biomedical applications, such as searching for clinically relevant targets on the surface of a cell. Here we present simulation-based results on the efficiency of concurrent search for multiple targets by multiple molecular spiders. Spiders and their targets are simulated on a finite two-dimensional grid of chemical sites (Sect. 3), which models the DNA origami surface [3,4] on which we will eventually carry out laboratory experiments.

Molecular spiders' legs are made of catalytic single-stranded DNA. These molecules are not known in nature, though they easily might have evolved in nature. Being both catalytic and information-carrying, they are candidate components for a synthetic biochemical artificial life. When we study them in simulation, we are not only taking inspiration from natural life, as in the field of artificial life, but we are also prototyping how the natural world can be refashioned and engineered, which one might call *real artificial life*.

The salient kinetic parameter governing the *walking behavior* of a molecular spider is the ratio r between the residence time of a spider's leg on a previously visited site (spent product) and the residence time on a new site (fresh substrate) [5–8]; r is at most 1 because substrates are generally "stickier" than products. As r is decreased, the motion exhibits an increasingly strong superdiffusive transient [7].

The conclusion of our *searching behavior* simulations (Sect. 4) is that the efficiency of search again strongly depends on r and the configuration of the spiders, but it is also influenced by the spiders' indirect (stigmergic) interactions through the modification of the surface. Among the studied spider configurations we found that those spiders whose legs have more freedom of movement find their targets faster than spiders with more restrictive rules of leg movement. Also, for the studied finite surface, we found that the efficiency of search does not vary monotonically with r; indeed, in each of the several configurations of multi-legged spiders we simulated, there was a distinct optimal r value. This behavior is a consequence of the interplay between the effects of the spiders' having multiple legs and the kinetic bias (Sect. 5).

2 Molecular Spiders

Cells in nature accomplish many of their complex tasks using self-assembled filament tracks and (linear) molecular motors that walk directionally along the filaments [9–14]. These *natural* protein motors solve the problem of efficient molecular cargo transport across the cell. Recent advances in single-molecule chemistry have led to *synthetic* molecular motors [15, 16], including molecular assemblies that walk over surfaces, typically following fabricated or self-assembled

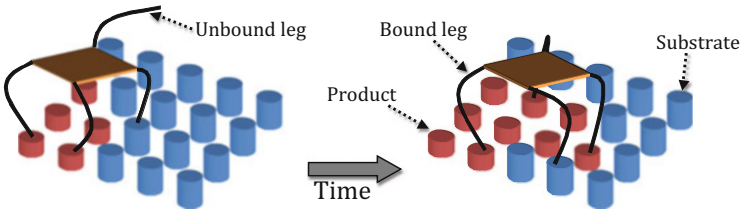

Fig. 1 A molecular spider moves over a surface covered with fixed chemical substrate sites as its legs bind to and unbind from the sites

tracks [2,17–24]. Such systems can be viewed from the nanotechnology perspective as cargo-transport elements for future systems, such as drug delivery systems that can walk over a cell surface and sense and analyze disease markers expressed on it. From a basic-science perspective, they can be useful as model systems for the study of alternative cell chemistries and origins-of-life scenarios, given that they are much simpler than the protein motors found in nature today. Among these are *molecular spiders*, DNA-based autonomous synthetic molecular motors [1,2].

A molecular spider consists of an inert body to which are attached flexible enzymatic legs (Fig. 1, left). We have reported spiders with up to six legs, using a streptavidin or streptavidin dimer scaffold for the body [1]. Each leg is a deoxyribozyme—an enzymatic sequence of single-stranded DNA that can bind to and cleave a complementary strand of a DNA substrate. The hip joint between the body and a leg is a flexible biotin linkage. When a molecular spider is placed on a surface coated with the single-stranded DNA substrate, its legs bind to the substrate. A bound leg can either detach from the substrate without modifying it, or it can catalyze the cleavage of the substrate, creating two product strands. The cleavage occurs at a designed ribonucleobase position within the otherwise DNA substrate. (The 8–17 enzyme we use for the legs was originally selected to cleave RNA [25].) Upon cleavage the two product strands eventually dissociate from the enzyme leg. The "lower" product remains bound to the surface. Because the lower product is complementary to the lower part of the spider's leg, there is a residual binding of the leg to the product; this binding is typically much weaker than the leg-substrate binding and thus much shorter-lived. The "upper" product remains in solution, so there can be some product rebinding. In laboratory experiments this effect is minimized with a flow setup; in our model, we neglect it.

Surface-plasmon-resonance experiments [1] show that a spider moves in a highly processive manner, cleaving thousands of substrates before eventually detaching from the surface (Fig. 2). We conclude that it moves in the direction of fresh substrates, leaving behind a trail of products (Fig. 1). But experimental observation of the motion has been limited by the small scale of a spider; although atomic-force microscopy and single-molecule fluorescence studies have been used to great effect [2], it has not yet been possible to establish the spider's movement and gait with certainty, nor to track the substrate cleavage site-by-site in real time. It has

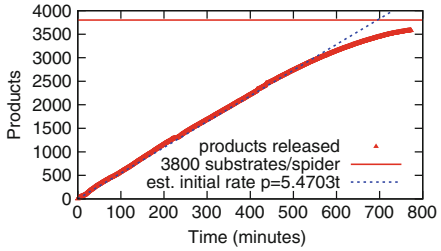

Fig. 2 Sensorgram for a no-flow SPR experiment: at high dilutions (1:3800) a four-legged spider cleaves 100 % of the substrate offered and shows an initial linear increase in the amount of released product (trend line drawn through the first 2,500 cleavage events). Redrawn from [1]

therefore been necessary to approach the problem using detailed modeling studies, through mathematical analysis and computer-based simulations [5–8, 26–28].

Molecular spiders, viewed as random walkers, have been modeled at different levels of abstraction, and with various parameter settings. Their asymptotic behavior is diffusive, just as with ordinary random walkers. However, and more important for the connection with laboratory experiments, superdiffusive behavior is observed in the transient, and it lasts for significant amounts of time, over which a spider covers significant distances. We showed [7] that in the presence of a residence-time bias between substrates and products this behavior can be explained by the spider's switching between two states—being on the boundary of the fresh substrates and being in the sea of already cleaved products. A spider on the boundary extracts chemical energy from the landscape, moves preferentially towards fresh substrates, and thus carries the boundary along. A spider that has stepped back into the products wanders aimlessly, i.e., diffuses.

Depending on the level of detail captured by the model, many kinetic parameters can be used to characterize the motion [8]. The basic parameter, however, already present in the most abstract model [5], is the chemical kinetic ratio r mentioned above. Because all our models are at heart continuous-time Markov processes, r is defined as the ratio between the transition rate out of a leg-on-substrate state and that out of a leg-on-product state. In this paper, we only use this abstract model, and so r is the only kinetic parameter.[1]

Once we have characterized how well the spiders are able to *walk*, the question becomes what tasks spiders can usefully *do*, such as carrying cargo molecules, following predesigned tracks, or searching for targets in unstructured landscapes. The latter is the topic of the present study. We describe models of future experiments wherein several identical molecular spiders will search for multiple specific target sites located on a finite surface made using the DNA origami technique. We are interested in the time it takes for *all* the spiders to find their targets.

[1]Elsewhere we study additional kinetic details using more elaborate and more computationally expensive models [8]. These models do permit useful characterization of mechanical motor properties, but they do not alter the basic walking behavior.

3 Abstract Model of Molecular Spider Motion and Search

Our model is a natural extension of the Antal–Krapivsky model [5, 6], modified to account for the movement of multiple spiders in two dimensions.

3.1 Motion

In our model, a spider walks on a two-dimensional rectangular lattice representing the chemical sites (initially substrates). There is complete exclusion: when one of the spider's k legs detaches from a site, it moves to an unoccupied site in a certain neighborhood of its current location; sites occupied by another leg (whether of the same spider or of a different one) are excluded. The concrete definition of a neighborhood is different for different instances of the model, representing different spider gaits. Figure 3 illustrates the neighborhoods n we use in our model instances: a black circle is the current leg site and the surrounding white circles are sites accessible within one step. Because any leg can rebind immediately to the site which it just left, the current site is also considered to be a part of the neighborhood.

Another constraint on the leg movement is the maximum distance S between any two legs of a spider. The distance S can be measured differently in different models. In the following, we use two distance metrics, the Manhattan (L_1) distance S_m and the Chebyshev (L_∞; maximum) distance S_z.

Together, the parameters n, S, and k define the gait of a spider. Within the constraints of the gait, a spider's motion is governed by the chemical kinetics of its legs, which we model as a continuous-time Markov process. Each leg independently interacts with the chemical site it is on; at the high level of abstraction of this model, the interaction is completely described by a single transition rate. A leg detaches at rate one from a product, and at rate r from a substrate, where $r \leq 1$. When a leg leaves a substrate, that substrate is transformed into a product. At the high level of abstraction of the model, the reattachment of the leg is instantaneous. There is no directional bias in the model: if in the current state there are several moves that do not violate any restrictions, the leg that is moving chooses any one of them with equal probability.

3.2 Search

In our search model, the lattice is of a finite fixed size, 22 by 32; the numbers may seem arbitrary, but they reasonably describe the DNA-origami tiles used in past molecular spider experiments [2]. We use three searching spiders, initially in one corner of the lattice. The search targets are the three special *trap* sites, in the three opposite corners. We assume that a leg that attaches to a trap remains forever bound

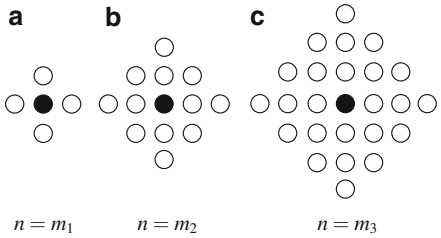

Fig. 3 Neighborhoods studied. By exploring a variety of neighborhood sizes and shapes in this abstract model, we hope to guide the choice of parameters for physical spiders in future laboratory experiments, primarily the leg length (which can be adjusted using spacers)

to it.[2] Furthermore, when a spider's leg is thus trapped, *all* its legs cease moving. All remaining sites initially are ordinary cleavable substrates. Because there are as many targets as there are spiders, eventually each spider reaches a target. When all target sites have been found all motion stops (in this crude abstraction).

The chemical kinetics parameter r and the spiders' gait parameters n, S, and k are the variable parameters of the model, and they will influence how fast the spiders move and search for targets. In the following section we begin our exploration of this parameter space.

3.3 Interaction

In the model, spiders influence each other's motion in two ways. First, through the exclusion principle: if the legs of one spider occupy some set of sites, then another spider's legs are prevented from occupying any of those sites. Second, through stigmergy, or communication via modification of the environment: when one spider makes several moves, it leaves behind a trail of products; another spider can subsequently encounter those products on its way and they will affect its future moves.

In 1D when one of a spider's legs is attached to a substrate while its remaining legs are attached to products, in other words when the spider is located on a boundary between substrates and products, it will move with a bias towards substrates [6]. A similar bias will occur in 2D, and thus when a spider encounters products (its own trace or other spiders' traces), it will more likely move towards fresh substrates. The boundary in 2D can have much more different shapes than in 1D, and sometimes this might lead to a bias in a wrong direction. For example, just

[2]In the laboratory, an uncleavable, pure-DNA substrate has been used [2] for the purpose. In envisaged applications, the targets presented on the cell surface will not necessarily be DNA strands. To bind to non-DNA targets, in addition to the legs a spider may carry an "arm," an aptamer molecule that specifically binds to the target.

several substrates sites can mislead a spider into a big sea of products. However, we expect such scenarios to be rare and in most cases the bias to be in a direction of truly unexplored areas of the surface. There is thus a repulsive stigmergic interaction: we expect spiders preferentially to avoid the traces of other spiders.

In consequence, in a target search application we expect the spiders to preferentially search for targets in new, unvisited locations. To evaluate the importance of exclusion and stigmergy for the search task we compare the performance of the model with two alternative modifications. In the first modification we make spiders move on separate and initially identical surfaces, i.e., each spider has its own surface and they cannot see one another. However, they still share the same trap sites, so if one spider finds a particular trap site, that site will appear as occupied for all other spiders. This modification to the original model removes both exclusion and stigmergy from the model and allows us to evaluate how fast these *independent* spiders can find all traps. In the second modification, all spiders move on the same surface, but each surface site contains three different chemical components, keyed to the three spiders. When a leg of a particular spider a leaves some site x it affects only the component x_a of site x corresponding to a: if x_a was a substrate it is converted into a product. Thus, in this modification, a spider can see only its own traces and has no access to other spiders' traces. While a site x is occupied by spider a, the other two spiders are barred from it. Thus, this modification of the model eliminates stigmergic communication between spiders but preserves the exclusion principle, allowing us to evaluate the importance of stigmergy alone.

3.4 Model and Simulation

Combining the states of all the spiders and the surface gives a continuous-time Markov process for our model. We use the Kinetic Monte Carlo method [29] to simulate multiple trajectories of this Markov process. The simulation stops when all spiders are trapped, i.e., the search is complete, and then the simulated time is recorded. This time is an observation of a first-passage-time random variable. The results below will show the mean first passage time estimated from our traces.

4 Simulation Results

We carried out simulations using the following seven spider gaits: (a) $k = 2$, $S_z = 1$, $n = m_1$; (b) $k = 2$, $S_m = 2$, $n = m_3$; (c) $k = 2$, $S_m = 3$, $n = m_2$; (d) $k = 3$, $S_m = 2$, $n = m_3$; (e) $k = 3$, $S_m = 3$, $n = m_2$; (f) $k = 4$, $S_m = 2$, $n = m_3$; (g) $k = 4$, $S_m = 3$, $n = m_2$; and a simple random walker, which can be viewed as a spider with parameters (h) $k = 1$, $S_m = 1$, $n = m_1$. These gaits were

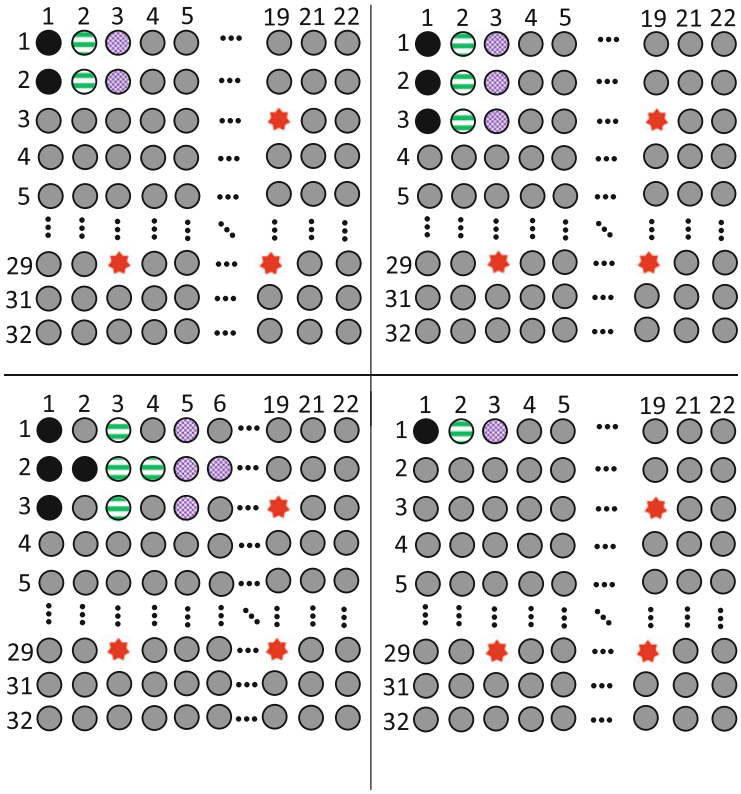

Fig. 4 The initial state of the system and the spider gait for each simulated configuration. Configurations (**a**)–(**g**) are multi-legged spiders and configuration (**h**) is the control one-legged spider. (**a**) $k = 2$, $S_z = 1$, $n = m_1$ (**b**) $k = 2$, $S_m = 2$, $n = m_3$ (**c**) $k = 2$, $S_m = 3$, $n = m_2$ (**f**) $k = 4$, $S_m = 2$, $n = m_3$ (**g**) $k = 4$, $S_m = 3$, $n = m_2$ (**h**) $k = 1$, $S_m = 1$, $n = m_1$

chosen to correspond to different physically realistic molecular spiders, but it must be admitted that the space of plausible possible gaits is much larger, and we must defer its exploration to a future study. Figure 4 describes the chosen gaits, along with the initial spider positions and the target site positions, shown graphically. Initial positions for the gaits with equal number of legs k are equivalent, and thus Fig. 4 groups the gaits by k.

Solid black, striped, and dotted circles represent the initial leg positions of the first, second, and third spiders, respectively; solid gray circles represent ordinary substrates; stars represent the three target traps. The targets are non-specific, that is, any spider can be trapped by any target.

Simulation results are shown comprehensively in Fig. 5, and also individually for each gait in Fig. 6 to reveal additional detail. Figure 6 also contains results for non-communicating spiders with and without exclusion.

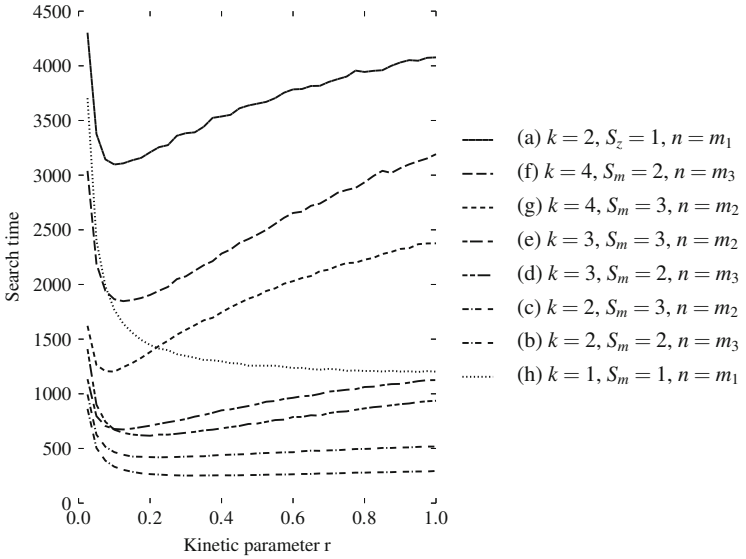

Fig. 5 Search time as a function of the kinetic parameter r for the eight configurations simulated. (**a**)–(**g**) are multi-legged spiders; (**h**) is a one-legged spider

4.1 The Effect of the Gait

In Fig. 5 curves corresponding to the studied gaits have different shapes and are vertically separated. Thus, the gaits of the spiders greatly influence their performance, and spiders with particular gaits can be faster than regular random walkers. For the simulated surface, spiders with gaits that allow more freedom for the legs to move (i.e., when a leg is moving to a new site it has more sites to choose from) achieve better performance. Two- and three-legged spiders with gaits (b), (c), (d), and (e) are the fastest among those simulated; they have either the larger neighborhood $n = m_3$ or the longest possible distance between legs $S_m = 3$, which makes these gaits the least restrictive. Spiders with gait (a) $k = 2$, $S_z = 1$, $n = m_1$ are the slowest, despite having the same number of legs $k = 2$ as the best performing spiders, with gait (c): gait (a) is much more restrictive, and with its parameters $S_z = 1$ and $n = m_1$ it gives a small candidate set of new positions for the legs. Gaits (f) $k = 4$, $S_m = 2$, $n = m_3$ and (g) $k = 4$, $S_m = 3$, $n = m_2$ are also slower than gaits (b) through (e): they have the same parameters S_m and n, but the addition of an extra leg reduces the choice of sites for a moving leg (because the new position must be within a certain distance from each of the legs that remains attached), which leads to slower performance.

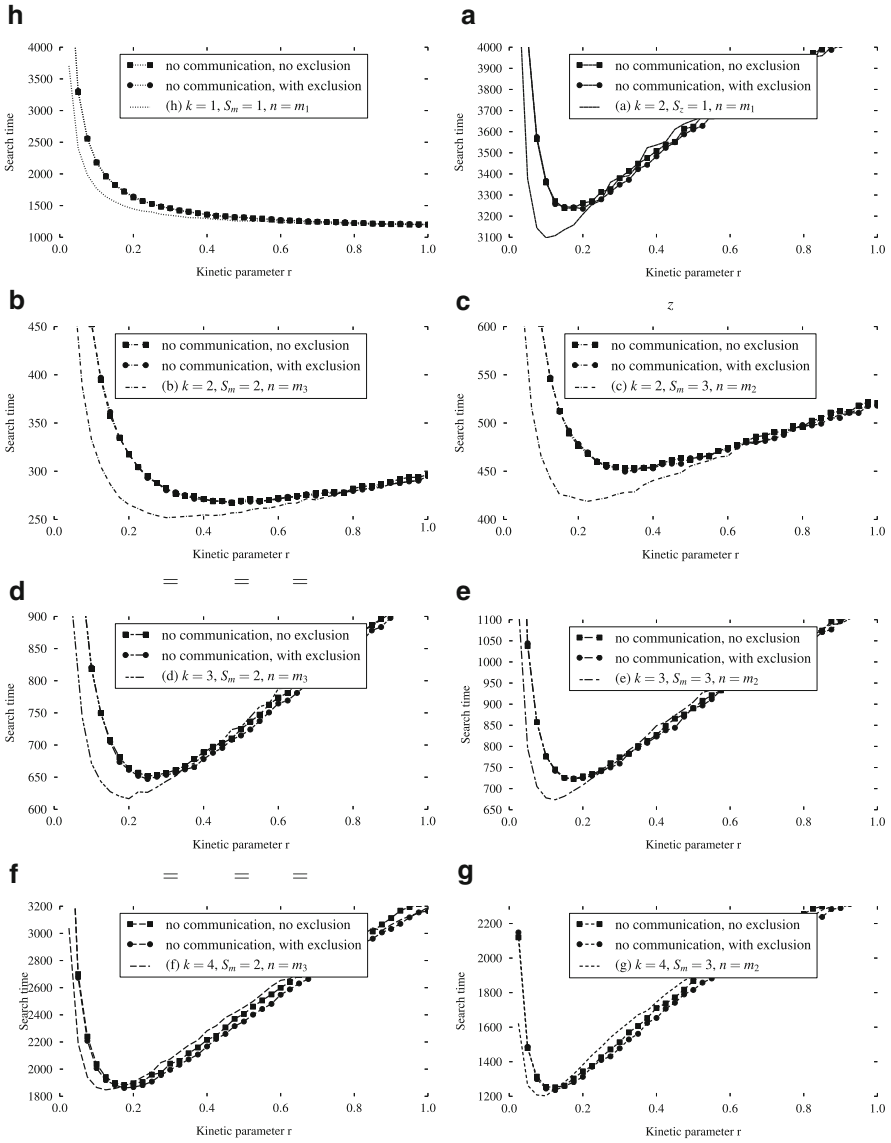

Fig. 6 Enlarged plots from Fig. 5, in which the existence of an optimum r value for the multi-legged spiders can be discerned better. Also shown in each plot is a comparison with the two model modifications, non-communicating spiders with and without exclusion. (**h**) $k = 1$, $S_m = 1$, $n = m_1$ (**a**) $k = 2$, $S_z = 1$, $n = m_1$ (**b**) $k = 2$, $S_m = 2$, $n = m_3$ (**c**) $k = 2$, $S_m = 3$, $n = m_2$ (**d**) $k = 3$, $S_m = 2$, $n = m_3$ (**e**) $k = 3$, $S_m = 3$, $n = m_2$ (**f**) $k = 4$, $S_m = 2$, $n = m_3$ (**g**) $k = 4$, $S_m = 3$, $n = m_2$

4.2 The Effect of the Kinetics

We now examine the influence of the kinetics (i.e., the difference between the substrates and the products displayed on the surface, which is amenable to adjustment in the laboratory) on the search performance of the spiders. We used 40 different values of the chemical kinetics parameter r in the range from 0.025 (heavy substrate-product bias) to 1.0 (no bias) in increments of 0.025. Results are shown in the eight panels of Fig. 6. The kinetic parameter r significantly affects the performance of the spiders. One-legged spiders always have better performance when r is bigger—this observation is similar to the results of the study of one-legged spiders in one dimension [6]. Thus, the presence of memory on the surface in the form of substrates and products does not improve the performance of a monovalent random walker. For multi-legged spiders each gait has an optimal r value that minimizes the search time; this behavior is similar to two-legged spiders in one dimension [7].

In previous work [7] we found that when $r < 1$, spider ensembles go through three different regimes of motion—initially, spiders move slowly; then they start moving faster and achieve better performance than regular diffusion; and, finally, in the time limit they slow down and move as regular diffusion. For lower r values the initial slow period is longer than for higher r values, but then later the superdiffusive period is longer and faster for smaller r values. For travel over shorter distances the length of the initial period is more important than for longer distances and thus smaller r values can result in higher first passage times. For travel over longer distances the superdiffusive period is important and smaller r values give better results. Thus for every particular distance there is an optimum value r that minimizes the mean first passage time. Figure 7 shows the mean first passage time for a single two-legged spider in one dimension. For example, for the distance of 50 sites the spider with $r = 0.1$ is faster than the other (sampled) r values; but for the distance of 20 sites the $r = 0.5$ spider is faster. Also, although spiders with $r = 0.01$ and $r = 0.005$ are the slowest for distances of 50 sites and less, they eventually overtake all other simulated spiders at greater distances. Thus, similarly to these results for two-legged spiders in one dimension [7], we speculate that for bigger search lattices the optimum r values will decrease.

4.3 The Effect of Spider Interactions (Exclusion and Stigmergy)

Figure 6 shows that for all gaits except (h) (random walk) spiders with stigmergy perform better than non-communicating spiders when r is around its optimum value. As expected, for values of r closer to 1.0, the role of stigmergic communication is less significant, because the difference between visited and unvisited sites is small, and thus the traces of spiders influence their behavior only weakly. For the values of r between the optimum and 1.0, the difference between communicating and non-communicating spiders depends on the spiders' gaits. For the best performing

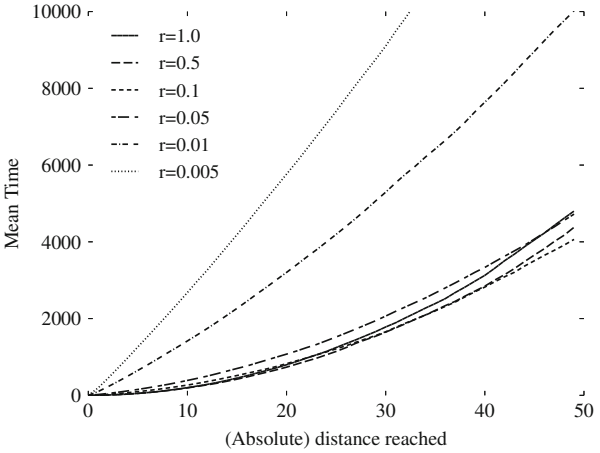

Fig. 7 Mean first passage time of a single two-legged spider moving over an infinite one-dimensional track initially covered with substrates

gaits (b) and (c), the advantage of stigmergic communication grows as r decreases from 1.0 to its optimum value. For the gait (h) (random walk) the advantage of communication also grows as r decreases, while the search time of both communicating and non-communicating spiders grows. This happens because the presence of substrates only serves to slow the random spiders down, and non-communicating random spiders initially have more substrates in total (for non-communicating spiders there is initially a field of substrates per spider, while communicating spiders share just one field). Thus, non-communicating spiders use more time to clear the substrates off their surfaces. For the gaits (d)–(g), as r decreases from 1.0 to its optimum value, the difference in search time between communicating and non-communicating spiders remains very small. For those gaits the advantage of stigmergic communication begins to emerge only when r is close to its optimum value.

Comparison of non-communicating spiders with and without exclusion shows that the influence of exclusion on search time is less significant than the influence of communication. The effect of exclusion also depends on the number of legs. For gaits with one and two legs, the difference in search time between spiders with and without exclusion is very small. For bigger spiders, with three and four legs, this difference grows, and spiders with exclusion are a little faster than spiders without exclusion. In fact, for spiders with three and four legs and a range of r values between the optimum and 1.0, spiders with exclusion and without communication perform the best, by a small margin. As with regular random walkers this happens because non-communicating spiders effectively have more substrates initially, but in contrast to regular random walkers the substrates make them faster, and this improves the search performance. But even for those gaits, communicating spiders with exclusion overtake non-communicating spiders as r approaches its optimum value.

5 Discussion

The simulations were performed for a relatively small lattice and the gaits that are currently the fastest may not turn out to be so for bigger lattices. Also, it appears that r values have more influence on search time for some gaits than for others: gaits (f) $k = 4$, $S_m = 2$, $n = m_3$ and (g) $k = 4$, $S_m = 3$, $n = m_2$ get the most improvement in performance when r decreases from one to its optimal value, and gaits (b) $k = 2$, $S_m = 2$, $n = m_3$ and (c) $k = 2$, $S_m = 3$, $n = m_2$ get the least improvement. To understand this dependence, in future work we shall study gaits more systematically, i.e., have a scenario to vary the parameters k, S, and n.

Search performance can also be affected by the initial conditions we are using, and different initial condition may result in different optimal gaits and different optimal r values. To account for these effects, in future work we shall vary the placement of the targets and the spider starting positions, as well as the number of targets and spiders.

A number of assumptions had to be made to reduce the complex interactions of physical molecular spiders on DNA origami tiles, which have not yet been fully experimentally characterized and understood, to a tractable mathematical model amenable to efficient computer simulation. The highly abstract model presented here may sacrifice too much physical realism; at the opposite end of the abstraction spectrum are molecular dynamics approaches, but those are infeasible at the space and time scales of interest. Laboratory experiments can reveal the ground truth, but are too expensive for a full exploration of the parameter space. Instead, we plan to use mesoscale models [8] to refine these initial results.

Our system can be viewed as a hierarchical multi-agent system: the system consists of multiple spiders, and each spider consists of multiple legs. The legs and the spiders interact through exclusion, and also stigmergically as they modify the surface; and the legs of one spider interact through kinematic constraints. But we are not free to design this multi-agent system as we please to achieve some system-level behavior; instead, we are severely restricted in the design of the agents: they are just molecules of a particular kind, not intelligent agents. Thus, recalling the principle of complexity theory that simple local rules, iterated, may give rise to complex global behaviors, we are asking whether this also happens in a very primitive setting. We do not expect to be able to mimic the complexity of behaviors of even a single ant, which after all is billions times more structurally complex. However, we hope that our results will aid in the development of nanoscale molecular walkers. In future, we plan to continue the study of searching behaviors and to initiate a study of additional biochemically plausible modes of spider–spider interaction.

Acknowledgments We thank Paul L. Krapivsky and Mark J. Olah for many discussions of spider behaviors. A preliminary version of this work was presented at WIVACE 2012; we thank the conference reviewers for their incisive and detailed comments. This material is based upon work supported by the National Science Foundation under grants 0829896 and 1028238.

References

1. Pei, R., Taylor, S.K., Stefanovic, D., Rudchenko, S., Mitchell, T.E., Stojanovic, M.N.: Behavior of polycatalytic assemblies in a substrate-displaying matrix. J. Am. Chem. Soc. **128**(39), 12693–12699 (2006)
2. Lund, K., Manzo, A.J., Dabby, N., Michelotti, N., Johnson-Buck, A., Nangreave, J., Taylor, S., Pei, R., Stojanovic, M.N., Walter, N.G., Winfree, E., Yan, H.: Molecular robots guided by prescriptive landscapes. Nature **465**, 206–210 (2010)
3. Rothemund, P.W.K.: Folding DNA to create nanoscale shapes and patterns. Nature **440**, 297–302 (2006)
4. Pinheiro, A.V., Han, D., Shih, W.M., Yan, H.: Challenges and opportunities for structural DNA nanotechnology. Nat Nanotechnol. **6**, 763–772 (2011)
5. Antal, T., Krapivsky, P.L., Mallick, K.: Molecular spiders in one dimension. J. Stat. Mech. Theor. Exp. **2007**(08), P08027 (2007)
6. Antal, T., Krapivsky, P.L.: Molecular spiders with memory. Phys. Rev. E **76**(2), 021121 (2007)
7. Semenov, O., Olah, M.J., Stefanovic, D.: Mechanism of diffusive transport in molecular spider models. Phys. Rev. E **83**, 021117 (2011)
8. Olah, M.J., Stefanovic, D.: Multivalent random walkers: a model for deoxyribozyme walkers. In: DNA 17: Proceedings of the Seventeenth International Meeting on DNA Computing and Molecular Programming. Lecture Notes in Computer Science, vol. 6397, pp. 160–174. Springer, Berlin (2011)
9. Kolomeisky, A.B., Fisher, M.E.: Molecular motors: a theorist's perspective. Ann. Rev. Phys. Chem. **58**, 675–695 (2007)
10. Bier, M.: The energetics, chemistry, and mechanics of a processive motor protein. BioSystems **93**, 23–28 (2008)
11. Astumian, R.D.: Thermodynamics and kinetics of molecular motors. Biophys. J. **98**, 2401–2409 (2010)
12. Jamison, D.K., Driver, J.W., Rogers, A.R., Constantinou, P.E., Diehl, M.R.: Two kinesins transport cargo primarily via the action of one motor: implications for intracellular transport. Biophys. J. **99**, 2967–2977 (2010)
13. Lipowsky, R., Beeg, J., Dimova, R., Klumpp, S., Müller, M.K.I.: Cooperative behavior of molecular motors: cargo transport and traffic phenomena. Phys. E **42**, 649–661 (2010)
14. Driver, J.W., Jamison, D.K., Uppulury, K., Rogers, A.R., Kolomeisky, A.B., Diehl, M.R.: Productive cooperation among processive motors depends inversely on their mechanochemical efficiency. Biophys. J. **101**, 386–395 (2011)
15. Kay, E.R., Leigh, D.A., Zerbetto, F.: Synthetic molecular motors and mechanical machines. Angew. Chem. Int. Edit. **46**, 72–191 (2007)
16. Hugel, T., Lumme, C.: Bio-inspired novel design principles for artificial molecular motors. Curr. Opin. Biotech. **21**(5), 683–689 (2010)
17. Yurke, B., Turberfield, A.J., Mills, A.P., Jr., Simmel, F.C., Neumann, J.L.: A DNA-fuelled molecular machine made of DNA. Nature **406**, 605–608 (2000)
18. Shirai, Y., Osgood, A.J., Zhao, Y., Kelly, K.F., Tour, J.M.: Directional control in thermally driven single-molecule nanocars. Nano Lett. **5**(11), 2330–2334 (2005)
19. Tian, Y., He, Y., Chen, Y., Yin, P., Mao, C.: A DNAzyme that walks processively and autonomously along a one-dimensional track. Angew. Chem. Int. Edit. **44**, 4355–4358 (2005)
20. Venkataraman, S., Dirks, R.M., Rothemund, P.W.K., Winfree, E., Pierce, N.A.: An autonomous polymerization motor powered by DNA hybridization. Nature Nanotechnology **2**, 490–494 (2007)
21. Green, S.J. Bath, J., Turberfield, A.J.: Coordinated chemomechanical cycles: a mechanism for autonomous molecular motion. Phys. Rev. Lett. **101**, 238101 (2008)
22. Omabegho, T., Sha, R., Seeman, N.C.: A bipedal DNA brownian motor with coordinated legs. Science **324**, 67–71 (2009)

23. Bath, J., Green, S.J., Allen, K.E., Turberfield, A.J.: Mechanism for a directional, processive, and reversible DNA motor. Small **5**(13), 1513–1516 (2009)
24. Gu, H., Chao, J., Xiao, S.-J., Seeman, N.C.: A proximity-based programmable DNA nanoscale assembly line. Nature **465**, 202–206 (2010)
25. Santoro, S.W., Joyce, G.F.: A general purpose RNA-cleaving DNA enzyme. P. Natl. Acad. Sci. USA **94**, 4262–4266 (1997)
26. Samii, L., Linke, H., Zuckermann, M.J., Forde, N.R.: Biased motion and molecular motor properties of bipedal spiders. Phys. Rev. E **81**, 021106 (2010)
27. Semenov, O., Olah, M.J., Stefanovic, D.: Multiple molecular spiders with a single localized source: the one-dimensional case. In DNA 17: Proceedings of the Seventeenth International Meeting on DNA Computing and Molecular Programming. Lecture Notes in Computer Science, vol. 6397, pp. 204–216. Springer, Berlin (2011)
28. Samii, L., Blab, G.A., Bromley, E.H.C., Linke, H., Curmi, P.M.G., Zuckermann, M.J., Forde, N.R.: Time-dependent motor properties of multipedal molecular spiders. Phys. Rev. E **84**, 031111 (2011)
29. Bortz, A.B., Kalos, M.H., Lebowitz, J.L.: A new algorithm for Monte Carlo simulation of Ising spin systems. J. Comput. Phys. **17**(1), 10–18 (1975)

Towards the Use of Genetic Programming for the Prediction of Survival in Cancer

Marco Giacobini, Paolo Provero, Leonardo Vanneschi, and Giancarlo Mauri

Abstract Risk stratification of cancer patients, that is the prediction of the outcome of the pathology on an individual basis, is a key ingredient in making therapeutic decisions. In recent years, the use of gene expression profiling in combination with the clinical and histological criteria traditionally used in such a prediction has been successfully introduced. Sets of genes whose expression values in a tumor can be used to predict the outcome of the pathology (gene expression signatures) were introduced and tested by many research groups. A well-known such signature is the 70-genes signature, on which we recently tested several machine learning techniques in order to maximize its predictive power. Genetic Programming (GP) was shown to perform significantly better than other techniques including Support Vector Machines, Multilayer Perceptrons, and Random Forests in classifying patients. Genetic Programming has the further advantage, with respect to other methods, of performing an automatic feature selection. Importantly, by using a weighted average between false positives and false negatives in the definition of the fitness, we showed that GP can outperform all the other methods in minimizing false

M. Giacobini (✉)
Computational Epidemiology Group, Department of Veterinary Sciences, and Complex Systems Unit, Molecular Biotechnology Center, University of Torino, Italy
e-mail: mario.giacobini@unito.it

P. Provero
Molecular Biotechnology Center, University of Torino, Italy
e-mail: paolo.provero@unito.it

L. Vanneschi
ISEGI, Universidade Nova de Lisboa, 1070-312 Lisboa, Portugal

DISCo, University of Milano-Bicocca, 20126 Milan, Italy
e-mail: lvanneschi@isegi.unl.pt; vanneschi@disco.unimib.it

G. Mauri
DISCo, University of Milano-Bicocca, 20126 Milan, Italy
e-mail: mauri@disco.unimib.it

S. Cagnoni et al. (eds.), *Evolution, Complexity and Artificial Life*,
DOI 10.1007/978-3-642-37577-4_12,

negatives (one of the main goals in clinical applications) without compromising the overall minimization of incorrectly classified instances. The solutions returned by GP are appealing also from a clinical point of view, being simple, easy to understand, and built out of a rather limited subset of the available features.

1 Introduction

The side effects of cancer systemic therapies are extremely debilitating for patients, so that it is extremely important to be able to match type and dosage of the therapy to each individual patient based on his/her risk of relapse. Accurate methods of risk stratification, i.e. the prediction of the outcome of the disease, could spare many patients such side effects.

Patient classification is mostly based on clinical and histological parameters. The advent of gene expression profiling techniques has led, in the last decade, to a large body of research on the use of transcriptomics to better understand the molecular bases of cancer, on the one hand, and to improve patient classification, on the other.

A gene signature is defined as a set of genes whose expression profiles can be used to discriminate between biological states (see [1]): in the case of cancer, they have been used to distinguish cancerous from non-cancerous conditions and to classify cancer patients based on their risk of relapse.

A much larger body of research has been devoted to the problem of identifying gene signatures in various types of cancer rather than to the question of which algorithms can be used to maximize their predictive power. Machine learning methods [2] have been applied to the analysis of gene expression profiling datasets, including k-nearest neighbors [3], hierarchical clustering [4], self-organizing maps [5], Support Vector Machines [6, 7], or Bayesian networks [8]. In the last few years also Evolutionary Algorithms [9] found application to gene expression analysis, both for feature selection and for classification. In particular, Genetic Algorithms [10] were used to build selectors in which each allele of the representation corresponds to one gene and its state denotes whether the gene is selected or not [11]. Genetic Programming (GP), on the other hand, was shown to work well for recognizing structures in large datasets [12]. GP was applied to gene expression data to generate programs able to reliably predict the health/malignancy states of tissues, or to discriminate between different tissues. GP automatically selects a small number of feature genes during its evolution [13], so that the evolution of classifiers from the initial population integrates gene selection within classifier construction. For example, GP was applied by [14] to cancer gene expression to select relevant feature genes to include in molecular classifiers of tumor samples. Furthermore, GP has been successfully used to discover rule-based, easy-to-interpret classifiers from medical [15] and gene expression data [16].

We have recently compared the effectiveness of various machine learning approaches for risk stratification in cancer [17]. Such a comparison was based on the well-established "70-genes signature," a list of genes identified by [18] based

on correlation with survival on a large cohort of patients. This set of genes provides the basis for the breast cancer molecular prognostic test "MammaPrint"™ and is described in Sect. 2. In [17] both the outcome of the disease (survival status at a given endpoint) and of the gene expression profiles were preprocessed so as to transform them into binary variables, since this is the type of variables on which GP works in the simplest and most natural way by combining them using logical operators. Indeed many GP benchmarks such as the even parity or the multiplexer problems [19] are built in this way. Moreover, the solutions found by such GP algorithms are in general quite easy to interpret from a biological viewpoint.

In order to minimize possible biases we used default implementations and parameters of all machine learning methods, and the results we obtained in [17] showed that GP clearly outperforms all other techniques. All methods other than GP, on the other hand, showed comparable performance, which indicated GP as the most promising method. When we compared the best solution found by GP to the original scoring method proposed in [18, 20] we obtained a rather small difference, below statistical significance. As expected, due to its construction, the scoring method was superior to all machine learning algorithms in minimizing false negatives, which is a very important task in clinical applications: indeed, a false negative in the prediction algorithm could cause in principle the withdrawal of therapy from a patient who would actually benefit from it, an error that is naturally considered worse than the opposite one.

Nevertheless, GP showed some interesting aspects. For instance, it generated solutions based on a small number of genes as a result of a strong automatic feature selection. This added value of the GP approach, together with the promising performances shown by GP, motivated us to further pursue this line of research along the lines described in the present work.

First, we abandoned the binarization of the expression data performed in [17] to avoid the inherent loss of information: we used instead the original floating point valued expression profiles of the 70-genes dataset. Moreover, we introduced a parameterization of the fitness function so that false negatives can be penalized differently from false positives, making the classifier tunable in terms of sensitivity and specificity.

2 The 70-Genes Signature Dataset

We used the cancer gene expression profiling data of [20] ("NKI dataset"), which includes gene expression and survival data for 295 young breast carcinoma patients who did not undergo systemic treatment. We limited the analysis to the genes included in the "70-genes signature" [18]. The targets were the survival data transformed into an outcome in binary form, namely the survival status of the patient at the fixed endpoint time $t_{\text{end}} = 10.3$ years, chosen so as to balance the number of deceased and surviving patients: exactly half of the 148 patients for which the status at t_{end} is known had deceased.

Our dataset is represented by a matrix $H = [H_{(i,j)}]$ with 148 rows (instances) and 71 columns (features). Each line i represents the gene expression profile of a patient given by a vector $[H_{(i,1)}\ H_{(i,2)}\ \ldots\ H_{(i,70)}]$ of 70 floating point numbers, plus one variable expressing representing the target (0 = alive after t_{end} years, 1 = dead of breast cancer before t_{end} years), placed at position $H_{(i,71)}$.

Our purpose was then to generate a binary classification model F such that

$$F(H_{(i,1)}, H_{(i,2)}, \ldots, H_{(i,70)}) = H_{(i,71)}$$

for each line i in the dataset. For each machine learning method we performed 50 independent runs. In each run, the dataset was randomly split into a training set (70 % of the patients) and a test set (30 %).

3 Genetic Programming

We used tree-based genetic programming (GP) [19, 21], in which solutions are generated from a set of terminal symbols $\mathscr{T}$ and a set of functionals (internal nodes) $\mathscr{F}$. The set of terminals $\mathscr{T}$ included 70 floating point variables (i.e., one for each feature of our dataset). GP individuals were generated using the set of functions $\mathscr{F} = \{\texttt{plus}, \texttt{minus}, \texttt{mul}, \texttt{div}, \texttt{squaredsin}\}$, where `plus`, `minus`, and `mul` are the usual sum, subtraction, and multiplication, while `div` is the protected division [19] and `squaredsin` is the square of the sine trigonometric function.

GP individuals thus return a floating point value that is then turned into a binary classifier using a threshold value, which we defined as the average between the maximum and minimum values in the whole training set.

Initially, we adopted as fitness function the total number of incorrectly classified instances, thus turning the problem into a minimization one (lower values are better). Successively we modified the fitness function by introducing the possibility of weighing differently false positives and false negatives. In other words, the new fitness function was: $\alpha\ FN + \beta\ FP$, where α and β are two floating point values included in the range [0, 1], such that $\alpha + \beta = 1$. The tuning of α and β can be used to privilege sensitivity over specificity or vice versa.

Data from our dataset were fed to the GP without filtering nor preprocessing, to avoid interfering with GP's ability to automatically perform an implicit feature selection. The mechanism by which GP automatically selects features is simple [13, 22–24]: the search space explored by GP includes not only expressions using all the 70 variables but also expressions using only subsets of them. If expressions which use a smaller number of features have a better fitness, they survive into the new population, since fitness is the only principle used by GP to select solutions.

The parameters used by our implementation of GP and by the other machine learning methods are reported in Table 1. There is no special justification for the choice of those parameter values, if not the fact that they are standard for the

Table 1 Parameters used in the experiments

GP Parameters	
Population size	500 individuals
Population initialization	Ramped half and half [19]
Selection method	Tournament (tournament size = 10)
Crossover rate	0.9
Mutation rate	0.1
Maximum number of generations	5
Algorithm	Generational tree-based GP with no elitism
SVM Parameters	
Complexity parameter	0.1
Size of the kernel cache	10^7
Epsilon value for the round-off error	10^{-12}
Exponent for the polynomial kernel	1.0
Tolerance parameter	0.001
Multilayer Perceptron Parameters	
Learning algorithm	Back-propagation
Learning rate	0.03
Activation function for all the neurons in the net	Sigmoid
Momentum	0.2 progressively decreasing until 0.0001
Hidden layers	(Number of attributes + number of classes)/2
Number of epochs of training	500
Voted Perceptron Parameters	
Exponent for the polynomial kernel	1.0
Maximum number of alterations to the perceptron	10,000
Radial Basis Function Network Parameters	
Minimum standard deviation for the clusters	0.1
Number of clusters for K-means	2
Ridge value for the logistic	1.8×10^{-8}

computational tool we used, i.e. GPLab: a public domain GP system implemented in MatLab (for the GPLab software and documentation, see [25]). We purposely avoided all attempts at parameter optimization to prevent the introduction of biases that could undermine the credibility of the comparison between different methods.

4 Support Vector Machines

Support Vector Machines (SVM) are a set of related supervised learning methods used for classification and regression. They were originally introduced in [26] with the aim of devising a computationally efficient way of identifying separating

hyperplanes in a high-dimensional feature space. In particular, the method seeks separating hyperplanes that maximize the margin between sets of data. This should ensure a good generalization ability of the method, under the hypothesis that training and test data are drawn from the same distribution. To calculate the margin between data belonging to two different classes, two parallel hyperplanes are constructed, one on each side of the separating hyperplane, which are "pushed up against" the two datasets. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the neighboring data points of both classes since, in general, the larger the margin, the lower the generalization error of the classifier. The parameters of the maximum-margin hyperplane are derived by solving large quadratic programming (QP) optimization problems. There exist several specialized algorithms for quickly solving the problems that arise from SVMs, mostly reliant on heuristics for breaking the problem down into smaller, more manageable chunks. In this work we used the implementation of John Platt's [27] sequential minimal optimization (SMO) algorithm for training the support vector classifier. SMO works by breaking the large QP problem into a series of smaller two-dimensional sub-problems that may be solved analytically, eliminating the need for numerical optimization algorithms such as conjugate gradient methods. The implementation we used is the one contained in the Weka public domain software [28].

The main parameter values used in this work are reported in Table 1. All these parameter values correspond to the standard values set by the Weka software [28] as defined, for instance, in [27]. We are aware that, in several application domains, SVM have been shown to outperform competing techniques by using nonlinear kernels, which implicitly map the instances to very high (even infinite)-dimensional spaces. Unfortunately, the implementation of SVM in Weka sets as default the polynomial kernel with degree 1. This means that we evaluate the accuracy of linear SVM (SVM in the original feature space). Possible improvements, using more sophisticated kernel functions, will be included in our future work. By now we use this version of the SVM with the only goal of having an immediate experimental comparison with GP. Analogous considerations also hold for the other machine learning methods used, discussed below.

5 Multilayer Perceptrons

The multilayer Perceptron is a feed-forward artificial neural network model [29]. It modifies the standard Linear Perceptron by using three or more layers of neurons (nodes) with nonlinear activation functions, and is more powerful than the simple perceptron in that it can correctly classify data that are not linearly separable, i.e., separable by a hyperplane. It consists of an input and an output layer with one or more hidden layers of nonlinearly activated nodes. Each node in one layer connects with a certain weight to every other node in the following layer. The implementation we have adopted is the one included in the Weka software distribution [28]. We used the backpropagation learning algorithm [29] and the values used for all the

parameters are reported in Table 1. As for the previously discussed machine learning methods, also in the case of Multilayer Perceptron it is important to point out that we used a parameter setting as standard as possible, without doing any fine parameter tuning for this particular application. Our goal is, in fact, to compare different computational methods under standard conditions with GP and not to solve in the best possible way the specific problem. In particular, all the values reported in Table 1 correspond to the default ones adopted by the Weka software.

6 Voted Perceptron

The Voted Perceptron [30] is an artificial neural network which combines the Rosenblatt's Perceptron algorithm [29] with Helmbold and Warmuth's [31] leave-one-out method. Like Vapnik's maximal-margin classifier [26], this method takes advantage of data that are linearly separable with large margins. Compared to Vapnik's algorithm, however, it is simpler to implement, and usually more efficient in terms of computation time. It can also be efficiently used in very high-dimensional spaces using kernel functions. In particular, compared to the standard Rosenblatt's Perceptron algorithm, the Voted Perceptron stores more information during training. This information consists in the list of all prediction vectors that were generated after each and every mistake. This allows us to maintain an updated list of Perceptrons, labeled by the number of mistakes made during the training phase. At the end of the training phase, we are then able to perform the prediction by running all the obtained Perceptrons on the test data, and weighting their result using this label. Authors of [30] state that this elaborate information, consisting in more than one predictor and its relative importance or weight, allows one to generate better predictions on the test data compared to the simple Rosenblatt's Perceptron. In particular, for each prediction vector, the number of iterations it "survives"ï£¡ until the next mistake is made is counted (this count is often referred to as the "weight"ï£¡ of the prediction vector). To perform a classification, the binary prediction of each of the prediction vectors is computed and all these classifications are combined by a weighted majority vote. The weights used are the survival times described above. This makes intuitive sense as "good" prediction vectors tend to survive for a long time and thus may have better prediction ability, receiving larger weight in the majority vote. As for the other methods, the Weka standard implementation has been used, using the Weka default parameters reported in Table 1.

7 Radial Basis Function Network

A radial basis function network (RBFN) is an artificial neural network that uses a radial basis functions as activation. RBFNs are embedded in a two-layer neural network, where each hidden unit implements a radial activated function. The output

Table 2 Comparing the number of *incorrectly classified instances* on the test sets by each machine learning method

	GP	MP	RBFN	SVM	VP
Best	12	10	14	11	10
Average (SEM)	16.56 (0.37)	15.70 (0.49)	17.82 (0.28)	14.72 (0.35)	14.74 (0.31)

For each of the 50 runs performed with each method a different training/test partition of the dataset was used (see text for details). The first line indicates the method: Genetic Programming (GP), Multilayer Perceptron (MP), Radial Basis Function Network (RBFN), Support Vector Machines (SVM), and Voted Perceptron (VP). The second line shows the best value of the incorrectly classified instances obtained on the test set over the 50 runs, and the third line reports the mean (SEM) performance of each group of runs on their test sets

units implement a weighted sum of hidden unit outputs. The input into an RBFN is nonlinear while the output is linear. Their excellent approximation capabilities have been studied in [32, 33]. Due to their nonlinear approximation properties, RBFNs are able to model complex mappings, which generally Perceptron neural networks can only model by means of multiple intermediary layers [34]. In order to use an RBFN one needs to specify the hidden-unit activation function, the number of processing units, a criterion for modeling a given task, and a training algorithm for finding the parameters of the network. In this paper we have used the standard Weka implementation that consists in a normalized Gaussian radial function network. It uses the k-means clustering algorithm to provide the basis functions and learns a logistic regression (discrete class problems) on top of that. Symmetric multivariate Gaussians are fit to the data from each cluster. All the other parameters are specified in Table 1.

8 Experimental Results

The purpose of our experimental work focus was twofold. Firstly, we wanted to avoid the loss of information due to the forced binarization of the gene expression data performed in [17], we decided to use the original floating point valued expression data. The results of the comparison of machine learning techniques on this floating point dataset are discussed in Sect. 8.1. Secondly, we introduced a tunable fitness function in the GP runs: by assigning greater weight to false positives (negatives) in the definition of the fitness we aimed at tuning the algorithm towards better specificity (sensitivity): these results are presented in Sect. 8.2. In the final Sect. 8.3 we discuss the best solutions found by GP.

8.1 Comparing Different Algorithms

The results of the 50 runs performed for each method on the data described in Sect. 2 are summarized in Table 2. The first two lines indicate the different methods and the

Table 3 Comparing the number of *false negatives* on the test sets by each machine learning method

	GP	MP	RBFN	SVM	VP
Best	3	5	2	6	3
Average (SEM)	10.66 (0.48)	11.56 (0.49)	10.34 (0.43)	10.34 (0.36)	9.18 (0.36)

For each of the 50 runs performed with each method a different training/test partition of the dataset was used (see text for details). The first line indicates the method: Genetic Programming (GP), Multilayer Perceptron (MP), Radial Basis Function Network (RBFN), Support Vector Machines (SVM), and Voted Perceptron (VP). The second line shows the best value of the incorrectly classified instances obtained on the test set over the 50 runs, and the third line reports the mean (SEM) performance of each group of runs on their test sets

best (i.e., lowest) values of the incorrectly classified instances obtained on the test set over the 50 runs, respectively. In the third line we report the mean performances of each group of 50 runs on their test sets, along with the corresponding standard error of mean (SEM from now on).

To assess differences in performance we performed an ANOVA test that indicated that differences are significant (P-value 8.41×10^{-9}). As a post-test procedure we used Tukey's Honestly Significant Difference test to determine pairwise significant differences, which showed a difference between RBFN and all other methods, as well as showing that GP was outperformed by both VP and SVM. Finally, no statistically significant difference was detected between VP and SVM, and between MP and VP, SVM, and GP.

When implementing methods for risk stratification in cancer, it is especially imperative to minimize the number of false negative predictions (i.e., patients wrongly classified as low-risk). Table 3 summarizes the false negative predictions returned by each machine learning method on the 50 runs. As above, the first two lines indicate the different methods and the best (i.e., lowest) values of the false negatives obtained on the test set over the 50 runs while the third line reports the mean performances (together with the corresponding SEM) of each group of 50 runs on their test sets.

When considering only false negatives, the only difference evidenced by ANOVA (P-value of 4.16×10^{-4}) together with Tukey's Honestly Significant Difference test was the better performance of VP with respect to all the other methods.

8.2 Towards GP with Greater Sensitivity or Specificity

We thus concluded that, contrary to what observed with the binarized expression dataset in [17], when fed floating point-valued expression data, GP did not outperform the other machine learning techniques. On the other hand, it is well known that the fitness function driving the evolutionary dynamics in a GP framework can be modified in order to drive the emergence of solutions with desirable features. The

Table 4 Comparing the number of *incorrectly classified instances* on the test sets by each GP variant

	$GP_{1,9}$	$GP_{3,7}$	$GP_{5,5}$	$GP_{7,3}$	$GP_{9,1}$
Best	17	14	12	11	11
Average (SEM)	24.28 (0.45)	20.58 (0.48)	16.56 (0.37)	15.20 (0.26)	16.02 (0.39)

For each of the 50 runs performed with each variant a different training/test partition of the dataset was used (see text for details). The first line indicates the GP variant. The second line shows the best value of the incorrectly classified instances obtained on the test set over the 50 runs, and the third line reports the mean (SEM) performance of each group of runs on their test sets

fitness function used in the previous section gives the same weight to all incorrectly classified instances. However, as discussed above, minimizing the number of false negative predictions is recognized as one of the most important goals for a classifier intended for clinical use. In fact, the original scoring method originally introduced in [18, 20] was designed with the goal of minimizing the number of false negatives. It is therefore not surprising that in this respect the scoring method is far superior to all machine learning methods.

Therefore, we modified the GP fitness function in such a way that false negatives (positives) are penalized more than errors of the other type, thus allowing the algorithm to be tuned towards better sensitivity (specificity). In particular, solutions with greater sensitivity can emerge if larger weights are assigned to false negatives compared to false positives. The new fitness function can be written as a weighted average of the form:

$$Fitness = \alpha \times FalseNegative + \beta \times FalsePositive$$

where α and β are two floating point values with $\alpha, \beta \in [0, 1]$ and $\alpha + \beta = 1$. The choice $\alpha = \beta = 0.5$ corresponds to the GP fitness discussed in the previous section (referred to as $GP_{5,5}$ in the following). We then evolved four more GP classes, namely

- $GP_{1,9}$ where $\alpha = 0.1$ and $\beta = 0.9$;
- $GP_{3,7}$ where $\alpha = 0.3$ and $\beta = 0.7$;
- $GP_{7,3}$ where $\alpha = 0.7$ and $\beta = 0.3$;
- $GP_{9,1}$ where $\alpha = 0.9$ and $\beta = 0.1$.

In Table 4 we report the results of the 50 runs returned by each GP variant. As in the tables above, the first two lines indicate the different methods and the best (i.e., lowest) values of the incorrectly classified instances obtained on the test set over the 50 runs, respectively, while the third line reports the mean performances (together with the corresponding SEM) of each group of 50 runs on their test sets.

We performed a statistical analysis of these results among them and with the machine learning method that showed the best performance on minimizing the total number of incorrectly classified instances, namely the Support Vector Machines, as discussed in Sect. 8.1. ANOVA test showed a significant (P-value of 8.74×2.26^{-58})

Table 5 Comparing the number of *false negatives* on the test sets by each GP variant

	$GP_{1,9}$	$GP_{3,7}$	$GP_{5,5}$	$GP_{7,3}$	$GP_{9,1}$
Best	12	5	3	2	0
Average (SEM)	21.72 (0.55)	16.60 (0.73)	10.66 (0.48)	6.64 (0.37)	5.0 (0.37)

For each of the 50 runs performed with each variant a different training/test partition of the dataset was used (see text for details). The first line indicates the GP variant. The second line shows the best value of the incorrectly classified instances obtained on the test set over the 50 runs, and the third line reports the mean (SEM) performance of each group of runs on their test sets

overall difference. Tukey's Honestly Significant Difference test highlighted the outperformance of SVM, $GP_{9,1}$, $GP_{7,3}$, and $GP_{5,5}$ (that do not show any statistical difference among them) with respect to $GP_{3,7}$, and of $GP_{3,7}$ with respect to $GP_{1,9}$.

While the GP variants did not result in better performances on all incorrectly classified instances, this is not true when considering specifically false negatives. This issue is discussed below, while for reasons of space we do not discuss the symmetrical problem of false-positive minimization. Table 5 summarizes the results of the 50 runs returned by each GP variant. As above, the first two lines indicate the different methods and the best (i.e., lowest) values of the false negatives obtained on the test set over the 50 runs, respectively; the third line reports the mean performances (together with the corresponding SEM) of each group of 50 runs on their test sets.

A statistical analysis of the results shown in Table 5 results in the following ranking in the performance among the five GP variants and the VP, the best performing among all other machine learning techniques in minimizing false negatives (as discussed in Sect. 8.1): $GP_{9,1}$, $GP_{7,3}$, $GP_{5,5}$, $GP_{3,7}$, $GP_{1,9}$, with VP statistically indistinguishable from $GP_{5,5}$, but $GP_{9,1}$ outperforming $GP_{7,3}$ in a statistically significant way and $GP_{7,3}$ outperforming VP in a statistically significant way. Therefore we can conclude that the tunable fitness function is indeed a valuable tool when minimizing the false negative predictions.

The seventy-gene signature was used by authors of [18, 20] by assigning a coefficient to each of the genes (features) and computing a score for each patient as the scalar product of these coefficients and the patient gene expression. To compare the performance of the various machine learning algorithms, we proceeded as described in [17]. The mean number of false predictions was 16.7, while the mean number of false negative predictions was 1.7.

In [17], we compared this method's performances to those of the other machine learning techniques on the binarized dataset. Concerning the total number of false predictions, the scoring method appeared to be superior to all machine learning algorithm other than GP, and slightly inferior to GP. The difference between the performances of GP and the scoring method was not statistically significant. On the other hand, not surprisingly, with respect to false negative predictions, the scoring method is far superior to all machine learning methods, including GP.

It is intersecting to compare the predictive power of the best-performing GP classifier presented in this paper to the scoring method: while the scoring method

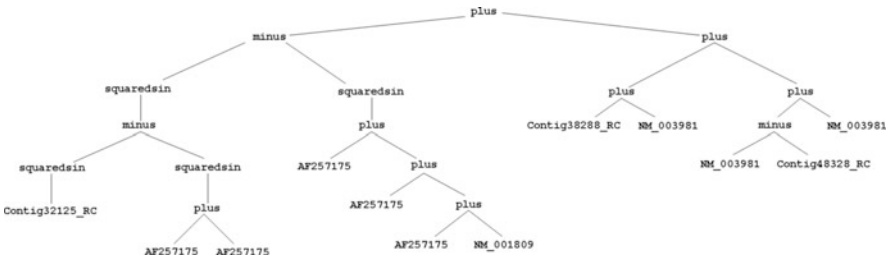

Fig. 1 Tree-based representation of the individual found by $GP_{1,9}$ generating the best value of incorrectly classified instances

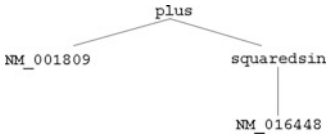

Fig. 2 Tree-based representation of the individual found by $GP_{1,9}$ with the best value of the false negatives

still outperforms GP in minimizing the number of false negative predictions, the opposite is true when considering the total number of incorrectly classified instances.

8.3 Analysis of the Best Solutions Found by GP

In Fig. 1, we report the individual who generated the minimum number of incorrectly classified instances found by $GP_{1,9}$, represented as a tree. This individual uses only 6 over the 70 totally available features, and this confirms the ability of GP to perform an automatic feature selection at the same time as the learning phase. Furthermore, some of the features, for instance `AF257175`, recur frequently in this individual's expression. We hypothesize that this indicates an interesting correlation between these features and the target, or at least the fact that these features are important in determining the target value itself. In the same way, Fig. 2 shows the individual with the best value of the false negatives found by $GP_{1,9}$. In this case, the automatic feature selection performed by GP is even stronger, given that only two features over the 70 globally available ones are used by this solution. Interestingly, one of those two features (`NM001809`) was used also by the solution shown in Fig. 1. This is a clear indication of the impact of this particular feature in determining the target. Another interesting characteristic shared by the individuals in Fig. 1 and in Fig. 2 is the fact that, in both cases, some operators (like `squaredsin`, `plus`, and `minus`) are used more frequently than the other ones (indeed the multiplication and division operators do not appear at all in these solutions).

Table 6 Features used in the solutions found by GP reported in Figs. 1 and 2. The two columns show accession ID and gene description

Accession ID	Gene description
Contig32125RC	–
AF257175	Homo sapiens hepatocellular carcinoma-associated antigen 64 (HCA64) mRNA, complete cds
NM001809	Homo sapiens centromere protein A (CENPA), transcript variant 1, mRNA
Contig38288RC	ESTs, weakly similar to quiescin
NM003961	Homo sapiens rhomboid, veinlet-like 1 (Drosophila) (RHBDL1), mRNA
Contig48328RC	–
NM016448	Homo sapiens denticleless homolog (Drosophila) (DTL), mRNA

The experiments presented so far, as well as a wide set of further experiments which returned similar results (not presented here to save space), lead us to hypothesize a strong impact of the features contained in the two solutions shown in Figs. 1 and 2 in determining the target value of the used datasets. For this reason, in Table 6 we list these features, showing the gene accession IDs and an informal description of the genes.

9 Conclusions and Future Work

Predicting the breast cancer risk is an ambitious and important task, which possibly constitutes one of the most relevant challenges in the attempt of developing personalized therapies nowadays. The goal is still far from being accomplished, and this paper represents only a partial and initial contribution. As a starting point, we considered the well-known "70-genes signature" and we compared the results obtained with several machine learning methods. This choice is motivated by the fact that the "70-genes signature" has been widely studied so far and the genes contained in it are widely accepted to have an important impact on the target. One of the first objectives of our study is to further refine this information, trying to understand, with the help of machine learning schemes, which of these genes are the most important in target determination and why. For this reason, we believe that Genetic Programming (GP) can be a very promising technique, given its well-known ability of performing an automatic feature selection at learning time.

In the first phase of our investigation, we showed that all the machine learning algorithms we used do have predictive power in classifying breast cancer patients into risk classes and GP is outperformed by Support Vector Machines and Voted Perceptron in the minimization of the number of incorrectly classified instances and by Voted Perceptron in the minimization of false negatives. To minimize the possible bias, in this first phase we tried to use default implementation and parameter settings for all methods.

In the second phase, given the different impact of false negatives and false positives in therapeutic aspects, we tried to enrich GP by changing its fitness

function into a weighted average between these two error measures. Changing the fitness function is a possibility that GP offers in a quite natural and straightforward manner, and that has no equivalent in many other machine learning methods. In this way, we are able to adapt the GP learning phase to our requirements, biasing the search towards particular regions of the solutions space. In particular, we performed some experiments with different values of the weights and we showed that, when larger weight is given to false negatives, GP obtains results that are comparable to all the other machine learning methods in minimizing the number of incorrectly classified instances and it is able to outperform the other machine learning methods in a statistically significant way in minimizing false negatives.

Another interesting study that we have performed consisted in the analysis of the data contained in the "70-genes signature" both with the original floating point valued expressions and with a preliminary binarization. Our results indicated that GP used on the original floating point valued expression data outperforms GP used on binarized data (the results obtained on binarized data are presented in [17]). Interestingly, this is true both in the minimization of the incorrectly classified instances and in the minimization of the false negatives and these results are always statistically significant.

Also, we point out that the improvement in performance shown by GP in minimizing false negatives compared to the original scoring method presented in [18, 20] was rather small and not statistically significant.

The results presented in this paper pave the way for further investigation on the use of GP for risk prediction in breast cancer. In particular, we think that GP is a promising approach and that the presented results can be further improved in the future because:

- The parameter setting used in this paper was purposely not optimized, and we can expect substantial improvements in performance from a fine-tuning of the various GP parameters.
- GP can potentially offer biological insight and generate hypotheses for experimental work (see also [14]). Indeed, an important result of our analysis is that the trees produced by GP tend to contain a limited number of features, often repeated several times, and therefore are easily interpretable in biological terms.
- We can change the behavior of GP by modifying the fitness function. In particular, the algorithm can be tuned towards better sensitivity (specificity), simply by defining a fitness function in which false negatives (positives) are penalized more than errors of the other type.

With these premises, we will focus our future work on both improving the performance of GP and interpreting the results from the biological viewpoint. Furthermore, we are currently investigating the impact of the employed $t_{\text{end}} = 10.3$ years endpoint, in particular on the sensitivity/specificity experiments. Using different thresholds, in fact, widely changes the dataset and the relationships between features and target, possibly offering new insights. Last but not least, we plan to extend our investigations to other datasets on breast cancer as well as to other kinds of diseases.

Acknowledgments This work was partially supported by Neuroscience Program of the Compagnia di San Paolo in Torino.

References

1. Nevins, J.R., Potti, A.: Mining gene expression profiles: expression signatures as cancer phenotypes. Natl. Rev. Genet. **8**(8), 601–609 (2007)
2. Lu, Y., Han, J.: Cancer classification using gene expression data. Inf. Syst. **28**(4), 243–268 (2003)
3. Michie, D., Spiegelhalter, D., Taylor, C.: Machine learning, neural and statistical classification. Prentice-Hall, Englewood Cliffs, NJ (1994)
4. Alon, U., Barkai, N., Notterman, D., Gish, K., Ybarra, S., Mack, D., Levine, A.J.: Broad patterns of gene expression revealed by clustering analysis of tumour and normal colon tissues probed by oligonucleotide arrays. Proc. Natl. Acad. Sci. **96**, 6745–6750 (1999)
5. Hsu, A., Tang, S., Halgamuge, S.: An unsupervised hierarchical dynamic self-organizing approach to cancer class discovery and marker gene identification in microarray data. Bioinformatics **19**(16), 2131–2140 (2003)
6. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Mach. Learn. **46**, 389–422 (2002)
7. Hernandez, J.C.H., Duval, B., Hao, J.: A genetic embedded approach for gene selection and classification of microarray data. Lect. Notes Comput. Sci. **4447**, 90–101 (2007)
8. Friedman, N., Linial, M., Nachmann, I., Peer, D.: Using bayesian networks to analyze expression data. J. Comput. Biol. **7**, 601–620 (2000)
9. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
10. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA (1989)
11. Liu, J., Cutler, G., Li, W., Pan, Z., Peng, S., Hoey, T., Chen, L., Ling, X.-B.: Multiclass cancer classification and biomarker discovery using ga-based algorithms. Bioinformatics **21**, 2691–2697 (2005)
12. Moore, J., Parker, J., Hahn, L.: Symbolic discriminant analysis for mining gene expression patterns. Lect. Notes Artif. Int. **2167**, 372–381 (2001)
13. Rosskopf, M., Schmidt, H., Feldkamp, U., Banzhaf, W.: Genetic programming based dna microarray analysis for classification of tumour tissues. Technical Report 2007-2003, Memorial University of Newfoundland (2007)
14. Yu, J., Yu, J., Almal, A.A., Dhanasekaran, S.M., Ghosh, D., Worzel, W.P., Chinnaiyan, A.M.: Feature selection and molecular classification of cancer using genetic programming. Neoplasia **9**(4), 292–303 (2007)
15. Bojarczuk, C., Lopesb, H., Freitasc, A.: Data mining with constrained-syntax genetic programming: applications to medical data sets. Proc. Intell. Data Anal. Med. Pharmacol. (2001)
16. Hong, J., Cho, S.: The classification of cancer based on dna microarray data that uses diverse ensemble genetic programming. Artif. Intell. Med. **36**, 43–58 (2006)
17. Vanneschi, L., Farinaccio, A., Giacobini, M., Antoniotti, M., Mauri, G., Provero, P.: Identification of individualized feature combinations for survival prediction in breast cancer: a comparison of machine learning techniques. In: Giacobini, M., et al. (eds.) Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics. Proceedings of the Nineth European Conference, EvoBIO 2010. Lecture Notes in Computer Science, LNCS 6023, pp. 110–121. Springer, Berlin (2010)
18. van 't Veer, L.J., Dai, H., van de Vijver, M.J., He, Y.D., Hart, A.A.M., Mao, M., Peterse, H.L., van der Kooy, K., Marton, M.J., Witteveen, A.T., Schreiber, G.J., Kerkhoven, R.M., Roberts, C., Linsley, P.S., Bernards, R., Friend, S.H.: Gene expression profiling predicts clinical outcome of breast cancer. Nature **415**(6871), 530–536 (2002)

19. Koza, J.R.: Genetic Programming. MIT, Cambridge, MA (1992)
20. van de Vijver, M.J., He, Y.D., van't Veer, L.J., Dai, H., Hart, A.A.M., Voskuil, D.W., Schreiber, G.J., Peterse, J.L., Roberts, C., Marton, M.J., Parrish, M., Atsma, D., Witteveen, A., Glas, A., Delahaye, L., van der Velde, T., Bartelink, H., Rodenhuis, S., Rutgers, E.T., Friend, S.H., Bernards, R.: A gene-expression signature as a predictor of survival in breast cancer. N. Engl. J. Med. **347**(25), 1999–2009 (2002)
21. Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk (2008) (With contributions by J.R. Koza)
22. Archetti, F., Lanzeni, S., Messina, E., Vanneschi, L.: Genetic programming for human oral bioavailability of drugs. In: Cattolico, M., et al. (eds.) Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 255–262. Seattle, Washington, DC (2006)
23. Archetti, F., Messina, E., Lanzeni, S., Vanneschi, L.: Genetic programming and other machine learning approaches to predict median oral lethal dose (LD50) and plasma protein binding levels (%PPB) of drugs. In: Marchiori, E., et al. (eds.) Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics. Proceedings of the Fifth European Conference, EvoBIO 2007. Lecture Notes in Computer Science, LNCS 4447, pp. 11–23. Springer, Berlin (2007)
24. Archetti, F., Messina, E., Lanzeni, S., Vanneschi, L.: Genetic programming for computational pharmacokinetics in drug discovery and development. Genet. Program. Evol. M. **8**(4), 17–26 (2007)
25. Silva, S.: GPLAB: a genetic programming toolbox for MATLAB, version 3.0. http://gplab.sourceforge.net (2007)
26. Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998)
27. Platt, J.: Fast training of support vector machines using sequential minimal optimization. Advances in Kernel Methods – Support Vector Learning, pp. 185–208. MIT Press, Cambridge (1998)
28. Weka: A multi-task machine learning software developed by Waikato University. www.cs.waikato.ac.nz/ml/weka (2006)
29. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice-Hall, London (1999)
30. Freund, Y., Schapire, R.E.: Large margin classification using the perceptron algorithm. In: The Eleventh Annual Conference on Computational Learning Theory, Machine Learning, **37**(3), 277–296 (1999)
31. Helmbold, D.P., Warmuth, M.K.: On weak learning. J. Comput. Syst. Sci. **50**(3), 551–573 (1995)
32. Park, J., Sandberg, J.W.: Universal approximation using radial basis functions network. Neural Comput. **3**, 246–257 (1991)
33. Poggio, T., Girosi, F.: Networks for approximation and learning. P. IEEE **78**(9), 1481–1497 (1990)
34. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice-Hall, London (1999)

A Neuro-Evolutionary Approach to Electrocardiographic Signal Classification

Antonia Azzini, Mauro Dragoni, and Andrea G.B. Tettamanzi

Abstract This chapter presents an evolutionary Artificial Neural Networks (ANN) classifier system as a heartbeat classification algorithm designed according to the rules of the PhysioNet/Computing in Cardiology Challenge 2011 (Moody, Comput Cardiol Challenge 38:273–276, 2011), whose aim is to develop an efficient algorithm able to run within a mobile phone that can provide useful feedback when acquiring a diagnostically useful 12-lead Electrocardiography (ECG) recording. The method used to solve this problem is a very powerful natural computing analysis tool, namely evolutionary neural networks, based on the joint evolution of the topology and the connection weights relying on a novel similarity-based crossover. The chapter focuses on discerning between usable and unusable electrocardiograms tele-medically acquired from mobile embedded devices. A preprocessing algorithm based on the Discrete Fourier Transform has been applied before the evolutionary approach in order to extract an ECG feature dataset in the frequency domain. Finally, a series of tests has been carried out in order to evaluate the performance and the accuracy of the classifier system for such a challenge.

A. Azzini
Università degli Studi di Milano, Dipartimento di Tecnologie dell'Informazione, via Bramante, 65 - 26013 Crema (CR) Italy
e-mail: antonia.azzini@unimi.it

M. Dragoni (✉)
Fondazione Bruno Kessler (FBK-IRST), Via Sommarive 18, Povo, Trento, Italy
e-mail: dragoni@fbk.eu

A.G.B. Tettamanzi
I3S Laboratory UMR 7271 - UNS/CNRS/INRIA, Ple GLC, WIMMICS Research Team, 930 route des Colles - Bt. Les Templiers, 06903 Sophia Antipolis CEDEX, Nice, France
e-mail: andrea.tettamanzi@unice.fr

S. Cagnoni et al. (eds.), *Evolution, Complexity and Artificial Life*,
DOI 10.1007/978-3-642-37577-4_13,

1 Introduction

In the last decades, cardiovascular diseases (CVD) have represented one of the most important causes of death in the world [1] and the necessity of a trustworthy heart state evaluation is increasing. Electrocardiography (ECG) is one of the most useful and well-known methods for heart state evaluation. Indeed, ECG analysis is still one of the most common and robust solutions for diagnosing heart diseases, also because it is one of the simplest noninvasive diagnostic methods for various heart diseases [2].

In such a research field, one of the most important critical aspects regards the quality of heart state evaluations since, often, the lack of medically trained experts, working from the acquisition process to the discernment between usable and unusable medical information, increases the need of easy and efficient measuring devices, which can send measured data to a specialist. Furthermore, the volume of the data that have to be recorded is huge and, very often, the ECG records are non-stationary signals, and critical information may occur at random in the time scale. In this situation, the disease symptoms may not be always appreciatable, but would show up at irregular intervals during the day.

In this sense, the Physionet Challenge [3], on which this chapter focuses, aims at reducing, if not eliminating, all the fallacies that currently plague usable medical information provided tele-medically, by obtaining efficient measuring systems through smart phones.

In this challenge, whose aim is ECG classification, several approaches were explored; in particular, in order to inform inexperienced user about the quality of measured ECGs, artificial-intelligence-based (AI-based) systems have been considered, to reduce the percentage of bad-quality ECGs sent to the specialist and, thus, contribute to a more effective use of her time.

Moody and colleagues [4] reported that some of the top competitors in this challenge employed a variety of techniques, using a wide range of features including entropy, higher order moments, intra-lead information, etc., while the classification methods also included Decision Trees, Support Vector Machines (SVMs), Fuzzy Logic, and heuristic rules.

An example of SVM-based approach is reported in [1], where the authors developed a decision support system based on an algorithm that combines simple rules in order to discard recordings of obvious low quality and a more sophisticated classification technique for improving the quality of AI-based systems for mobile phones, including the fine-tuning of detection sensitivity and specificity. Another example has also been given in [5], where a rule-based classification method that mimics the SVM has been implemented, by using a modified version of a real-time QRS-Complex detection algorithm and a T-Wave detection approach.

According to [4], Artificial Neural Networks (ANNs) have been extensively employed in computer-aided diagnosis because of their remarkable qualities: capacity of adapting to various problems, training from examples, and generalization capabilities with reduced noise effects. Also Jiang and colleague confirmed the

usefulness of ANNs as heartbeat classifiers, emphasizing in particular evolvable ANNs, due to their ability to change the network structure and internal configurations, as well as the parameters, to cope with dynamic operating environments. In particular, the authors developed an evolutionary approach for structure and weight optimization of block-based neural network (BbNN) models [6] for a personalized ECG heartbeat pattern classification.

We approach the heartbeat classification problem by another evolutionary algorithm for joint structure and weights optimization of ANNs [7], which exploits an improved version of a novel similarity-based crossover operator [8], based on the conjunction of topology and connection weight optimization.

This chapter is organized as follows: Sect. 2 briefly presents the problem, while a description of the evolutionary approach considered in this work is reported in Sect. 4. The results obtained in the experiments carried out are presented in Sect. 5, along with a discussion of the performances obtained. Finally, Sect. 6 provides some concluding remarks.

2 Problem Description

As previously reported, the ECG is a bio-electric signal that records the electrical activity of the heart. It provides helpful information about the functional aspects of the heart and of the cardiovascular system, and the state of cardiac health is generally reflected in the shape of ECG waveform, that is a critical information. For this reason, computer-based analysis and classification and automatic interpretation of the ECG signals can be very helpful to assure a continuous surveillance of the patients and to prepare the work of the cardiologist in the analysis of long recordings.

Moreover, as indicated by the main documentation of Physionet, according to the World Health Organization, CVD are the number one cause of death worldwide. Of these deaths, 82 % take place in low- and middle-income countries. Given their computing power and pervasiveness, the most important question is to verify whether mobile phones can aid in delivering quality health care, particularly to rural populations distant from physicians having the expertise needed to diagnose CVD.

Advances in mobile phone technology have resulted in global availability of portable computing devices capable of performing many of the functions traditionally requiring desktop or larger computers. In addition to their technological features, mobile phones have a large cultural impact. They are user-friendly and are among the most efficient and most widely used means of communication. With the recent progress of mobile-platforms, and the increasing number of mobile phones, a solution to the problem under consideration can be the recording of ECGs by untrained professionals, and the subsequent transmission to a human specialist.

The aim of the PhysioNet/Computing in Cardiology Challenge 2011 [9] is to develop an efficient algorithm, able to run in near real-time within a mobile phone that can provide useful feedback to a layperson in the process of acquiring a

diagnostically useful ECG recording. In addition to the approaches already cited in Sect. 1, referring to such a challenge, Table 3 reports other solutions already presented in the literature, capable of quantifying the quality of the ECG looking at individual or combined leads, which can be implemented on a mobile platform. As reported later, all such approaches are used to compare their results with those obtained in this work.

3 Neuro-Evolutionary Classifiers

Generally speaking, a supervised ANN is composed of simple computing units (the *neurons*) which are connected to form a network [10–12]. Whether a neuron a influences another neuron b or not depends on the ANN structure. The extent of such influence, when there is one, depends on the weight assigned to each connection among the neurons. It is very difficult to find an optimal network (structure and weights) for a given problem.

Even though some authors do not consider supervised classification as a good domain for neuroevolution, preferring alternatives as (SVMs), Bayesian methods, or analytic optimization methods, neural networks are nevertheless one of the most popular tools for classification. The recent vast research activities in neural classification establish that neural networks are an effective alternative to various conventional classification methods and a large number of successful applications presented in the recent literature demonstrate that ANN design can be further improved by synergetically combining it with evolutionary algorithms, being able to take into account all aspects of ANN design at one time [13].

The review by Zhang [14], which provides a summary of the most important advances in classification with ANNs, shows clearly that the advantages of neural networks are manyfold: they are capable of adapting themselves to the data without any explicit specification of functional or distributional form for the underlying model; they are universal functional approximators; they represent nonlinear and flexible solutions for modeling real-world complex relationships; and, finally, they are able to provide a basis for establishing classification rules and performing statistical analysis. On the other hand, different neuro-evolutionary approaches have been successfully applied to a variety of benchmark problems and real-world classification tasks [15–18]. Our neuro-evolutionary algorithm, too, has already been tested and applied with success to several real-world problems, showing how such an approach can be useful in different classification problems, like automated trading strategy optimization [19, 20], incipient fault diagnosis in electrical drives [21], automated diagnosis of skin diseases [22], etc. Further insights on the evolutionary optimization of ANNs can be found in some broad surveys on the topic [23–25].

4 The Neuro-Evolutionary Algorithm

The overall algorithm is based on the evolution of a population of individuals, represented by Multilayer Perceptron neural networks (MLPs), through a joint optimization of their structures and weights, briefly summarized here; a more complete and detailed description can be found in [7]. In this work the algorithm uses the Scaled Conjugate Gradient method (SCG) [26] instead of the more traditional error back-propagation (BP) algorithm, in order to speed up the convergence of such a conventional training algorithm. Accordingly, it is the genotype which undergoes the genetic operators and reproduces itself, whereas the phenotype is used *only* for calculating the genotype's fitness. The rationale for this choice is that the alternative of applying SCG to the genotype as a kind of "intelligent" mutation operator would boost exploitation while impairing exploration, thus making the algorithm too prone to being trapped in local optima.

The population is initialized with different number of layers and neurons for each individual according to two exponential distributions, in order to maintain diversity among all of them in the new population. Such dimensions are not bounded in advance, even though the fitness function may penalize large networks. The number of neurons in each hidden layer is constrained to be greater than or equal to the number of network outputs, in order to avoid hourglass structures, whose performance tends to be poor. Indeed, a layer with fewer neurons than the outputs destroys information which later cannot be recovered.

Thanks to this encoding, individual ANNs are not constrained to a preestablished topology. Unlike NeuroEvolution of Augmenting Topologies (NEAT) [27], which starts with minimal network topologies and then applies evolutionary mechanisms to augment them, our approach randomly initializes the network's population with different hidden layer sizes and numbers of neurons for each individual according to two exponential distributions, in order not to constrain search and to provide a balanced mix of topologies.

4.1 Evolutionary Process

In the evolutionary process, the genetic operators are applied to each network until the termination condition is satisfied, i.e., until the maximum number of generations is reached or no further improvement of the fitness function can be obtained. In each new generation, a new population of size n has to be created, and the first half of such a new population corresponds to the best parents that have been selected by the truncation operator, while the second part of the new population is filled with offspring of the previously selected parents. A child individual is generated by applying the crossover operator to two individuals, selected from the best half of the population (parents), if their local similarity condition is satisfied. Otherwise, the child corresponds to a randomly chosen copy of either of the two selected parents.

Elitism allows the best individual to survive unchanged into the next generation. Then, the algorithm mutates the weights and the topology of the offspring, trains the resulting networks, calculates the fitness on the test set, and finally saves the best individual and statistics about the entire evolutionary process. Although the joint application of truncation, selection, and elitism could seem to exert a strong selection pressure, which could produce too fast a convergence of the solutions, all the experiments carried out with the `SimBa` [8] crossover outperform the other approaches without showing any premature convergence of the results.

The general framework of the evolutionary process can be described by the following pseudo-code. Individuals in a population compete and communicate with each other through genetic operators applied with independent probabilities, until the termination condition is met.

1. Initialize the population by generating new random individuals.
2. For each genotype, create the corresponding MLP, and calculate its cost and its fitness values.
3. Save the best individual as the best-so-far individual.
4. While not termination condition do:

   (a) Apply the genetic operators to each network.
   (b) Decode each new genotype into the corresponding network.
   (c) Compute the fitness value for each network.
   (d) Save statistics.

The application of the genetic operators to each network is described by the following pseudo-code:

1. Select $\lfloor n/2 \rfloor$ individuals from the population (of size n) by truncation and create a new population of size n with copies of the selected individuals.
2. For all individuals in the population:

   (a) Randomly choose two individuals as possible parents.
   (b) If their local similarity is satisfied

   then generate the offspring by applying crossover according to the crossover probability.
   else generate the offspring by randomly choosing either of the two parents.

   (c) Mutate the weights and the topology of the offspring according to the mutation probabilities.
   (d) Train the resulting network using the training set.
   (e) Calculate the fitness f on the test set.

3. Save the individual with lowest f as the best-so-far individual if the f of the previously saved best-so-far individual is higher (worse).
4. Save statistics.

For each generation of the population, all the information about the best individual is saved.

Table 1 Parameters of the algorithm

Symbol	Meaning	Default value
n	Population size	60
p^{+}_{layer}	Probability of inserting a hidden layer	[0.05,0.15,0.30,0.45]
p^{-}_{layer}	Probability of deleting a hidden layer	[0.05,0.15,0.30,0.45]
p^{+}_{neuron}	Probability of inserting a neuron in a hidden layer	[0.05,0.15,0.30,0.45]
p_{cross}	"Desired" probability to apply crossover	[0.2,0.4,0.6,0.8,1.0]
δ	Crossover similarity cutoff value	0.1
N_{in}	Number of network inputs	[a]
N_{out}	Number of network outputs	[a]
α	Cost of a neuron	2
β	Cost of a synapse	4
λ	Desired trade-off between network cost and accuracy	0.2
k	Constant for scaling cost and MSE in the same range	10^{-6}

[a] Benchmark dataset dependent

Table 1 lists all the parameters of the algorithm; their values, reported in the third column, have been experimentally found as those that provide the most satisfactory results.

4.1.1 Selection

Truncation selection, the selection method implemented in this work, is taken from the breeder genetic algorithm [28], and differs from natural probabilistic selection in that evolution only considers the individuals that best adapt to the environment. Truncation selection is not a novel solution and previous work considered such a selection in order to prevent the population from remaining too static and perhaps not evolving at all [29]. It is a very simple technique which produces satisfactory solutions in conjunction with other strategies, like elitism, which allows the best individual to survive unchanged into the next generation and solutions to monotonically get better over time.

4.1.2 Mutation

The main function of this operator is to introduce new genetic material and to maintain diversity in the population. Generally, the purpose of mutation is to simulate the effects of transcription errors that can occur with a very low probability, the mutation rate, when a chromosome is replicated. The evolutionary process applies two kinds of neural network perturbations: weights mutation and topology mutation.

Weights mutation perturbs the weights of the neurons before performing any structural mutation and applying SCG. This kind of mutation uses a Gaussian

distribution with zero mean and variance given by matrix $\mathbf{Var}^{(i)}$ for each network. as illustrated in Table 2. This solution is similar to the approach implemented by *evolution strategies* [30], algorithms in which the strategy parameters are proposed for self-adapting the mutation concurrently with the evolutionary search. The main idea behind these strategies is to allow a control parameter, like mutation variance, to self-adapt rather than changing its value according to some deterministic algorithm. Evolution strategies perform very well in numerical domains and are well suited to (real) function optimization. This kind of mutation offers a simplified method for self-adapting each single value of the Variance matrix $\mathbf{Var}_j^{(i)}$, whose values are defined as log-normal perturbations of their parent parameter values.

Topology mutation can apply four types of mutation by considering neuron and layer addition and elimination. The addition and the elimination of a layer and the insertion of a neuron are applied with independent probabilities, corresponding, respectively, to the three algorithm parameters p^{+}_{layer}, p^{-}_{layer}, and p^{+}_{neuron}, while a neuron is eliminated only when its contribution becomes negligible (less than 5 %) with respect to the overall behavior of the network [19]. The parameters used in such a kind of mutation are set at the beginning and maintained unchanged during the entire evolutionary process.

All the topology mutation operators are aimed at minimizing their impact on the behavior of the network; in other words, they are designed to be as little disruptive, and as much neutral, as possible, preserving the parent's effectiveness in its offspring better than by adding random nodes or layers.

4.1.3 Fitness Function

Although it is customary in EAs to assume that better individuals have higher fitness, as previously considered [19, 31], the convention that a lower fitness means a better ANN is adopted in this work. This maps directly to the objective function of an error- and cost-minimization problem, which is the natural formulation of most problems ANNs can solve.

The fitness function is calculated, after the training and the evaluation processes, by Eq. (1) and is defined as a function of the confusion matrix M obtained by that individual:

$$f_{\text{multiclass}}(M) = N_{\text{outputs}} - \text{Trace}(M), \tag{1}$$

where N_{outputs} is the number of output neurons and $\text{Trace}(M)$ is the sum of the diagonal elements of the row-wise normalized confusion matrix, which represent the conditional probabilities of the predicted outputs given the actual ones.

4.1.4 Crossover

In general, recombination in EAs is most useful before convergence, during the exploration phase, when it may help locating promising areas of the search space. This is exactly why the local-similarity-based crossover presented in this work turns out to be beneficial. Indeed, this operator exploits the concept of similarity among individuals, which is one of the first ideas developed in the literature for solving such a problem.

We have given particular attention to two empirical studies [32, 33], which focus on the evolution of the single network unit involved in the crossover operator, the hidden node. Their aim was to emphasize the equivalence between hidden nodes of ANNs, in order to identify similarly performing units prior to crossover, avoiding all the disruptive effects stated above. Following such an idea, we extend our neuro-genetic approach already presented in the literature [7, 19], which implements a joint optimization of weights and network structure, by defining a novel crossover operator. This operator allows recombination of individuals that have different topologies, but with hidden nodes that are similarly performing in the cutting point of the hidden layer randomly chosen (indicated in the approach as *local similarity*). The evolutionary process does not consider only a part, but complete MLPs, achieving satisfactory performances and generalization capabilities, as well as reduced computational costs and network sizes.

The crossover is applied with a probability parameter p_{cross}, defined by the user together with all the other genetic parameters, and maintained unchanged during the entire evolutionary process. A significant aspect related to the crossover probability considered in this work is that it refers to a “desired" probability, a genetic parameter set at the beginning of the evolutionary process that indicates the probability with which the crossover should be applied. However, the “actual" crossover probability during the execution of the algorithm is less than or equal to the desired one, because the application of the crossover operator is conditional on a sort of “compatibility" of the individuals involved. We report a summary of the crossover operator, already discussed in the literature [8, 31].

The `SimBa` crossover starts by looking for a “local similarity" between two individuals selected from the population. We refer to “local similarity" as a situation in which, in both individuals, there are two consecutive layers (i and $i + 1$) with the same number of neurons. This is a necessary condition for the application of our crossover operator because its aim is to overcome the problem related to structure incompatibility between individuals. The contribution of each neuron of the layers selected for the crossover is computed, and the neurons of each layer are reordered according to their contribution (i.e., the output obtained by evaluating the neural network, up to that neuron, over the training dataset), which strongly depends on its input connections. We also choose a cutoff threshold δ that will be used for swapping the neurons.

Then, each neuron of the layer selected in the first individual is associated with the most “similar" neuron (the neuron with the most similar output) in the

other individual's layer, and the neurons of the layer of the second individual are re-ranked by considering the associations with the neurons of the first one. All the neuron associations linked with an output difference higher than the cutoff value δ are discarded. Such a cutoff value δ can be seen as the "local-similarity" threshold. In this work it has been experimentally defined, in the beginning, equal to 0.1 and is maintained unchanged during the entire evolutionary process. Only the neurons above such a similarity threshold are eligible for being swapped, while the others will remain unchanged. Finally, a cut-point is randomly selected and the neurons above the cut-point are swapped by generating the offspring of the selected individuals.

5 Experiments and Results

The data used for the PhysioNet/CINC 2011 Challenge consist of 2,000 twelve-lead ECGs (I, II, III, aVR, aVF, aVL, V1, V2, V3, V4, V5, and V6), each 10 s long, with a standard diagnostic bandwidth defined in the range (0.05–100 Hz). The twelve leads are simultaneously recorded for a minimum of 10 s; each lead is sampled at 500 Hz with 16-bit resolution.

The proposed approach has been evaluated by using the dataset provided by the challenge organizers. This dataset, described above in Sect. 2, is public and has been distributed in two different parts:

- Set A: this dataset has to be used to train the approach. It is composed of 998 instances provided with reference quality assessments;
- Set B: this dataset has to be used for testing the approach. It is composed of 500 instances and the reference quality assessments are not distributed to the participants. The reports generated by the approach must be sent to the submission system in order to receive results valid for the challenge.

We split Set A into two parts: a training set consisting of 75 % of the instances contained in Set A, and a validation set, used to stop the training algorithm, consisting of the remaining 25 %, while Set B is used as test set for the final evaluation of the approach.

Each instance of the dataset represents an ECG signal composed of 12 series (one for each lead) of 5,000 values representing the number of recordings performed for each lead. These data have been preprocessed in order to extract the features that we used to create the datasets input to the algorithm. We have applied the FFT to each lead in order to transform each lead into the frequency domain. After the transformation, we summed the 5,000 values by groups of 500 in order to obtain ten features for each lead. Finally, the input attributes of all datasets have been rescaled, before being fed as inputs to the population of ANNs, according to a Gaussian distribution with zero mean and standard deviation equal to 1.

The experiments have been carried out by setting the parameters of the algorithm to the values obtained in a first round of experiments aimed at identifying the best

Table 2 Default parameters of the algorithm

Symbol	Meaning	Default value
n	Population size	60
p^{+}_{layer}	Probability of inserting a hidden layer	0.05
p^{-}_{layer}	Probability of deleting a hidden layer	0.05
p^{+}_{neuron}	Probability of inserting a neuron in a hidden layer	0.05
p_{cross}	"Desired" probability of applying crossover	0.7
δ	Crossover similarity cutoff value	0.9
N_{in}	Number of network inputs	120
N_{out}	Number of network outputs	1
α	Cost of a neuron	2
β	Cost of a synapsis	4
λ	Desired trade-off between network cost and accuracy	0.2
k	Constant for scaling cost and MSE in the same range	10^{-6}

parameter setting. These parameter values are reported in Table 2. We performed 40 runs, with 40 generations and 60 individuals for each run, while the number of epochs used to train the neural network implemented in each individual has been set to 250.

The challenge has been articulated in two different events: a closed event and an open one. While in the closed event it is possible to develop the classification algorithm in any language, in the open event it is mandatory to develop the algorithm in Java. For this reason, considering that the proposed approach has been developed in Java too, we compared the results we obtained to those obtained by the other systems that participated in the challenge in the open event. It is important to highlight that we did not aim at obtaining the best performance, but at showing that, even if our system is trained with a training set that embeds very little information, the performance obtained by our approach does not lag too much behind the results obtained by the best state-of-the-art systems.

Table 3 shows the results obtained by the other participants compared with the results obtained by the proposed approach. Besides comparing our approach with the other approaches presented at the challenge, we have also compared it with the other following neuro-genetic approaches:

- Simple ANN with Conjugated Gradient: the classifiers are encoded with a population of ANNs, trained with the Conjugated Gradient method over 1,000,000 epochs. Also in this case, the networks are then evaluated over the validation and the test sets, respectively, through the computation of the mean square error.
- NEAT approach [27]: an evolutionary approach applied to neural network design that: (1) uses a crossover on different topologies, (2) protects structural innovation by using speciation, and (3) applies an incremental growth from minimal network structures.
- Evolved ANN without crossover: the population of ANNs are evolved through the joint optimization of architecture and connection weights reported in this chapter, but in this case no crossover is implemented. The number of epochs is set to 250.

Table 3 Results of the open event challenge

Participant	Score
Xiaopeng Zhao [34]	0.914
Proposed Approach (Best)	**0.902**
Benjamin Moody [35]	0.896
Proposed Approach (Average)	**0.892**
Lars Johannesen [36]	0.880
Philip Langley [37]	0.868
NEAT (Average)	*0.856*
Evolved ANN without crossover (Average)	*0.845*
Dieter Hayn [38]	0.834
Vclav Chudcek [39]	0.833
Simple ANN with Conjugated Gradient (Average)	*0.818*

Table 4 Results of the tenfold cross validation

Training set	Validation set	Test set	Avg accuracy	Std deviation
F1...F7	F8, F9	F10	0.8984	0.0035
F2...F8	F9, F10	F1	0.8988	0.0067
F3...F9	F10, F1	F2	0.9002	0.0075
F4...F10	F1, F2	F3	0.9022	0.0107
F5...F10, F1	F2, F3	F4	0.9040	0.0071
F6...F10, F1, F2	F3, F4	F5	0.9002	0.0029
F7...F10, F1...F3	F4, F5	F6	0.9002	0.0018
F8...F10, F1...F4	F5, F6	F7	0.8976	0.0054
F9, F10, F1...F5	F6, F7	F8	0.9032	0.0090
F10, F1...F6	F7, F8	F9	0.8986	0.0047

We report both the best and the average performance obtained by the proposed approach. It is possible to observe that if we consider the best performance, we obtained the second best accuracy; while the average accuracy, computed over the 40 runs, obtained the fourth performance. The robustness of the approach is also proved by observing the low value of the standard deviation that, in the performed experiments, was 0.011. In italics, we show the performance obtained by the other approaches that we have used for classifying the data in order to compare them with the approach proposed in this paper. The results demonstrated that the proposed approach outperforms the other ones. Indeed, the NEAT approach obtained only the seventh accuracy, while the other two approaches obtained, respectively, the eighth and the eleventh performance.

Besides the evaluation on the test set, we performed a tenfold cross validation on the training set. We split the training set into tenfolds F_i and we performed ten different sets of ten runs in order to observe which is the behavior of the algorithm when training, validation, and test data change. Table 4 shows the results of the tenfold cross validation. By observing the results we can observe the robustness of the algorithm. In fact, the accuracies obtained by changing the folds used for training, validation, and test are very close; moreover, the standard deviation of the results is very low.

6 Conclusions

In this chapter, we have proposed an ECG classification scheme which implements a neuro-evolutionary approach, based on the joint evolution of the topology and the connection weights together with a novel similarity-based crossover, to aid classification of ECG recordings. The signals were first transformed into the frequency domain by using a Fast Fourier Transform algorithm and then normalized through a gaussian distribution with 0 mean and standard deviation equal to 1. The present system was validated on real ECG records taken from the PhysioNet/Computing in Cardiology Challenge 2011.

A series of tests has been carried out in order to evaluate the capability of the neuro-evolutionary approach to discern between usable and unusable electrocardiograms tele-medically acquired from mobile embedded devices. The results show an overall satisfactory accuracy and performances in comparison with other approaches carried out in this challenge and presented in the literature.

It is important to stress the fact that the proposed method was able to achieve top-ranking classification accuracy despite the use of a quite standard preprocessing step and a very small number of input features. No attempt was made to fine-tune the signal pre-processing and the feature selection steps, when it is well known that these two steps are often critical for the success of a signal classification methods. For this reason, we believe that the proposed neuro-evolutionary approach has a tremendous improvement potential.

Future work will involve the adoption of more sophisticated preprocessing techniques, by working, for example, on a multi-scale basis, where each scale represents a particular feature of the signal under study. Other ideas could regard the study and the implementation of feature selection algorithms in order to provide an optimized selection of the signals given as inputs to the neural networks.

References

1. Data Driven Approach to ECG Signal Quality Assessment using Multistep SVM Classification. Contribution sent to the 38th Physionet Cardiology Challenge (2011)
2. Jokic, S., Krco, S., Delic, V., Sakac, D., Jokic, I., Lukic, Z.: An efficient ECG modeling for heartbeat classification. IEEE 10th Symposium on Neural Network Applications on Electrical Engineering, NEUREL 2010, Belgrade, Serbia, 23–25 Sept 2010
3. Moody, G.B.: Improving the quality of ECGs collected using mobile phones: the 12th annual physionet/computing in cardiology challenge. Comput. Cardiol. Challenge **38**, 273–276 (2011)
4. Silva, K., Moody, G.B., Celi, L.: Improving the quality of ECGs collected using mobile phones: the physionet/computing in cardiology challenge 2011. Contribution Sent to the 38th Physionet Cardiology Challenge (2011)
5. Tat, T.H.C., Chen Xiang, C., Thiam, L.E.: Physionet challenge 2011: improving the quality of electrocardiography data collected using real time QRS-complex and T-wave detection. Contribution Sent to the 38th Physionet Cardiology Challenge (2011)
6. Jiang, W., Kong, S.G.: Block-based neural networks for personalized ECG signal classification. IEEE T. Neural Networks **18**(6), 1750–1761 (2007)

7. Azzini, A., Tettamanzi, A.G.B.: A new genetic approach for neural network design. In: Abraham, A., Grosan, C., Pedrycz W. (eds.) Engineering Evolutionary Intelligent Systems. Studies in Computational Intelligence, vol. 82. Springer, Berlin (2008)
8. Azzini, A., Tettamanzi, A.G.B., Dragoni, M.: SimBa-2: improving a novel similarity-based crossover for the evolution of artificial neural networks. 11th international conference on intelligent systems design and applications (ISDA 2011), pp. 374–379. IEEE (2011)
9. PhysioNet: Research Resource for Complex Physiologic Signals, http://www.physionet.org
10. Camargo, F.: Learning algorithms in neural networks. Technical Report, Computer Science Department, Columbia University (1990)
11. Gurney, K.: An Introduction to Neural Networks. Taylor & Francis, Bristol (1997)
12. Kröse, B., Van der Smagt, P.: An Introduction to Neural Networks. The University of Amsterdam, Amsterdam (1996) ftp://ftp.informatik.uni-freiburg.de/papers/neuro/ann_intro_smag.ps.gz
13. Yao, X.: Evolving artificial neural networks. In: Proceedings of the IEEE, pp. 1423–1447 (1999)
14. Zhang, G.: Neural networks for classification: a survey. IEEE T. Syst. Man Cy. C: Appl. Rev. **30**, 451–462 (2000)
15. Bahrammirzaee, A.: A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert system and hybrid intelligent systems. Neural Comput. Appl. **19**, 1165–1195 (2010)
16. Castellani, M., Rowlands, H.: Evolutionary artificial neural network design and training for wood veneer classification. Eng. Appl. Artif. Intel. **22**, 1165–1195 (2009)
17. Fernández, J., Hervás, C., Martínez-Estudillo, F., Gutiérrez, P.: Memetic pareto evolutionary artificial neural networks to determine growth/no-growth in predictive microbiology. Appl. Soft Comput. **11**, 534–550 (2011)
18. Wang, P., Weise, T., Chiong, R.: Novel evolutionary algorithms for supervised classification problems: an experimental study. Evol. Intel. **4**(Special Issue), 1–14 (2011)
19. Azzini, A., Tettamanzi, A.G.B.: Evolving neural networks for static single-position automated trading. J. Artif. Evol. Appl. **2008**, 1–17 (2008) (Article ID 184286)
20. Azzini, A., da Costa Pereira, C., Tettamanzi, A.G.B.: Modeling turning points in financial markets with soft computing techniques. In: Brabazon, A., O'Neill, M., Maringer D. (eds.) Natural Computing in Computational Finance, Studies in Computational Intelligence, vol. 293, pp. 147–167. Springer, Berlin (2010)
21. Azzini, A., Lazzaroni, M., Tettamanzi, A.G.B.: Incipient fault diagnosis in electrical drives by tuned neural networks. Instrumentation and measurement technology conference, IMTC'06, pp. 1284–1289 (2006)
22. Azzini, A., Marrara, S.: Dermatology disease classification via novel evolutionary artificial neural network. In: Proceedings of the 18th International Conference on Database and Expert Systems Applications, DEXA'07, pp. 148–152 (2007)
23. Azzini, A., Tettamanzi, A.G.B.: Evolutionary ANNs: a state of the art survey. Intelligenza Artificiale **5**, 19–35 (2011)
24. Yao, X., Xu, Y.: Recent advances in evolutionary computation. Int. J. Comput. Sci. Technol. **21**, 1–18 (2006)
25. Floreano, D., Durr, P., Mattiussi, C.: Neuroevolution: from architectures to learning. Evol. Intel. **10**, 47–62 (2008)
26. Hestenes, M., Stiefel, E.: Methods of conjugate gradients for solving linear systems. J. Res. Natl. Bur. Stand. **49**(6), 409–436 (1952)
27. Stanley K., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evol. Comput. **10**, 99–127 (2002)
28. Muhlenbein, H., Schlierkamp-Voosen, D.: The science of breeding and its application to the breeder genetic algorithm (bga). Evol. Comput. **1**, 335–360 (1993)
29. Goldberg, D.: Genetic Algorithms in Search, Optimization and Machine Learning. Kluwer Academic, Boston (1989)
30. Schwefel, H.: Numerical Optimization for Computer Models. Wiley, Chichester (1981)

31. Azzini, A., Dragoni, M., Tettamanzi, A.G.B.: A novel similarity-based crossover for artificial neural network evolution. In: Schaefer, R., Cotta, C., Kolodziej, J., Rudolph, G. (eds.) Parallel Problem Solving from Nature PPSN XI. Lecture Notes in Computer Science, vol. 6238, pp. 344–353. Springer, Berlin (2010)
32. Garcia-Pedrajas, N., Ortiz-Boyer, D., Hervas-Martinez, C.: An alternative approach for neural network evolution with a genetic algorithm: crossover by combinatorial optimization. Neural Networks **19**, 514–528 (2006)
33. Hancock, P.J.B.: Genetic algorithms and permutation problems: a comparison of recombination operators for neural net structure specification. Proceedings of IEEE International Workshop on Combinations of Genetic Algorithms and Neural Networks, COGANN'92, pp. 108–122. IEEE, Baltimore (1992)
34. Xia, H., McBride, J., Sullivan, A., De Bock, T., Bains, J., Wortham, D., Zhao, X.: A multistage computer test algorithm for improving the quality of ECGs. Contribution Sent to the 38th Physionet Cardiology Challenge (2011)
35. Moody, B.E.: A rule-based method for ECG quality control. Contribution Sent to the 38th Physionet Cardiology Challenge (2011)
36. Johannesen, L.: Assessment of ECG quality on an android platform. Contribution Sent to the 38th Physionet Cardiology Challenge (2011)
37. Langley, P., Di Marco, L., King, S., Di Maria, C., Duan, W., Bojarnejad, M., Wang, K., Zheng, D., Allen, J., Murray, A.: An algorithm for assessment of ECG quality acquired via mobile telephone. Contribution Sent to the 38th Physionet Cardiology Challenge (2011)
38. Hayn, D., Jammerbund, B., Schreier, G.: Real-time visualization of signal quality during mobile ECG recording. Contribution Sent to the 38th Physionet Cardiology Challenge (2011)
39. Chudcek, V., Zach, L., Kulek, J., Spilka, J., Lhotsk, L.: Simple scoring system for ECG signal quality assessment on android platform. Contribution Sent to the 38th Physionet Cardiology Challenge (2011)

Self-Organisation and Evolution for Trust-Adaptive Grid Computing Agents

Yvonne Bernard, Lukas Klejnowski, David Bluhm, Jörg Hähner, and Christian Müller-Schloer

Abstract The Organic Computing (OC) initiative aims at introducing new, self-organising algorithms in order to cope better with the complexity of today's systems. One approach to self-organisation is the introduction of agents which are able to continuously adapt their behaviour to changing environmental conditions and thus collectively create an efficient and robust system. In this paper, we introduce an evolutionary approach to an agent which acts autonomously and optimises its behaviour at run-time. The behaviour of the Evolutionary Agent is defined by ten chromosomes. When two agents interact, the inferior agent copies a part of the genes of the more successful agent. Therefore, the most successful gene combination will spread throughout the network. Application scenario for our evaluation is the Trusted Desktop Grid, a distributed system where computing resources are shared by autonomously acting agents.

1 Introduction

Organic Computing (OC) offers a variety of algorithms and mechanisms to manage large-scale, complex systems. Trust as a basic concept can be used to reduce the information uncertainty in such systems and foster the cooperation between their subsystems while securing the robustness of the system regarding misbehaving entities. Our aim is to create provisions which help develop such trustworthy systems, even in safety- or mission-critical environments. One means to achieve this is to control negative emergent behaviour. The exhibition of emergent behaviour

Y. Bernard (✉) · L. Klejnowski · D. Bluhm · J. Hähner · C. Müller-Schloer
Institut für Systems Engineering, FG System- und Rechnerarchitektur Leibniz Universität Hannover, Appelstrasse 4, D-30167 Hannover, Germany
e-mail: bernard@sra.uni-hannover.de; klejnowski@sra.uni-hannover.de; bluhm@sra.uni-hannover.de; haehner@sra.uni-hannover.de; cms@sra.uni-hannover.de

S. Cagnoni et al. (eds.), *Evolution, Complexity and Artificial Life*,
DOI 10.1007/978-3-642-37577-4_14, © Springer-Verlag Berlin Heidelberg 2014

is one of the main characteristics of OC systems. However, such behaviour is at times detrimental to a system, and so has to be limited by analysing, verifying and restraining the interactions between agents during design-time or at run-time and fostering cooperative behaviour. A new research focus in this OC context is Social OC [1], which transfers concepts and knowledge gained from social systems and institutional economics into system architectures. A project within this new research area is the OC-Trust project which aims at improving both the cooperation among subsystems and the robustness regarding malicious behaviour using trust-based algorithms. Ensuring the trustworthiness of subsystems will enable system designers to realise the openness of complex, highly dynamic systems, i.e. the dynamic inclusion of formerly unknown agents.

One approach to the management of complex, dynamic systems is the usage of Adaptive Agents [2]. These agents are able to fit their behaviour to the current situation they observe based on predefined thresholds. These thresholds are tailored to the situation, e.g. if there is a high workload, an agent needs to ask more (and occasionally even less trustworthy) agents for cooperation. For each situation the system designer defines a suited threshold, based on his knowledge of the system at design-time. However, we want these agents to learn and optimise at run-time the threshold best suited in a given situation. Thus, optimising agent behaviour at run-time is crucial for a successful adaptation to changing environmental conditions as requested in the open, dynamic systems we regard. One possibility for optimisation is an evolutionary approach where during agent interaction, a new population arises, continuing life with the dominant genes of the successful agents from the last generation. This completely distributed way of learning and optimisation seems to be worthwhile considering for the agents in our application scenario. Therefore, in this paper, we introduce and evaluate a new class of agents called Evolutionary Agents which optimise the decision making in both worker and submitter role at run-time by imitation of the fitter agents in combination with mutation. We will investigate which strategies evolve as an emergent phenomenon of interaction of Evolutionary Agents.

This chapter is organised as follows: First, in Sect. 2 the application scenario Trusted Desktop Grid, which has been used for the evaluation in this paper, will be introduced and a short overview on related work is given. In Sect. 3, the design and implementation of our Evolutionary Agent will be given. We will evaluate how the Evolutionary Agents behave, both in a homogeneous system and in a heterogeneous system with Adaptive and Egoistic Agents, in Sect. 4. In the last section, we will conclude the paper and give an outlook on our future work.

2 Application Scenario: Trusted Desktop Grid

The application scenario for our research is a desktop grid and volunteer computing system (DGVCS, [3]) with agents acting on behalf of the users. The system is designed as a distributed system without central control. Clients have the capabilities

to be both submitters and workers, which is described below in more detail. The clients are assumed to be heterogeneous in terms of administrative domains, machine resources, usage patterns, volatility, etc. Such a grid is suitable for scenarios where most clients run applications that produce grid jobs and thus are in high demand of computing resources.

In the Trusted Desktop Grid, agents become *submitters* whenever a user application on their machine produces a grid job. These jobs are split into single work units (WUs) which are distributed among available worker clients. The *workers* process them and return the results to the submitters which validate the results. However, these systems are exposed to threats by clients that plan to exploit or damage the system. A worker can, for example, return an incorrect result or not return a result at all. Workers can also refuse to accept a WU. In the area of volunteer desktop grid systems, such cheating behaviour is a serious issue [4]. Here, trust mechanisms can help the agents to estimate the future behaviour of other agents. By extending each client with an agent component and modelling the relations between the agents with a trust mechanism, we expect to counter these threats and thus increase the efficiency of such a system. If, for instance, an agent chooses only those workers that it already had good experiences with, the expected outcome is better. In this paper, the desktop system introduced above has been evaluated in a multi-agent simulation. Agents in this system can act as submitter (i.e. decide which worker to give WUs to) and worker (decide whose WUs to accept) at the same time. However, agents following static rules according to a fixed trust model cannot succeed in a highly dynamic system. Volatile peers, changing trust relations, different workloads and user goals all require the agents to adapt in order to be successful. Moreover, we want the agents to autonomously decide between a more egoistic and a more altruistic behaviour and learn which behaviour is successful in a situation. Additionally, we want our algorithms and mechanisms to provide functionality with a minimal amount of overhead. In the Evolutionary Agent approach introduced in this paper, we show and evaluate how learning and run-time optimisation using an evolutionary approach can be used in our application scenario.

2.1 Related Work

The idea of using evolutionary approaches for Grid Computing has, for instance, been introduced in [5]. The paper shows the development of genes in a simulated evolutionary peer-to-peer overlay scheme. In [6], a peer-to-peer management approach based on natural selection and the survival of the fittest is introduced. In contrast to this, the behaviour of our Evolutionary Agent approach is not only evolutionary but also based on trust. An overview of trust and reputation concepts can be found in [7]. Our agents are given trust and reputation values determined by former interactions. Thus, trust is used as a constitutional part of the agent cooperation relations as discussed in Sect. 2.

Approaches using trust and adaptation algorithms for matchmaking in Grid systems can, for instance, be found in [8] and [9]. The H-Trust system [8] introduces a Desktop Grid system using the Hirsch-Trust index, which has originally been introduced to rank the impact of research papers, to decide on cooperation partners. Organic Grid [9] is based on the autonomous organisation of mobile agents, introducing a tree-based overlay structure for matchmaking decisions.

Our Evolutionary Agent approach is based on the idea of using evolution to model trust relations in Multi-Agent Systems (MAS) as described in [10]. The authors of [10] have introduced a chromosome structure which is able to model trust-based behaviour of agents. With this model, trust creation, destruction and rebuilding can be realised and analysed using a graphical representation of the agents' genes. We have modified this model by creating a new chromosome structure tailored to the Grid agents in our application scenario Trusted Desktop Grid. Thus, we added the relevant observables in the Trustd Desktop Grid, workload, reputation and fitness as perception genes in the chromosome structure of the agent.

3 Evolutionary Agent

The Evolutionary Agent is an approach to run-time optimisation of trust-based interaction. Based on [10], we aimed at creating an Evolutionary Agent model, which enables cooperation and trust-building. This model has been adapted for our application scenario Trusted Desktop Grid in order to make it the basis of the worker and submitter decisions.

In this section, we will introduce the design of this agent type in general as well as how this design is used to make the agent decisions in the application scenario Trusted Desktop Grid in both worker and submitter role.

The Evolutionary Agent consists of a chromosome structure, which contains ten genes. The bit values of the genes represent the alleles of the agent. The combinations of these genes influence the behaviour and decisions of the agent. The genes contain instructions that are interpreted as characteristics of the agents. Each agent follows its own strategy that is induced by the sequence of bits in its chromosome.

Table 1 describes the chromosome and the genes it contains, which encode the behaviour of the agent. The chromosome consists of ten genes: Gene 1 is used to define the general character of the agent. Genes 2–5 define how the agent comes to trust decisions whereas gene 10 marks the actual trust decision. Genes 6–9 define which characteristics of other agents are taken into account for decision making.

Each gene can have the value 1 or 0. Thus, there exist 2^{10} different types of Evolutionary Agents. Gene 1 defines the character of the agent. It decides whether the agent is an egoist (E) ($G_1 = 0$) or a cooperator (C) ($G_1 = 1$) and tries to do its job as well as possible. This means that an egoist will accept work units, if it trusts its partner, but will abort them with a high probability before they are finished. A cooperator will accept work units if the number of work units in its queue is not too

Table 1 Chromosome structure of the Evolutionary Agent

Genes	Alleles	Rules
1	0	E (Egoist: aborts Work Units (WUs) with high probability).
	1	C (Cooperator: tries to process WUs as well as possible).
2	0	Don't involve your own intentions (G_1) into building trust.
	1	Involve your own intentions (G_1) into building trust, given (G_6).
3	0	Ignore the partner's reputation.
	1	Pay attention to the partner's reputation, given (G_7).
4	0	Ignore the fitness of the partner.
	1	Pay attention to the fitness of the partner, given (G_8).
5	0	Ignore the workload of the partner.
	1	Pay attention to the workload of the partner, given (G_9).
6	0	Assume that others are the opposite of your gene (G_1).
	1	Assume that others are the same as your gene (G_1).
7	0	Distrust those who have a relatively high reputation. Trust those who have a relatively low reputation.
	1	Trust those who have a relatively high reputation. Distrust those who have a relatively low reputation.
8	0	Distrust those who have a relatively high fitness. Trust those who have a relatively low fitness.
	1	Trust those who have a relatively high fitness. Distrust those who have a relatively low fitness.
9	0	Distrust those who have a relatively high workload. Trust those who have a relatively low workload.
	1	Trust those who have a relatively high workload. Distrust those who have a relatively low workload.
10	0	Distrust everybody (reject all WUs).
	1	Trust everybody (try to process all WUs).

high and it trusts its partner. Then it will try to process the work units and avoid aborting them. The genes 2 to 5 influence the decisions of the agent if it will trust its partner. Those genes determine the signals which the agent has to pay attention to:

- Its own intentions (Gene 2)
- Reputation of the partner (Gene 3)
- Fitness of the partner (Gene 4)
- Workload of the partner (Gene 5)

The genes 6 to 9 determine how to interpret those signals. If the value of the signalling gene (2–5) is 0, then the corresponding gene (6–9) is ignored.

The following example (cf. Fig. 1) will illustrate the signals and the corresponding behaviour of an agent A_i which decides how to interact with an agent A_j: We assume that the values of gene 2 and gene 4 are 1 and the values of gene 3 and gene 5 are 0. Gene 2 has the value 1, so its own intentions (Gene 1) will be included in the process of building trust. This depends on the value of gene 6. If gene $6 = 0$ the agent assumes that the partner is the opposite of the agent's gene 1. If gene 6 has the value 1, the agent assumes that the partner's gene 1 has the same value as

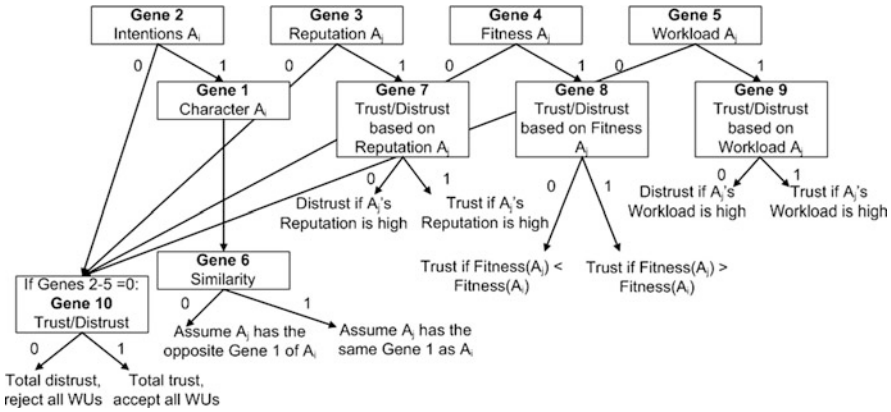

Fig. 1 The chromosome structure of Evolutionary Agent A_i decides how to treat agent A_j

the agent's gene 1. The assumption that the partner is a cooperator will increase trust while the assumption that the partner is an egoist will decrease trust. Since gene $4 = 1$, the agent pays attention to the partner's fitness. If gene $8 = 0$, the agent will trust those which have a lower fitness than itself and distrust those with a higher fitness. If gene $8 = 1$, the behaviour is inverted. The number of signals that recommend trust is normalised with the total number of signals that the agent pays attention to, so that it results in a value between 0 and 1. This value represents the probability that the agent will trust a partner. If an agent will pay attention to none of the four signals, then gene 10 decides if the agent will trust the partner. Based on total trust ($G_{10} = 1$) or total distrust ($G_{10} = 0$), the agent will accept all work units or reject all work units, respectively.

3.1 Gene Initialisation

The genes of each agent are set randomly at creation. For each gene there exists a parameter which has a value between 0 and 1. This parameter determines the probability that the corresponding gene is set to 1 when an Evolutionary Agent is created. Regarding the population of all Evolutionary Agents the value of the parameter corresponds to the expected number of agents which have the corresponding gene equal to 1. The standard value is 0.5, so that each gene of half of the Evolutionary Agents takes the value 1.

3.2 Evolution and Spreading of the Genes

To ensure the evolution of the genes and the corresponding trust strategies, a gene exchange between the agents can occur when they come into contact. In this process,

the genetic instructions will be transferred with a fixed probability from the agents with a higher fitness (cf. Sect. 4.1) to those who have a lower fitness. If two agents interact, the partner with the lower fitness replaces a random part of his chromosome structure with a part of the fitter partner's chromosome. In this procedure, each bit in a chromosome can be replaced independently of the others. Whether a bit of the agent with a lower fitness is replaced by the fitter agent's bit is determined by the recombination probability. In this case the recombination probability was 50 %.

Furthermore, to increase heterogeneity, mutation occurs during the gene replacement. Thus, during the transfer of the genes from the fitter agent to the weaker agent random copying errors (mutations) can arise. The probability that a copying error during a gene replacement occurs is 1 %. This value allows for sufficient heterogeneity without affecting the stability of evolution. Thus, Evolutionary Agents are able to leave local optima in their fitness landscape and have a higher probability to reach the global optimum.

3.3 Worker: Acceptance of Work Units

To decide whether a work unit is accepted or not the chromosome structure is analysed. If the agent pays attention to more than one signal of the partner, each bit is equally taken into account. Partners which send out mixed cooperation signals will be trusted with a corresponding probability. Let's assume that three of the signal genes are used, where two show trust in the partner and the third distrust. Then the agent will trust the partner and accept the work unit with a probability of $\frac{2}{3}$. In this paper, the signal genes are weighted equally.

3.4 Submitter: Distribution of Work Units

In our Grid agent model, a ranking of the suited worker agents is created to distribute the work units [2]. In this paper, this is done by calculating a score of reputation, fitness and workload whereby genes 3, 4 and 5 determine which of these characteristics are included and gene 7, 8 and 9 determine whether the total score will be increased or decreased. After creating the ranking, its own work unit is offered, in order of ranking, to the other agents until one of them accepts or the submitter has to process the work unit itself.

4 Experimental Results

In the experiments presented in this paper, we investigated how the Evolutionary Agents behave with other types of agents and with each other. Section 4.1 introduces the system model and the parameters used in this evaluation. In experiment 1

(Sect. 4.2), we analysed how Evolutionary Agents behave in a heterogeneous system of both Evolutionary and Adaptive Agents. Adaptive Agents have been the most successful agent type in former experiments [2]. Experiment 2 (Sect. 4.3) has been conducted in order to evaluate the behaviour in a homogeneous system of Evolutionary Agents. The figures presented in this section show typical results of ten runs conducted for each experiment. In experiment 3 (Sect. 4.4), we investigated the behaviour of Evolutionary Agents interactin with Egoistic Agents. Egoistic Agents introduce disturbance to the system because they try to exploit the other agents. Egoistic Agents try to distribute their WUs to other agents and thus minimise their own work by aborting the WUs they have accepted from other agents with high probability. This is a misbehaviour we would like the agents to recognise in a distributed fashion and adapt to at run-time.

4.1 System Model

In the experiments, we observed 100 agents over a period of 100,000 ticks (time units). Each gene of a chromosome of an Evolutionary Agent is initialised with one with a probability of 50 %. Additionally, each gene has a recombination probability of 50 %. The first experiment consisted of a population of 100 agents of which 50 were Evolutionary Agents and 50 Adaptive Agents. In the second experiment, we generated a homogeneous population of 100 Evolutionary Agents. In the third experiment, a population of 50 Evolutionary Agents and 50 Egoistic Agents was investigated. These three experiments have been chosen in order to cover both heterogeneous and homogeneous system configurations. The probability that an Evolutionary Agent with $gene_1 = 0$ aborts a work unit was set to 75 % in all experiments.

In all experiments, we measured the aggregated trust value $T_{i,j}$ of agent A_i in agent A_j as a function of the form

$$T_{i,j} = f(rep_j, exp_{i,j}), -1 \leq T_{i,j} \leq 1. \tag{1}$$

where rep_j is the reputation of the agent in the system and $exp_{i,j}$ is an aggregation of the personal experiences A_i has had with A_j in the past. The function contains a weight: the fewer the agents' personal experiences, the higher the reputation weight. Nonetheless, the reputation is always taken into account to a certain degree in order to recognise changes in agent behaviour with more than just knowledge from own experience.

In all experiments, we measured the fitness of the agents. Our fitness function consists of the benefit the agent has from participating in the system as well as the effort he spent in order to reach this benefit:

$$\text{fitness} = \alpha * \text{benefit} + (1 - \alpha) * (1 - \text{effort}) \tag{2}$$

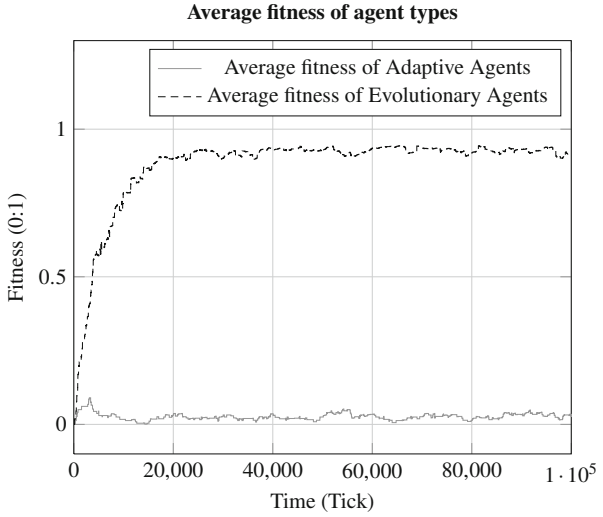

Fig. 2 Results of experiment 1: Average fitness of Evolutionary Agents and Adaptive Agents

The fitness function is evaluated as soon as an agent has finished a job whose WUs have been distributed among the grid workers. The weight between benefit and effort is a factor α between 0 and 1 defined by the system designer. In these experiments, α was 0.8, which means that the benefit is valued much higher than the effort term. The agent fitness is between 0 and 1.

The benefit is the time an agent has saved by distributing the job in the Grid rather than computing it on its own. In order to reach this benefit, the agents need an effort in terms of gaining reputation by calculating WUs for other agents.

4.2 Experiment 1: Evolutionary Agents vs. Adaptive Agents

Figure 2 shows the average fitness of the Evolutionary Agents and the Adaptive Agents. The average fitness of the Evolutionary Agents is much higher than the fitness of the Adaptive Agents. The average reputation of the Adaptive Agents shown in Fig. 3 is higher than the one of the Evolutionary Agents, but still the Evolutionary Agents have a good reputation greater than 0.5. Thus, a good reputation can help to reach a higher fitness, but a high reputation does not necessarily imply a high fitness. Already after tick 30,000 a dominant chromosome structure has evolved: Gene 1, 4, 5, 6, 7, 8 and 9 have the value 1 and gene 2, 3 and 10 the value 0. Figure 4 shows that the workload of the Evolutionary Agents decreases and the workload of the Adaptive Agents increases which means that the Evolutionary Agents successfully distribute the work units to the Adaptive Agents. In other words, the Evolutionary Agents are able to exploit the Adaptive

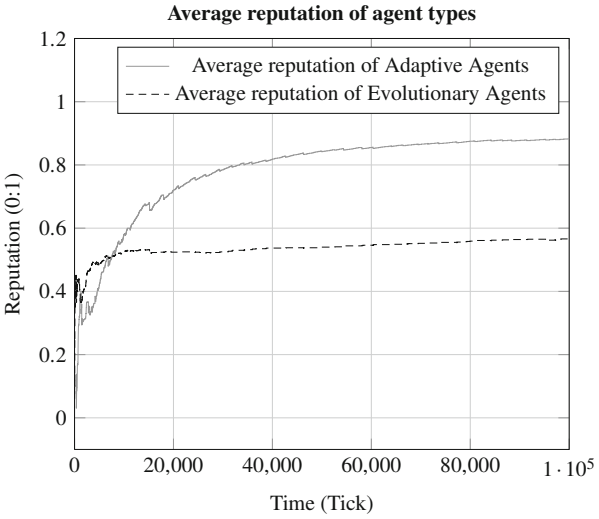

Fig. 3 Results of experiment 1: Average reputation of Evolutionary Agents and Adaptive Agents

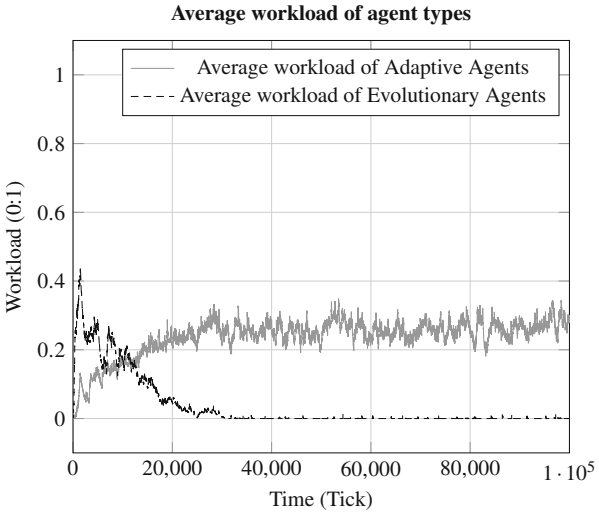

Fig. 4 Results of experiment 1: Average workload of Evolutionary Agents and Adaptive Agents

Agents. This behaviour can also be observed in systems with other agent types. Evolutionary Agents are successful regardless of what the other agents in the system might be because their behaviour continuously adapts to the system configuration. The agents with the highest fitness are copied, thus, the most successful strategy for a given situation evolves and spreads over time. Therefore, in unknown system configurations, an evolutionary approach is worthwhile. This holds as long as there

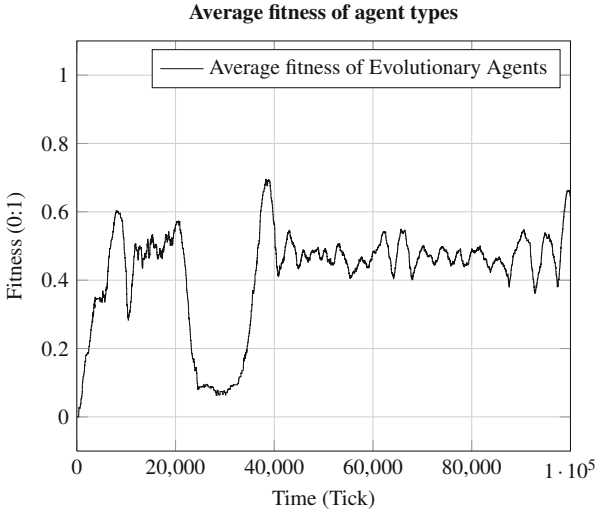

Fig. 5 Results of experiment 2: Average fitness of Evolutionary Agents

exist enough agents using this strategy. Our further experiments have shown that the enforcement of successful chromosomes needs about 25 % of the system population being Evolutionary Agents in order to be fast enough to successfully adapt to the environment.

4.3 Experiment 2: Homogeneous System of Evolutionary Agents

It can be seen from Fig. 7 that value 1 for gene 1 wins from almost immediately after the start of the simulation. This shows that being cooperative is a more successful strategy than being egoistic. In Fig. 5, it is particularly noticeable that at tick 20,000 the fitness strongly drops and rises again at tick 30,000. This matches with the fact that during the same period the value 0 of gene 4 in Fig. 8 has established and so the largest part of the agents will not pay attention to the fitness of the partner in accepting work units. In Fig. 6 it is noticeable that due to the increased workload in this period and due to the lack of trust the agents cannot distribute their work units any more and thus they have to process them on their own.

As soon as the value 1 for gene 4 prevails, the fitness increases and the workload decreases again, because new trust is created between the agents. Thus, it is important to which of the chromosome signals the Evolutionary Agents pay attention: paying attention to the others' fitness is crucial to one agent's success. After gene 4 won through, the chromosome structure stabilises and it is obvious that in this experiment it is not important for the Evolutionary Agents to pay attention

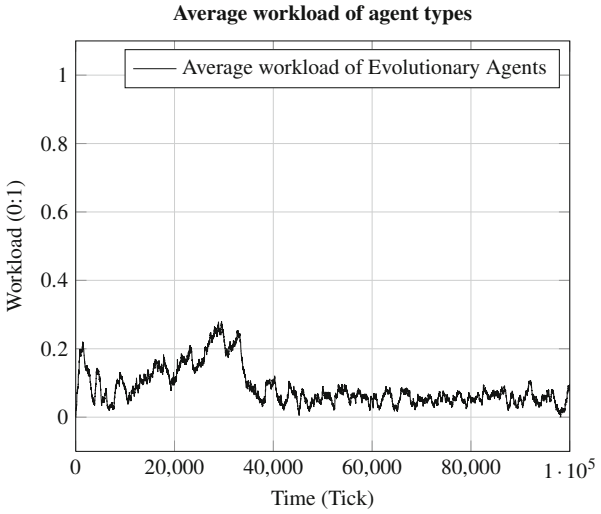

Fig. 6 Results of experiment 2: Average workload of Evolutionary Agents

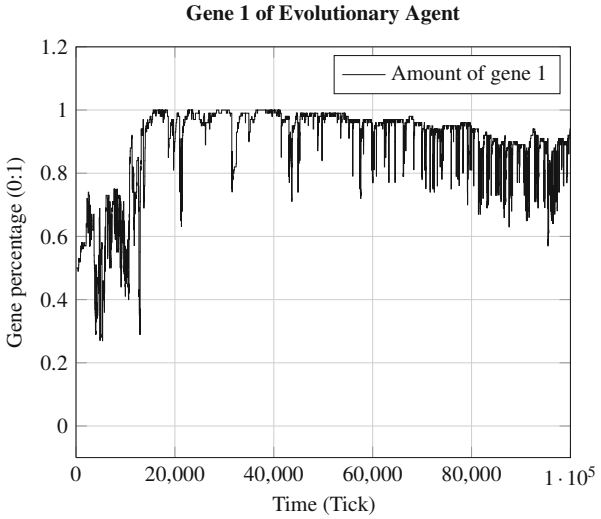

Fig. 7 Results of experiment 2: Amount of gene 1 of Evolutionary Agents

to their own intentions based on gene 1 and to the reputation of the partners. Furthermore, it can be seen in Fig. 8 that the amount of agents with a gene 8 develops to a high value. This means that agents trust agents with a fitness higher than their own, which also leads to a stable population of agents with a high fitness.

Thus, agents pay attention to other agents' fitness (percentage of gene 4 near 1) and imitate agents with a high fitness (percentage of gene 8 near 1).

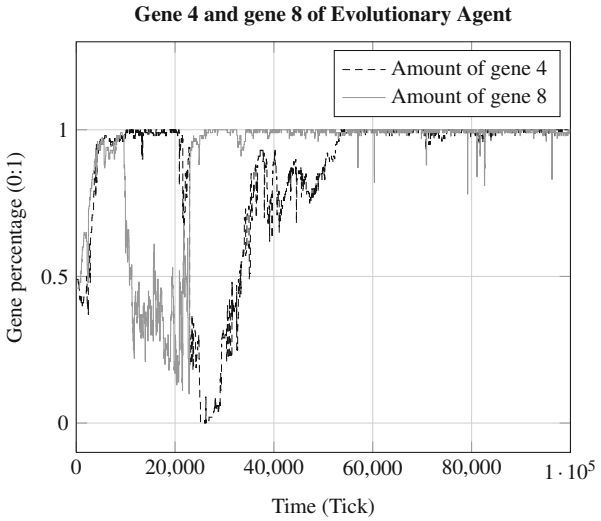

Fig. 8 Results of experiment 2: Amount of gene 4 and gene 8 of Evolutionary Agents

4.4 Experiment 3: Evolutionary Agents vs. Egoistic Agents

In Fig. 9, the average fitness of the Evolutionary Agents and the Egoistic Agents is plotted. Comparing the two fitness curves, it can be seen that the fitness of Egoistic Agents does not exceed a value of 0.1. The fitness of the Evolutionary Agents increases at the beginning and reaches a value of 0.5, but at tick 23,000 the fitness starts to decrease. After a stabilisation in the lower range (< 0.2), at tick 65,000 the fitness starts to increase again. So the fitness curve of the Evolutionary Agents can be separated into three sections. The first section goes from tick 0 to tick 30,000. In this section the Evolutionary Agent has a high fitness. The second section is the interval from tick 30,000 to tick 80,000 in which the Evolutionary Agents have a low fitness. But in the third section from 80,000 to 100,000 the fitness is high again. The decreasing fitness in the second interval is due to the fact that the strategy of the Evolutionary Agents changed at tick 30,000. This can be seen in Fig. 10 in the distribution of gene 5. At tick 30,000 the Evolutionary Agents stop to pay attention to the workload. Therefore, they start to accept WUs from Egoistic Agents, which have a very low workload during the whole simulation. This causes the drop of the fitness of the Evolutionary Agents because they process the WUs of the Egoistic Agents but do not get their distributed WUs processed in return. Later, the strategy changes back and the Evolutionary Agents are able to detect the Egoistic Agents because of their low workload.

Overall, Evolutionary Agents are also quite successful in homogeneous systems, although the amplitude of the fitness is large as there are changes in genes while trying to adapt the chromosome structure to the self-referential fitness landscape in a continuously changing environment (cf. [11] (Chap. 2)).

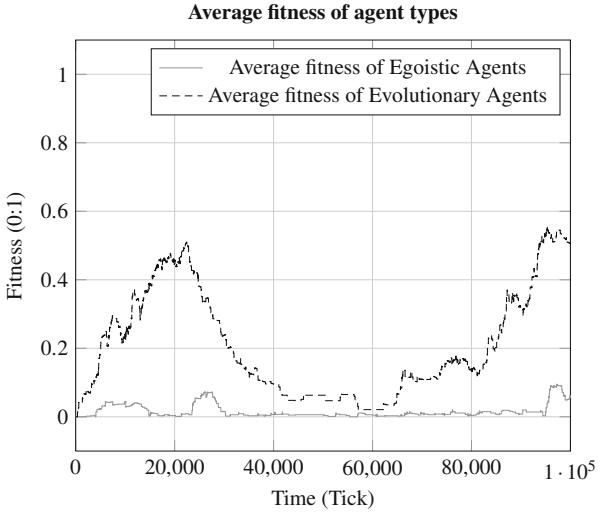

Fig. 9 Results of experiment 3: Average fitness of Evolutionary Agents and Egoistic Agents

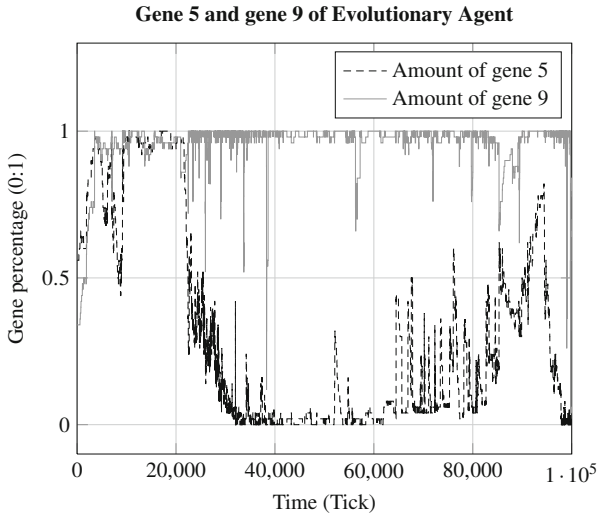

Fig. 10 Results of experiment 3: Amount of gene 5 and gene 9 of Evolutionary Agents

5 Conclusion and Future Work

In this paper, the Evolutionary Agent as an approach to a self-optimising trust-adaptive agent has been introduced. The results of experiment 1 clearly show that, in a heterogeneous system, Evolutionary Agents can achieve a higher fitness than Adaptive Agents, with a good reputation and low workload. In other words,

Evolutionary Agents learn to exploit Adaptive Agents without any malicious behaviour coded in their genes. Experiment 2 shows that Evolutionary Agents are also able to interact in homogeneous systems as well. In experiment 3, we have seen that Evolutionary Agents are also successful in a system with disturbances (Egoistic Agents). They are able to detect the misbehaving Egoistic Agents, although the fitness is not always stable. Surprisingly, Evolutionary Agents do not learn trust-based strategies, but ignore the trust values they have access to. This can be explained because the observation of trust is relevant only if it leads to advantageous behaviour modification which is not necessarily the case in the current experimental setting of our simulation. Therefore, we plan a trust-reactive behaviour extension for the generally well-suited Evolutionary Agent approach to run-time optimisation for Grid Computing agents. In the future of our project, we also aim at investigating further learning techniques like Learning Classifier Systems, Neural Networks or Bayesian Networks in order to convey the optimisation more directly rather than relying on random mutation effects in order to make sure to find an optimal solution over time. Furthermore, we will also investigate the system level of our Trusted Desktop Grid. The agents are able to form the so-called Trusted Communities [2], i.e. agent organisations based on trust mechanisms. They are able to enhance the efficiency and robustness at both individual and system level.

Acknowledgements This research is funded by the research unit "OC-Trust" (FOR 1085) of the German Research Foundation (DFG).

References

1. Müller-Schloer, C., Schmeck, H.: Organic computing - Quo Vadis? In: Müller-Schloer, C., Schmeck, H., Ungerer, T. (eds.) Organic Computing - A Paradigm Shift for Complex Systems, chapter 6.2. Birkhäuser, Basel (2011)
2. Bernard, Y., Klejnowski, L., Hähner, J., Müller-Schloer, C.: Efficiency and robustness using trusted communities in a trusted desktop grid. In: Proceedings of the 2011 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop (SASOW). IEEE Computer Society Press (2011)
3. Choi, S., Kim, H., Byun, E., Baik, M., Kim, S., Park, C., Hwang, C.: Characterizing and classifying desktop grid. In: 7th IEEE International Symposium on Cluster Computing and the Grid, 2007, pp. 743–748. CCGRID 2007 (2007)
4. Anderson, D.P.: Public computing: reconnecting people to science. In: Conference on Shared Knowledge and the Web. Residencia de Estudiantes, Madrid (2003)
5. Amoretti, M.: A framework for evolutionary peer-to-peer overlay schemes. In: Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing, EvoWorkshops 2009, pp. 61–70. Springer (2009)
6. Tyson, G., Grace, P., Mauthe, A., Kaune, S.: The survival of the fittest: an evolutionary approach to deploying adaptive functionality in peer-to-peer systems. In: Proceedings of the 7th Workshop on Reflective and Adaptive Middleware, ARM '08, pp. 23–28. ACM, New York (2008)
7. Mui, L., Mohtashemi, M., Halberstadt, A.: A computational model of trust and reputation. In: Proceedings of the 35th Annual Hawaii International Conference on HICSS System Sciences, pp. 2431–2439, 7–10 Jan 2002

8. Zhao H, Li, X.: H-trust: a robust and lightweight group reputation system for peer-to-peer desktop grid. In: 28th International Conference on Distributed Computing Systems Workshops. ICDCS '08, pp. 235–240, June 2008
9. Chakravarti, A.J., Baumgartner, G., Lauria, M.: The organic grid: self-organizing computation on a peer-to-peer network. IEEE T. Syst. Man Cy. A: Syst. Hum. **35**(3), 373–384 (2005)
10. Macy, M.W., Skvoretz, J.: The evolution of trust and cooperation between strangers: a computational model. Technical Report, October (1998)
11. Cakar, E.: Population-based runtime optimisation in static and dynamic environments. Ph.D. thesis, Leibniz Universität Hannover (2011)

Honest vs Cheating Bots in PATROL-Based Real-Time Strategy MMOGs

Stefano Sebastio, Michele Amoretti, Jose Raul Murga, Marco Picone, and Stefano Cagnoni

Abstract Massively multiplayer online games (MMOGs) are being increasingly successful, since they allow players to explore huge virtual worlds and to interact in many different ways, either cooperating or competing. To support the implementation of ultra-scalable real-time strategy MMOGs, we are developing a middleware, called PATROL, that is based on a structured peer-to-peer overlay scheme. Among other features, PATROL provides AI-based modules to detect cheating attempts, that the decentralized communication infrastructure may favor. In this work we illustrate how we implemented honest and cheating autonomous players (bots). In particular, we show how honest bots can detect cheating bots in real-time, using strategies based on neural networks.

1 Introduction

Most research on multiplayer online games (MMOGs) focuses on scalability and high speed, but other issues such as the chance of cheating have an equally large practical impact on game success. There are significant technical barriers to achieving all these properties at the same time, and few existing games do so. Our open source middleware for the development of real-time strategy (RTS) MMOGs,

S. Sebastio
IMT Institute for Advanced Studies Lucca, Lucca, Italy
e-mail: stefano.sebastio@imtlucca.it

M. Amoretti (✉)
Centro Interdipartimentale SITEIA.PARMA, Università degli Studi di Parma, Parma, Italy
e-mail: michele.amoretti@unipr.it

J.R. Murga · M. Picone · S. Cagnoni
Dip. di Ingegneria dell'Informazione, Università degli Studi di Parma, Parma, Italy
e-mail: josemurgav@gmail.com; picone@ce.unipr.it; stefano.cagnoni@unipr.it

S. Cagnoni et al. (eds.), *Evolution, Complexity and Artificial Life*,
DOI 10.1007/978-3-642-37577-4_15, © Springer-Verlag Berlin Heidelberg 2014

called PATROL (http://code.google.com/p/patrol/), integrates several modules, each of which is highly specialized in one aspect of the game.

In our previous work [1] we focused on the module that enables peer-to-peer connectivity and communication, and on the module for detecting cheating behaviors. The former allows one to implement ultra-scalable RTS MMOGs, where each player's software installation is a node of a fully distributed structured overlay network scheme, which guarantees efficient data sharing as well as fair and balanced workload distribution among participants. The latter, based on Multi Layer Perceptron (MLP) neural networks, allows each node to detect other players' cheating behaviors.

In this work we present the Rule Engine, the PATROL module that allows one to enforce both general and specific rules into the nodes and matches game events with existing rules. The Rule Engine allowed us to implement autonomous playing agents, called *bots*, that are able to play against one another, in an RTS game where participants place and move units and structures (generally speaking, *resources*) to secure areas of the virtual world and/or destroy the assets of their opponents. We show how honest bots are able to detect cheating ones in real-time, thanks to the cheating detection module.

The chapter is organized as follows. In Sect. 2 we summarize some recent research work in the context of peer-to-peer RTS MMOGs. In Sect. 3 we describe the PATROL architecture, with details on the rule engine and on the cheating detection strategy. In Sect. 4 we present the example of the RTS MMOG we have implemented, based on the proposed architecture, where autonomous bots play against one another. In Sect. 5 we illustrate experimental results, focusing on the performance of the module for intelligent cheating detection, presenting and discussing experimental results. Finally, in Sect. 6, we conclude the chapter by describing plans for further extension of our work.

2 Related Work

An MMOG needs a messaging infrastructure for game actions and player communication. To this purpose, possible paradigms are Client-Server (CS), Peer-to-Peer (P2P), Client-Multi-Server (CMS), or Peer-to-Peer with Central Arbiter (PP-CA) [2]. Each solution has pros and cons, with respect to robustness, efficiency, scalability, and security. In particular, when the architecture of the game is decentralized (e.g., P2P), with a large number of players, then facing malicious behaviors is particularly challenging.

In [2], the authors propose a Mirrored-Arbiter (MA) architecture that combines the features of CMS and PP-CA. This architecture not only provides all the benefits of PP-CA but also solves the main problems in PP-CA by using interest management techniques and multicast. Clients are divided into groups, each group being handled by an arbiter that maintains a global state of the game region and takes care of the consistency issue. When the arbiter receives an update from a client which conflicts

with its game region state, it ignores the update and sends the correct region state to all clients in the group. The authors implemented a multiplayer game called "TankWar" to validate the design of the proposed MA architecture. In our opinion, such a scheme is complex in the decision of the arbiters and their group assignments, and does not guarantee high scalability and security. Indeed, an arbiter may be a cheating node itself, which compromises the game for a large number of nodes.

In [3], the authors present a Peer-to-Peer (P2P) MMOG design framework, Mediator, based on a super-peer network with multiple super-peer (Mediator) roles. In this framework, the functionalities of a traditional game server are distributed, capitalizing on the potential of P2P networks, and enabling the MMOG to scale better with respect to both communication and computation. Mediator integrates four elements: a reward scheme, distributed resource discovery, load management, and super-peer selection. The reward scheme differentiates a peer's contribution from its reputation and pursues symmetrical reciprocity as well as discouraging misdemeanors. The authors suggest to adopt the EigenTrust reputation management algorithm [4] and the DCRC anti-free-riding algorithm [5] as possible implementations for the reward scheme. Unfortunately, such schemes are complex and bandwidth-consuming.

The main component on which our work is focused is the game engine. In our opinion, games are made of *nouns* (i.e., elements of the game, and variables related to them), *verbs* (the actions that players and player stand-ins can enact), and *rules* (limiting the scope of the nouns and creating relationships and interactions between them; limiting also which verbs can be enacted, when and in which context). Current programming paradigms do not provide an appropriate language for expressing these structures. Object-Oriented Programming (OOP) is very good at representing different types of objects ("nouns") and the relationships they may have between each other, with minimal duplicated code or wasted work. Unfortunately, OOP dictates that each class must encapsulate methods, i.e., actions that are strictly related to the class itself. Thus, OOP is not suitable for expressing verbs and rules as entities separated from nouns. In general, imperative languages (that are mostly used in game programming) are not good for clearly expressing verbs and rules. Instead, declarative languages, like Prolog, are much more suitable.

Currently, one of the best known rule engines is Drools Expert [6] that uses the rule-based approach to implement an Expert System and is more correctly classified as a Production Rule System. A Production Rule System is Turing complete, with a focus on knowledge representation to express propositional and first order logic in a concise, non-ambiguous and declarative manner. The core of a Production Rule System is an Inference Engine that is able to scale to a large number of rules and facts. The Inference Engine matches facts and data against Production Rules—also called Productions or just Rules—to infer conclusions which result in actions. A Production Rule is a two-part structure that uses First Order Logic for reasoning over knowledge representation. There are a number of algorithms used for Pattern Matching by Inference Engines including Linear, Rete, Treat. Drools implements and extends the Rete algorithm and is a sound product, but it is quite large and cannot

be installed on portable devices. For this reason, we decided to adopt tuProlog [7], as discussed in the next section.

3 PATROL Middleware

In order to increase security, the game infrastructure should properly manage the interaction events among nodes. In RTS games, the most frequent events are those for: (1) moving resources, (2) receiving updates about the virtual world, and (3) submitting the attacks. Our PATROL middleware manages these events through protocols that are appropriate for maintaining an adequate level of efficiency and security.

PATROL provides the following modules (illustrated in Fig. 1):

- GUI/GamePeer Connector
- Overlay Manager
- Rule Engine
- Cheating Detector

Since the Overlay Manager and the Cheating Detector have already been described in detail in previous work [1], here we just recall them shortly and we devote more space to the Rule Engine. The GUI/GamePeer Connector decouples the (game-specific) GUI from the GamePeer, which integrates the three previously listed general-purpose modules. For lack of space, we omit its description, but we emphasize that GUI decoupling also allows one to implement games for mobile devices, where only the visualization may be running locally, while most computation processes can be executed remotely. Such an approach is used, for example, by the GaiKai cloud gaming platform (http://www.gaikai.com).

3.1 Overlay Manager

PATROL's Overlay Manager adopts the Chord P2P overlay scheme [8] to support fair and robust information sharing among available players. Chord is a highly structured P2P architecture where all peers are assigned the same role and amount of work. It is based on the Distributed Hash Table (DHT) approach for an efficient allocation and recovery of resources. The overlay network in PATROL also supports a distributed algorithm for cheating detection, based on feedbacks among peers and AI tools such as neural networks. This approach allows the peers to dynamically recognize malicious behaviors, without the need of specific and centralized control components.

Chord [8] is probably the best known peer-to-peer protocol based on the Structured Model (SM), which uses DHTs as infrastructures for building large scale applications. Data are divided into *blocks*, each identified by a unique *key* (a hash

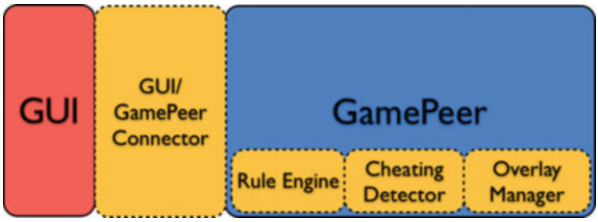

Fig. 1 A PATROL-based gaming node: PATROL modules are those with dashed border

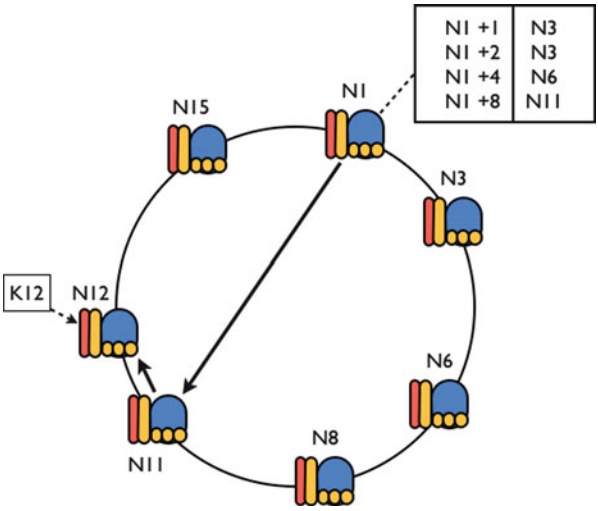

Fig. 2 The finger table entries for node $N1$ and the path taken by a query from $N1$, searching for key $K12$ using the scalable lookup algorithm

of the block's name) and described by a *value* (typically a pointer to the block's owner). Each peer is assigned a random ID in the same space of data block keys and is responsible for storing (key,value) pairs for a limited subset of the entire key space.

According to Chord's lookup algorithm, each node n maintains a routing table, called the *finger table*, with up to m entries. The ith entry in the table at node n contains the identity of the first node s whose identifier follows n by at least 2^{i-1}; i.e., $s = \text{successor}(n + 2^{i-1})$, where $1 \leq i \leq m$ and all the arithmetic is module 2^m. We call node s the ith *finger* of node n, and denote it by $n.finger[i]$. A finger table entry includes both the Chord identifier and the IP address (and port number) of the relevant node. Figure 2 illustrates the scalable lookup algorithm based on finger tables. In general, if node n searches for a key whose ID falls between n and the successor of n, node n finds the key in the successor of n; otherwise, n searches its finger table for the node n' whose ID most immediately precedes the desired key, and then the basic algorithm is executed starting from n'. It is demonstrated that the

number of nodes that must be contacted to find a successor in an N-node network is $O(\log N)$ [8] in the majority of cases.

PATROL distributes the responsibility to maintain knowledge about resources (i.e., items, war units, and structures) uniformly among the peers, using the DHT to share information about whom is responsible for what (each peer is responsible for a subset of the key space). In a game, each existing resource has a position in the virtual world. Such a position is hashed, and the resulting key is assigned to the peer whose key subset includes the resource key. It is very unlikely that two resources that are close in the virtual world have keys that are also close in the key space (and vice versa). Moreover, Chord provides data replication, in order to improve robustness against unexpected node departures.

As a consequence, the control over a region of the game space is distributed among several peers, thus limiting the damages that a hacked peer can do. Moreover, our approach is much more robust than existing decentralized solutions, because the departure of a node does not affect the games too much, thanks to the proactive data replication dictated by the Chord protocol.

3.2 Rule Engine

In general, a rule engine is a software system that, depending on the context, decides which rules to apply, and computes the result of their application, that may be a new knowledge item, or an action to perform. Usually, a rule engine includes the following components:

- a *rule base*, containing production rules whose structure is WHEN ⟨conditions⟩ THEN ⟨actions⟩;
- a *knowledge base*, also known as *work area*, that contains known facts;
- an *inference engine*, for processing rules.

Rules operate on facts of the knowledge base, that is dynamic as it can change over time, with new facts being added, and old facts being removed. Conditions of production rules are evaluated against facts. If a condition is true, the resulting action is the insertion of a new fact in the knowledge base, and possibly an action on the environment (e.g., an action within the game).

PATROL's Rule Engine (from now on, RE) can be used for implementing several RTS MMOGs: it is sufficient to set the appropriate rule base and knowledge base. Rules and facts must be written in Prolog, chosen because of its intuitiveness. RE's inference engine is based on tuProlog, a Java-based lightweight Prolog reasoner for Internet applications and infrastructures [7]. tuProlog provides a straightforward API to embed Prolog programs within Java code, or to read existing Prolog expressions from a file or from a database. Once one or more Prolog *theories* (i.e., ensembles of rule base and initial knowledge base) have been acquired, it is possible to use them to evaluate facts and derive new facts.

The RE can be used to decide which actions are allowed to the player, depending on his/her state and on the state of the game. Moreover, the RE can be used to implement bots, i.e., autonomous playing agents that allow, for example, to test game strategies before entering a game against other real players. A bot must be able to make decisions in all typical RTS situations, such as: resource accumulation, resource purchasing or building, resource improvement, displacement of mobile resources, attack against an enemy's resources, defense from an enemy's attack, goal checking.

The RE includes a PrologEngine class that provides methods for setting and managing a theory, and for solving queries. Such a class can be specialized (by means of inheritance) into different classes, each referring to an aspect of the game. Such specialized classes can be reused with different theories, and within different RTS MMOGs.

3.2.1 Game Events

The system uses a bootstrap server to support peers in joining the network (which includes authentication, as well as Chord initialization) and configuring themselves for entering a game. In this way the bootstrap server has control over the accounts of the players and consequently provides a basic level of security.

Information about the virtual world may not be granted indiscriminately to any peer. Each peer has its own resources, which are placed in different positions of the virtual world, and has the right to receive information that refers to areas that are within the field of view of such resources, according to the rules of the game. Periodically, each peer needs to update its view on the virtual world. To do so, it sends specific requests to peers that are responsible for the positions that are visible. Before responding to such a request, peer N_j, that we suppose to be responsible for position (x, y, z), checks its cache for updated information, and sends a request to verify the credentials for another peer N_k. If everything is ok, it finally sends the response message.

To perform any action that involves a change of game state, players must submit a request to the responsible of the location that is affected by the action. For example, suppose that a peer N_k selects a resource to be moved in the virtual world; to perform the action "move resource to position (x, y, z)" the peer must submit the request to the node responsible for the key resulting from the hash of that position, i.e., $h(x, y, z)$. The responsible for the key is discovered by means of the lookup strategy defined by the Chord protocol, discussed above and illustrated in Fig. 3. The peer j that must become the new responsible for the moved resource of peer N_k searches for the manager of the resource's current position (declared by peer N_k). Such a peer is discovered by means of the hash of the current position. Thus the old manager checks in its cache whether it has the information on peer N_k and whether this information corresponds to what was declared to N_j. If the check is successful, peer N_j can decide, according to the game rules and considering the time elapsed between

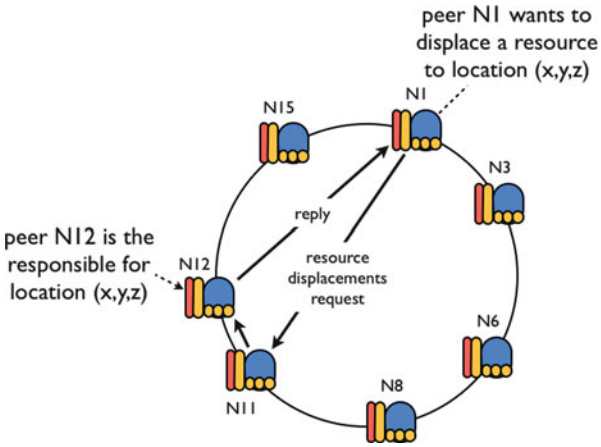

Fig. 3 Submission of an action request to the responsible of the location that is affected by the action. N_1 knows N_{11}, which in turns knows N_{12}, i.e., the peer that must serve as the responsible for location (x, y, z)

the changes of game state following the transition between the two positions, if it can accept the move and execute it, becoming the new responsible for the resource. If the position declared by N_k is not true, the request for resource displacement submitted by peer N_k is ignored and the state of the game remains the same.

While troops can be moved asynchronously by each player, attacks can be either asynchronous or synchronous (i.e., with a turn-based approach). In case of synchronous attacks, knowing the decisions of other players before submitting his own move may be a considerable advantage for a player. But, of course, this would be unfair.

To avoid cheating, PATROL uses request hashing, a mechanism that is widely adopted in other P2P architectures and derives from distributed security systems. Players who submit their decisions have to send a hash of the message describing the attack concatenated with a *nonce*, which is a use-once random value, chosen by the first player that submits a decision. The nonce is used to prevent a cheater from storing in a table all matches between hash values and attack decisions, revealing the decisions of honest players. The last player that submits its decision may send it manifestly. At this point, all players that have previously sent the hash must send their nonce and attack decision manifestly. Thus, other players can re-calculate the hash and verify that it corresponds to what was previously declared. The properties of hash functions guarantee that it is almost impossible for two different attacks to have the same hash, and therefore for a player to submit a different attack, with respect to the hashed one.

3.3 Cheating Detector

PATROL provides a good level of security for the overall state of the game. However, the DHT does not prevent the game from offering cheaters (provided with hacked clients) the possibility to alter the information for which they are responsible. In an RTS game, a modified client that saves a history of recent attacks and their outcomes may estimate the current level of resources available to other players and take advantage over them.

Using artificial intelligence techniques, a PATROL-based peer can detect anomalies in the behavior of other peers, compared to typical behavioral profiles, by means of temporal analysis of interaction events. Moreover, using the power of direct communication typical of P2P approaches, a peer may ask other peers their opinion about a given peer in order to improve the evaluation process.

Peer x calculates the probability $P_x\{y\}$ that peer y is cheating. Then x sends a request to peers that have interacted with y, in order to match their probabilities and understand whether y is considered to be a cheater: $P_i\{y\}\ \forall i \neq x, y$. If the global probability exceeds a certain threshold, there is the option to contact other peers and the bootstrap server to promote a collective motion against the cheating player in order to ban him from future games. If all peers agree with the "ban proposal," peer y is gracefully disconnected from the Chord ring. Every player builds its own "opinion" about other players, from its own interactions. By combining these opinions, before emitting a ban proposal, the probability that the collective decision is erroneous is low.

The Cheating Detector module (Fig. 1) analyzes all action events coming from an opponent. The opinions of the other players are requested only at the end of the local evaluation process, if the peer estimates the probability of cheating to be high. Of course, the peer must be careful since other peers may provide false reports related to their interactions with a given peer.

There are different strategies for a peer to learn from a sequence of events: sequence recognition, sequence playback, and temporal association. Among others, we focused on MLPs, implemented by means of the Weka library [9]. We analyzed their features in detail in our previous work [1].

Input values come from a registered sequence of time/value pairs representing the resources used by an opponent player during the last three interactions with the player that is recording logs. In our opinion, including three interactions is a reasonable choice. A longer trace would not permit the detection of a misleading behavior in an acceptable time. So, the input to the net consists of three timestamps and the three corresponding resource values (defense resources are marked with a minus sign). The timestamp is referred to the begin of the game session. In output we have a 0 if the sequence of actions corresponds to a honest player, otherwise—if the sequence corresponds to a cheater—we have a value close to 1, the larger the more a player is cheating.

MLPs are the traditional multi layer neural networks trained by the backpropagation algorithm. Weights are updated using the online algorithm, i.e., re-arranging

the weights after each epoch, i.e., a learning iteration during which all examples are processed by the network. Neural networks have been used in similar contexts, e.g., for the detection of cheating players in first-person shooters [10], but also with slightly different objectives, like the detection of bots that play in place of human participants [11, 12].

4 Game Bots

We have extended the PrologEngine class of the RE into specialized classes, to implement an RTS MMOG with space setting. In such an RTS MMOG, players have to find and conquer all the planets that are in the game space.

Players are provided with a mine resource that allows them to make money for buying two types of resources: defense and attack. The resources for attack (starships) are used to explore the virtual world, to the purpose of finding the planets and to tackle the starships of other players. Every resource has an associated velocity and field of view. The resources for defense are used to protect the planets owned from incoming attacks of other players' starships.

Thus, we have implemented the following modules (corresponding to Java classes): ExtractionEngine for managing the extraction of mineral resources of a planet, BuyResourceEngine for purchasing resources, MovementEngine for moving mobile resources (like starships), VisibilityEngine for deciding if a resource (e.g., a planet, or an enemy's starship) is visible to the player, GameEvolutionEngine for deciding the next operation depending on the current state (own state, and game state), GameEngine for checking if intermediate or final goals have been met.

For testing purposes, we have implemented *game bots*, i.e., virtual players that automatically handle game tasks—extracting mineral resources, purchasing resources, moving mobile resources, etc. A game bot is a weak AI agent which, for each instance of the program, may play against other bots and/or human players at the same time, either over the Internet, on a LAN or in a local session [13]. Advanced bots feature machine learning capabilities for dynamic learning of opponents' patterns and of previously unknown maps. Using bots to control characters that are supposed to be controlled by human players is incidentally against the rules of all of the current main Massively Multiplayer Online Role-Playing Games (MMORPGs), such as Ultima Online (http://www.uo.com) and World of Warcraft (http://eu.battle.net/wow/). However, the virtual worlds of all MMOGs are characterized by a number of *non player characters (NPC)* with different objectives, either collaborative or competitive with regard to players. For this reason, they can be considered as artificial life systems.

Based on the previously listed engines, game bots pass through three different phases:

Fig. 4 Time distribution of bots' actions, considering all the games that have been played to collect a significant number of actions

1. resource accumulation
2. space exploration
3. planet conquest

These phases are repeated in an infinite loop. The time periods each bot spends in such phases are random variables.

We have defined two different types of bot profiles: honest and cheater. The latter reproduces the behavior of a hacked client. It owns a mine which is five times as productive as the others and more initial money. Moreover, it has a halved cycle decision period (i.e., it can take more decisions at the same time).

In Fig. 4, we report the distribution of honest and cheating bots' actions during the recorded games. On the horizontal axis we have the time at which the action is performed, considering 0 as the start time of the game, while on the vertical axis there is the value associated with the used resource. Here we can note that cheater bots, thanks to the speed hack, prefer to first explore the space and to perform their actions later than the honest bots. Moreover, the value associated with their resources is higher since the more money is available, the more they can spend for buying higher-valued resources.

5 Experimental Evaluation

We have collected 2,800 player profiles, each consisting of a sequence of three actions performed by the player. The choice of three moves as profile duration appears as a good trade-off between quickness and accuracy of evaluation of the opponent's behavior. The overall dataset has been split into a training set of 1,300

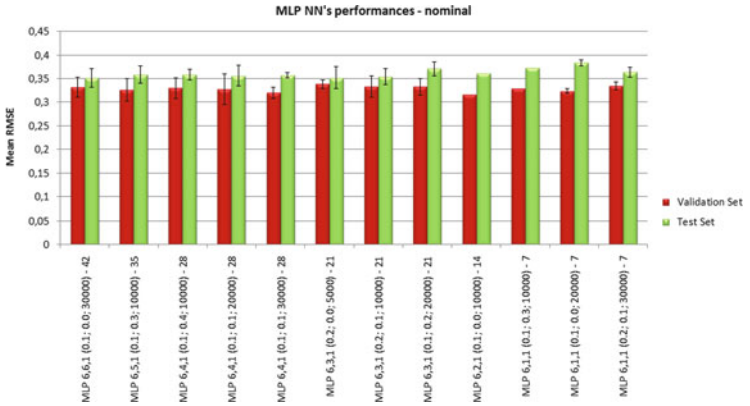

Fig. 5 The best RMSE values obtained with the MLP neural networks requested to provide a "cheating indicator" $\in [0, 1]$

profiles, a validation set of 100 profiles and a test set of 1,400 profiles. For all these sets, 50 % are profiles related to honest players, and 50 % are profiles related to cheating players.

We have defined different configurations for the MLP neural networks, and we have compared these by evaluating their *Root Mean Square Error (RMSE)*. All the MLP neural networks we have considered have three layers, with x, y, z neurons each. In particular, we have used the following configurations: 6, 6, 1; 6, 5, 1; 6, 4, 1; 6, 3, 1; 6, 2, 1; 6, 1, 1. Such networks have been compared by measuring their RMSE with respect to the validation set. Those that have shown the best performances have been selected and tested using the test set. For all six configurations we have considered different values for the seed, the momentum, the learning rate, and the epoch number. In detail, we have used five seeds over which we have computed the average, σ^2 and σ of the RMSE. We have also computed the 95 % confidence interval using Student's T-test. The momentum and the learning rate have been varied with step equals to 0.1 in the range $[0, 1]$, and for the epoch number we considered the following values: {5,000; 10,000; 20,000; 30,000; 40,000; 50,000; 60,000}.

We trained 6 (different number of hidden nodes) $\times$10 (learning rates) $\times$11 (momentum) $\times$7 (epochs) $\times$5 (seeds) networks. i.e., we have trained 23,100 neural networks, $\times$2 (since we have used both the continuous and the binary versions).

Figures 5 and 6 compare the RMSE values of the best MLP neural networks we have found during the experiments, considering the validation set and the test set. Figure 5 refers to experiments where the networks are requested to provide a "cheating indicator" $\in [0, 1]$. Instead, the results in Fig. 6 are related to experiments where the output of the network is boolean (honest or cheating).

In the bot's configuration file it is possible to modify different parameters that are related to a cheating behavior, e.g., the period of money extraction from the mine, the amount of money generated by the mine, the money and the resources available at the beginning of the game, etc. Thus, if we consider that a honest player has, for

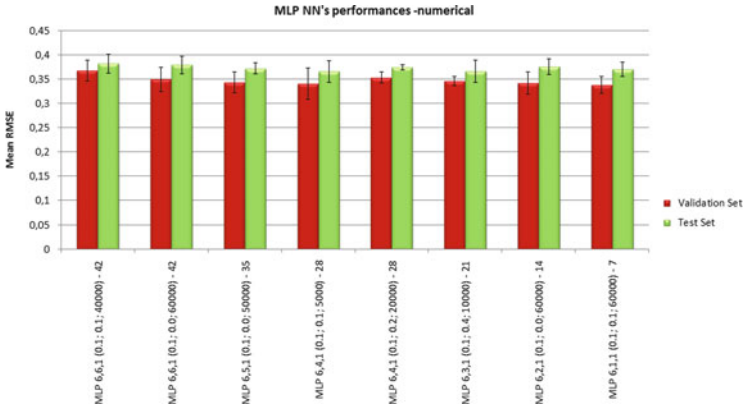

Fig. 6 The best RMSE values obtained with the MLP neural networks, where the output is a boolean (honest or cheating)

instance, an initial value for resources of 20 and a cheater of 100, it is possible to assign a cheating factor of 0.8. This is the value that a perfect evaluation of a neural network reaches in the continuous version, unlike the binary versions.

We can observe that the best performing MPL networks are those with smaller RMSE values and standard deviations on the test set, i.e.,:

- MLP $6, 6, 1(0.1; 0.0; 30{,}000) - 42$: 0.35188 ± 0.006782
- MLP $6, 3, 1(0.2; 0.0; 50{,}00) - 21$: 0.35176 ± 0.005714

where MPL $x, y, z(lr; m; e) - f$ means MLP with three layers of x, y, z neurons, respectively, learning rate lr, momentum m, epoch number e and f weights.

In general, MLPs have yielded good performance on the test set, in terms of error percentage: detected cheating actions are between the 83 % and 85 % of the total number of actions during a game. The RMSE is computed by taking into account the difference between the real output and the predicted one. Moreover, we use the network as a binary classifier such that, when the output is over 0.5, we consider the opponent to be a cheater (since the output is in the range of $[0, 1]$). Thus, the error rate recorded for cheating identification is about 15 %.

6 Conclusions

In this work we illustrated the most recent improvements of our PATROL framework for creating peer-to-peer online RTS games, characterized by a high degree of robustness, efficiency, and effectiveness against cheating behaviors. In particular, we focused on the Rule Engine, which allows us to enforce the rules of a game, and also to develop autonomous virtual players (bots). We have shown how cheating bots can be detected by means of a PATROL module that uses neural networks.

Tests have been encouraging, since cheating detection had a success rate of 85 %. It is possible to envisage the use of other means of temporal analysis based on neural networks (such as *Real Time Recurrent Learning* and *Context Units* like Elman and Jordan nets) and on other techniques. It would also be possible to investigate the effects of adding a component capable of evaluating the *trust of peers* based on the past history of players.

Acknowledgments We would like to thank the anonymous reviewer and Dr. Alberto Lluch Lafuente of IMT Lucca for their precious comments that helped us to improve this work.

References

1. Picone, M., Sebastio, S., Cagnoni, S., Amoretti, M.: Peer-to-peer architecture for real-time strategy MMOGs with intelligent cheater detection. In: Proceedings of DIstributed SImulation & Online gaming (DISIO), co-located with 3rd ICST/ACM International Conference on Simulation Tools and Techniques (SIMUTools 2010), Torremolinos (2010)
2. Yang, L., Sutinrerk, P.: Mirrored arbiter architecture:a network architecture for large scale multiplayer games. In: Summer Computer Simulation Conference (SCSC 2007), San Diego (2007)
3. Fan, L., Taylor, H., Trinder, P.: Mediator: a design framework for P2P MMOGs. In: Proceedings of NetGames'07, Melbourne (2007)
4. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The EigenTrust algorithm for reputation management in P2P networks. In: Proceedings of the 12th Int'l Conference World Wide Web, Budapest (2003)
5. Gupta, M., Judge, P., Ammar, M.: A reputation system for peer-to-peer networks. In: Proceedings of the 13th ACM NOSSDAV workshop, Monterey (2003)
6. M. Bali, Drools JBoss Rules 5.0 Developer's Guide, Packt Publishing, Birmingham (2009)
7. Denti, E., Omicini, A., Ricci, A.: Multi-paradigm Java-Prolog integration in tuProlog. Sci. Comput. Program. **57**(2), 217–250 (2005)
8. Stoica, I., Morris, R., Karger, D., Frans Kaashoek, M., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup service for Internet applications. In: Proceedings of ACM SIGCOMM '01 Conference, San Diego (2001)
9. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. ACM SIGKDD Explor. Newsl. **11**(1), 10–18 (2009)
10. Galli, L., Loiacono, D., Cardamone, L., Lanzi, P.L.: A cheating detection framework for unreal tournament III: a machine learning approach. In: Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG 2011), Seoul (2011)
11. Gianvecchio, S., Wu, Z., Xie, M., Wang, H.: Battle of botcraft: fighting bots in online games with human observational proofs. In: Proceedings of CCS09, Chicago (2009)
12. Prasetya, K., Zheng, W.: Artificial neural network for bot detection system in MMOGs. In: Proceedings of the 9th Annual Workshop on Network and Systems Support for Games (NetGames '10), Taipei (2010)
13. Kaminka, G.A., Veloso, M.M., Schaffer, S., Sollitto, C., Adobbati, R., Marshall, A.N., Scholer, A., Tejada, S.: GameBots: a flexible test bed for multiagent team research. Commun. Assoc. Comput. Mach. **45**(1), 43–45 (2002)

Part V
Evolution

Distribution Search on Evolutionary Many-Objective Optimization: Selection Mappings and Recombination Rate

Hernán Aguirre, Akira Oyama, and Kiyoshi Tanaka

Abstract This work studies distribution search in the context of evolutionary many-objective optimization where, in addition to good convergence towards the optimal Pareto front, it is required to find a set of trade-off solutions spread according to a given distribution. We particularly focus on the effectiveness of Adaptive ϵ-Ranking, which reclassifies sets of non-dominated solutions using iteratively a randomized sampling procedure that applies ϵ-dominance with a mapping function $\mathbf{f}(\mathbf{x}) \mapsto^{\epsilon} \mathbf{f}'(\mathbf{x})$ to bias selection towards the distribution of solutions implicit in the mapping. We analyze the effectiveness of Adaptive ϵ-Ranking with three linear mapping functions for ϵ-dominance and study the importance of recombination to properly guide the algorithm towards the distribution we aim to find. As test problems, we use functions of the DTLZ family with $M = 6$ objectives, varying the number of variables N from 10 to 50.

1 Introduction

Recently, there has been growing interest on applying multi-objective evolutionary algorithms (MOEAs) to solve many-objective optimization problems [1], where the number of objective functions to optimize simultaneously is more than three. Historically, most applications of MOEAs have dealt with two and three objective problems leading to the development of several evolutionary approaches that work successfully in these low-dimensional objective spaces. However, it is well known

H. Aguirre (✉) · K. Tanaka
Faculty of Engineering, Shinshu University, 4-17-1 Wakasato, Nagano, 380-8553 Japan
e-mail: ahernan@shinshu-u.ac.jp; ktanaka@shinshu-u.ac.jp

A. Oyama
Japan Aerospace Exploration Agency, Institute of Space and Astronautical Science,
3-1-1 Yoshinodai, Chuo-ku, Sagamihara, Kanagawa 252-5210, Japan
e-mail: oyama@flab.isas.jaxa.jp

S. Cagnoni et al. (eds.), *Evolution, Complexity and Artificial Life*,
DOI 10.1007/978-3-642-37577-4_16,

that conventional MOEAs [2, 3] scale up poorly with the number of objectives of the problem. The poor performance of conventional MOEAs is attributed to an increased complexity inherent to high-dimensional spaces, the use of operators of selection and variation that fail to take into account the characteristics of many-objective landscapes [4–6], and the use of population sizes inappropriate to support the evolutionary search on high-dimensional spaces [7].

The goal of MOEAs is to find trade-off solutions with good properties of convergence to the Pareto front, well spread, and well distributed along the front. A good distribution of solutions is usually assumed to be uniform. However, other distributions are often desired, either because of preferences or because they are required to extract relevant knowledge about the problem in order to provide useful guidelines to designers during the implementation of preferred solutions.

We focus on distribution search in the context of many-objective optimization where, in addition to good convergence towards the optimal Pareto front, we are required to find a set of trade-off solutions spread according to the distribution we aim to achieve. Methods based on relaxed forms of Pareto dominance, such as ϵ-dominance [8], can be used to search for a desired distribution of solutions. In this work, we adopt Adaptive ϵ-Ranking [9], a procedure that reclassifies sets of non-dominated solutions using iteratively a randomized sampling procedure that applies ϵ-dominance with a mapping function $\mathbf{f}(\mathbf{x}) \mapsto^{\epsilon} \mathbf{f}'(\mathbf{x})$ to bias selection towards the distribution implicit in the mapping. We analyze the effectiveness of Adaptive ϵ-Ranking with three linear mapping functions for ϵ-dominance and study the importance of recombination to properly guide the algorithm towards the distribution we seek to find. These mappings try to produce three different distributions in which solutions are spaced by the same distance, spaced by a distance that increases linearly from the center towards the extremes of the objective space, and solutions spaced by a distance that decreases linearly from the center towards the extremes. As test problems, we use functions of the DTLZ family [10] with $M = 6$ objectives, varying the number of variables N from 10 to 50. We show that in many-objective problems, in addition to setting a proper mapping function, recombination plays a significant role to induce the distribution we aim to achieve.

2 Pareto Dominance and ϵ-Dominance

Let us consider, without loss of generality, a maximization multi-objective problem with M objectives:

$$\textit{maximize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})) \tag{1}$$

where $\mathbf{x} \in \mathscr{S}$ is a solution vector in the feasible solution space $\mathscr{S}$, and $f_1(\cdot), f_2(\cdot), \ldots, f_M(\cdot)$ the M objectives to be maximized.

Pareto dominance is widely used to rank solutions in multi-objective optimization. It can be defined as follows.

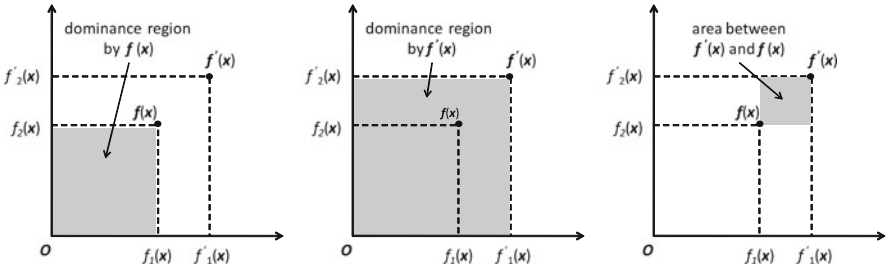

Fig. 1 A point $\mathbf{f}(\mathbf{x})$ in objective space, its corresponding mapped point $\mathbf{f}'(\mathbf{x})$, their dominance regions and the area between $\mathbf{f}'(\mathbf{x})$ and $\mathbf{f}(\mathbf{x})$

Pareto Dominance. A solution $\mathbf{x}$ is said to Pareto-dominate another solution $\mathbf{y}$, denoted by $\mathbf{f}(\mathbf{x}) \succ \mathbf{f}(\mathbf{y})$, if the two following conditions are satisfied:

$$\begin{aligned} &\forall i \in \{1, \ldots, M\} \quad f_i(\mathbf{x}) \geq f_i(\mathbf{y}) \ \wedge \\ &\exists i \in \{1, \ldots, M\} \quad f_i(\mathbf{x}) > f_i(\mathbf{y}). \end{aligned} \tag{2}$$

In addition, relaxed forms of Pareto dominance are used to maintain an archive of non-dominated solutions and also within the selection operator to rank solutions during evolution. One way to implement a relaxed form of dominance is ϵ-dominance [8]. In ϵ-dominance, the objective vector $\mathbf{f}(\mathbf{x})$ of a solution $\mathbf{x}$ is first mapped to another point $\mathbf{f}'(\mathbf{x})$ in the objective space and dominance is calculated using the mapped point. It can be defined as follows

ϵ-Dominance. A solution $\mathbf{x}$ is said to ϵ-dominate another solution $\mathbf{y}$, denoted by $\mathbf{f}(\mathbf{x}) \succ^{\epsilon} \mathbf{f}(\mathbf{y})$, if the following conditions are satisfied:

$$\begin{aligned} &\mathbf{f}(\mathbf{x}) \mapsto^{\epsilon} \mathbf{f}'(\mathbf{x}) \\ &\forall i \in \{1, \ldots, M\} \quad f_i'(\mathbf{x}) \geq f_i(\mathbf{y}) \ \wedge \\ &\exists i \in \{1, \ldots, M\} \quad f_i'(\mathbf{x}) > f_i(\mathbf{y}), \end{aligned} \tag{3}$$

where $\mathbf{f}(\mathbf{x}) \mapsto^{\epsilon} \mathbf{f}'(\mathbf{x})$ is a mapping function that depends on a parameter ϵ.

Figure 1 illustrates a point $\mathbf{f}(\mathbf{x})$ and its mapped point $\mathbf{f}'(\mathbf{x})$ in the objective space by a mapping function $\mathbf{f}(\mathbf{x}) \mapsto^{\epsilon} \mathbf{f}'(\mathbf{x})$, their dominance regions and the area between $\mathbf{f}(\mathbf{x})$ and $\mathbf{f}'(\mathbf{x})$. Note that a larger area between $\mathbf{f}(\mathbf{x})$ and $\mathbf{f}'(\mathbf{x})$ would correspond to a larger expansion of $\mathbf{f}'(\mathbf{x})$'s dominance region. Various kinds of mapping functions have been used for ϵ-dominance, such as multiplicative, additive, and logarithmic. In the next section we describe three additive mapping functions used in our study.

3 Mapping Functions

In this work we investigate distribution search using three additive mapping functions for ϵ-dominance in the context of many-objective optimization. The mappings are designed to induce distributions of solutions evenly spaced by ϵ,

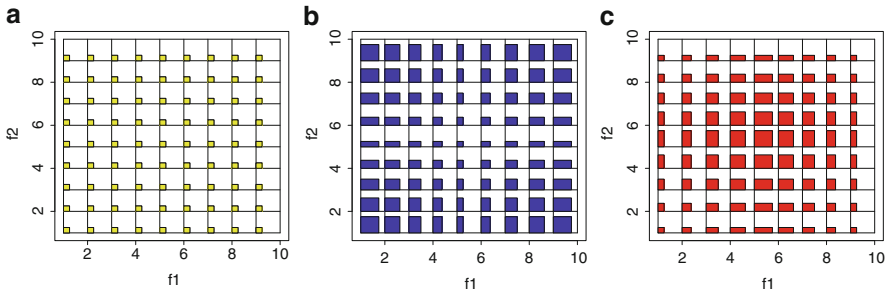

Fig. 2 Areas between $\mathbf{f}(\mathbf{x})$ and its mapped point $\mathbf{f}'(\mathbf{x})$ for several points by Additive, Expansion, and Contraction using the same ϵ in the three mapping functions. (**a**) Additive, (**b**) Linear Expansion, and (**c**) Linear Contraction

spaced by a distance that increases linearly from the center towards the extremes of objective space, and spaced by a distance that decreases linearly from the center towards the extremes. These mappings are called Additive, Linear Expansion from the Center, and Linear Contraction from the Center, respectively. Figure 2 illustrates the areas between $\mathbf{f}(\mathbf{x})$ and $\mathbf{f}'(\mathbf{x})$ for several points in objective space by these mappings. Their definitions are as follows.

Additive. Additive maps $\mathbf{f}(\mathbf{x})$ to $\mathbf{f}'(\mathbf{x})$ by adding the same value ϵ to all coordinates f_i, independently of the position of $\mathbf{f}(\mathbf{x})$ in objective space. The expression for Additive mapping is as follows

$$f_i'(\mathbf{x}) = f_i(\mathbf{x}) + \epsilon, \quad i = 1, \ldots, m \tag{4}$$

Linear Expansion from Center. Linear Expansion maps $\mathbf{f}(\mathbf{x})$ to $\mathbf{f}'(\mathbf{x})$ by adding the same value ϵ to all coordinates f_i and an additional δ_i value that increases linearly with the distance of f_i to a central point. The expression for Linear Expansion from Center is as follows

$$f_i'(\mathbf{x}) = f_i(\mathbf{x}) + \epsilon + \delta_i, \quad i = 1, \ldots, m \tag{5}$$

where δ_i is calculated by

$$\delta_i = \gamma \frac{|f_i(\mathbf{x}) - \bar{f_i}|}{\frac{1}{2}|f_i^{\max} - f_i^{\min}|}. \tag{6}$$

Here, $\gamma > 0.0$ is a parameter that controls the slope of the linear increase, $f_i^{\max}$ and $f_i^{\min}$ are the maximum and minimum values of objective f_i, calculated over all solutions in the population, and $\bar{f_i}$ is the central point calculated by

$$\bar{f_i} = \frac{1}{2}(f_i^{\max} + f_i^{\min}). \tag{7}$$

Linear Contraction from Center. Linear Contraction, similar to Linear Expansion, also adds the same ϵ to all coordinates f_i and a variable δ_i to map $\mathbf{f}(\mathbf{x})$ to $\mathbf{f}'(\mathbf{x})$ using Eq. (5). However, here δ_i is calculated so that its value decreases linearly from the central point. In case $f_i(\mathbf{x}) > \bar{f}_i$, δ_i is computed as

$$\delta_i = \gamma \frac{|f_i(\mathbf{x}) - f_i^{\max}|}{\frac{1}{2}|f_i^{\max} - f_i^{\min}|}. \tag{8}$$

Otherwise, δ_i is given by

$$\delta_i = \gamma \frac{|f_i(\mathbf{x}) - f_i^{\min}|}{\frac{1}{2}|f_i^{\max} - f_i^{\min}|}. \tag{9}$$

4 Adaptive ϵ-Ranking (AϵR)

Adaptive ϵ-Ranking (AϵR) [9] reclassifies sets of non-dominated solutions using iteratively a randomized sampling procedure that applies ϵ-dominance [8] with mapping function $\mathbf{f}(\mathbf{x}) \mapsto^{\epsilon} \mathbf{f}'(\mathbf{x})$ to favor the distribution of solutions implicit in the mapping. In our previous work, we have used AϵR with a multiplicative mapping function for ϵ-dominance. In this work we implement AϵR within the NSGA-II framework [11] and use three different additive mapping functions, as mentioned above.

AϵR in the framework of NSGA-II assigns a new primary ranking of solutions by reclassifying the fronts $\mathscr{F}_i$ $(i = 1, \ldots, N_F)$ found by non-domination sorting into fronts $\mathscr{F}^{\epsilon} = \{\mathscr{F}_j^{\epsilon}\}$ $(j = 1, 2, \ldots, N_F^{\epsilon})$, where $N_F^{\epsilon} \geq N_F$. The main steps are as follows:

Step 1 Assign the first front for reclassification, $\mathscr{A} = \mathscr{F}_1$. Set front counters $i = 1$ and $j = 1$.

Step 2 Call ϵ-sampling with mapping function $\mathbf{f}(\mathbf{x}) \mapsto^{\epsilon} \mathbf{f}'(\mathbf{x})$ and parameter $\epsilon > 0.0$ to get a sample $\mathscr{S}$ from $\mathscr{A}$ and their ϵ-dominated solutions $\mathscr{D}^{\epsilon}$.

Step 3 Consider the sampled solutions as the reclassified front, $\mathscr{F}_j^{\epsilon} = \mathscr{S}$.

Step 4 If there are still fronts to reclassify, $i + 1 < N_F$, join the ϵ-dominated solutions with solutions of the next front and assign them for reclassification, $\mathscr{A} = \mathscr{D}^{\epsilon} \cup \mathscr{F}_{i+1}$. Otherwise, $\mathscr{A} = \mathscr{D}^{\epsilon}$.

Step 5 Update counters $i = i + 1$, $j = j + 1$. If $\mathscr{A}$ is not empty, go to Step 2.

The sampling procedure referred to in Step 2 applies ϵ-dominance with mapping function $\mathbf{f}(\mathbf{x}) \mapsto^{\epsilon} \mathbf{f}'(\mathbf{x})$ and parameter $\epsilon > 0.0$ to virtually extend the dominance area of the sampled solution in order not to include closely located solutions in the sample. Thus, solutions in the sample are expected to be spread according to the mapping function. In other words, AϵR tries to search the distribution of solutions specified implicitly by the mapping function. The ϵ-sampling procedure is as follows:

Step 1 Select extreme solutions from $\mathscr{A}$ (without replacement) and assign them to the sample set $\mathscr{S}$.
Step 2 Select randomly one solution from $\mathscr{A}$ (without replacement) and assign it to $\mathscr{S}$.
Step 3 Remove from $\mathscr{A}$ solutions ϵ-dominated by the randomly selected solution and assign them to $\mathscr{D}^{\epsilon}$.
Step 4 If $\mathscr{A}$ is not empty, go to Step 2.

The number of rank-1 solutions $|\mathscr{F}_1^{\epsilon}|$ after reclassification depends on the value to which ϵ (≥ 0) is set. Larger values of ϵ imply that sampled solutions ϵ-dominate larger areas, increasing the likelihood of having more ϵ-dominated solutions excluded from the sample that form $\mathscr{F}_1^{\epsilon}$. AϵR adapts ϵ at each generation so that $|\mathscr{F}_1^{\epsilon}|$ is close to the size of the parent population $|\mathscr{P}|$ with the following rule. If $|\mathscr{F}_1^{\epsilon}| > |\mathscr{P}|$ it increases the step of adaptation $\Delta \leftarrow \min(\Delta \times 2, \Delta_{\max})$ and $\epsilon \leftarrow \epsilon + \Delta$. Otherwise, if $|\mathscr{F}_1^{\epsilon}| < |\mathscr{P}|$ it decreases $\Delta \leftarrow \max(\Delta \times 0.5, \Delta_{\min})$ and $\epsilon \leftarrow \max(\epsilon - \Delta, 0.0)$. In this work we set initial values $\epsilon_0 = 0.0$ and $\Delta_0 = 0.005$. Also, $\Delta_{\max} = 0.05$ and $\Delta_{\min} = 0.0001$.

5 Simulation Results and Discussion

5.1 Test Problems and Experimental Setup

We study the performance and behavior of NSGA-II and AϵR on the DTLZ test functions family [10]. These functions are scalable in the number of objectives and variables and thus allow for a many-objective study. In this work we present results for DTLZ2 with $M = 6$ objectives varying the total number of variables N from 10 to 50. DTLZ2 has a non-convex Pareto-optimal surface in the first quadrant of the M-dimensional unit hypersphere. For Linear Expansion and Contraction in AϵR we vary the parameter γ from 0.10 to 0.30, which determines the slope of the linear expansion or contraction, respectively. We run the algorithms 30 times and present average results, unless stated otherwise. We use a different random seed in each run, but all algorithms use the same seeds. The number of generations is set to 500 generations, and population size to 300 ($\mathscr{P} = \mathscr{Q} = 300$). The algorithms use the Simulated Binary Crossover (SBX) [12] and polynomial mutation, setting their distribution exponents to $\eta_c = 15$ and $\eta_m = 20$, respectively. Crossover rate is $pc = 1.0$, crossover rate per variable $pcv = \{0.5, 0.1\}$, and mutation rate per variable is $pm = 1/N$.

5.2 Effects on Convergence

In this section we analyze convergence by conventional NSGA-II and AϵR with different mappings. Figure 3 shows the average Generational Distance (GD) [13]

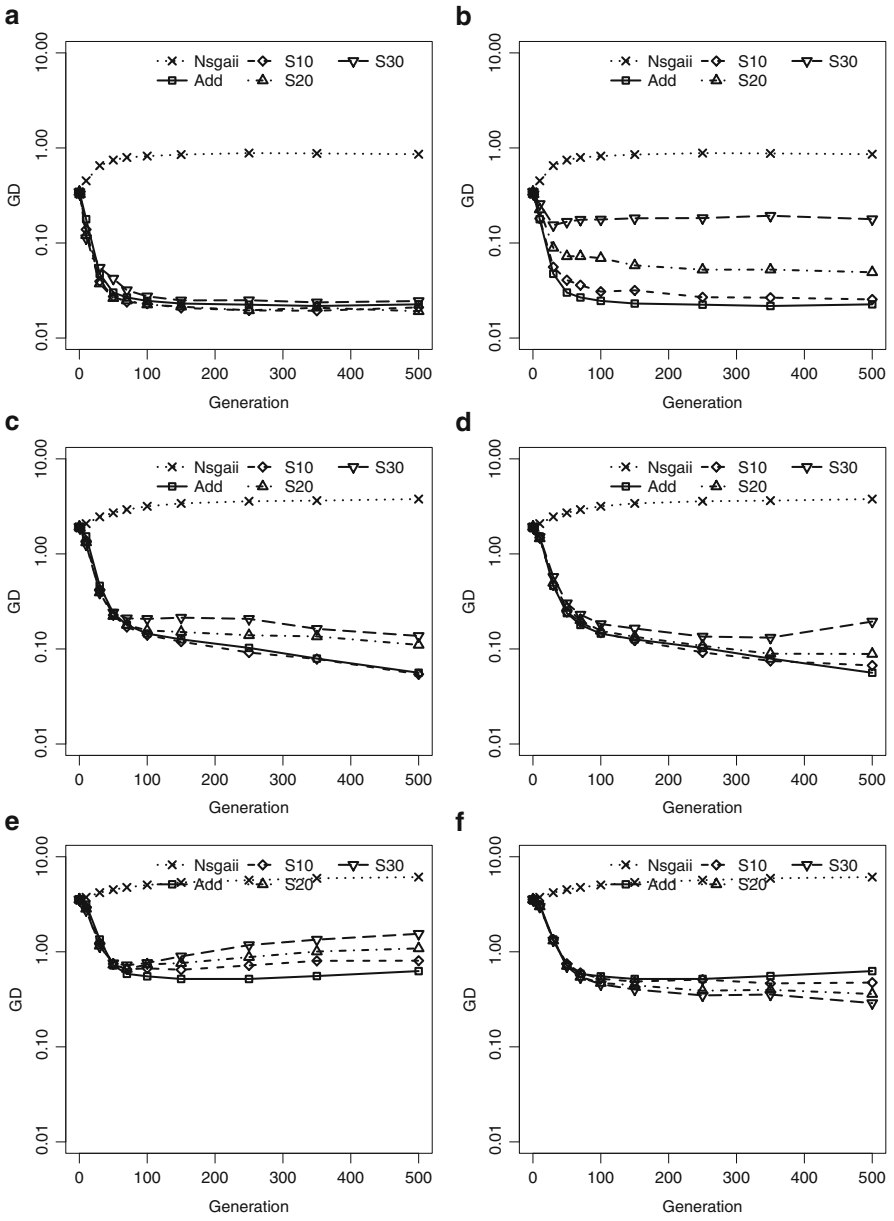

Fig. 3 Generational Distance by NSGA-II, Additive Epsilon, Linear Expansion, and Linear Contraction. DTLZ2, $M = 6$, $N = \{10, 30, 50\}$, $pcv = 0.5$. (**a**) Expansion $N = 10$, (**b**) Contraction $N = 10$, (**c**) Expansion $N = 30$, (**d**) Contraction $N = 30$, (**e**) Expansion $N = 50$, and (**f**) Contraction $N = 50$

over the generations setting the algorithms with $pcv = 0.5$, a widely used value for crossover rate per variable. Results are shown for the DTLZ2 function with $N = \{10, 30, 50\}$ variables. In each plot results by NSGA-II and AϵR with Additive mapping (labeled in the plots as Add) are included for reference, together with results by AϵR with Expansion or AϵR with Contraction using slope values of $\gamma = 0.10$, $\gamma = 0.20$ and $\gamma = 0.30$ (labeled in the plots as S10, S20, and S30, respectively). From this figure, note that GD by NSGA-II increases since the first generation, for any number of variables N, and remains higher than the GD of the random initial population. In other words, NSGA-II is diverging from the true Pareto front rather than converging towards it. On the contrary, GD by AϵR algorithms reduces with the number of generations and becomes significantly smaller compared to GD of the initial population. That is, sampling using ϵ-dominance helps the algorithm to converge towards the true Pareto front. These results are in accordance with previous reports on ϵ-dominance-based algorithms by several researches.

The bad performance of NSGA-II can be explained in terms of selection as follows. In many-objective problems most solutions are non-dominated since early generations. Thus, NSGA-II's selection relies mostly on the diversity estimation operator, which induces the population to spread rather than to converge. This misleads the algorithm towards dominance-resistant solutions in the extreme regions of the Pareto front, where the fitness of solutions takes values close to 0.0 in some objectives and very large values in other objectives.

Looking at the three variants of AϵR, it can be seen that overall GD by Additive mapping is better than by the two other variants. However, note that the best value of the achieved GD gets worse as we increase the number of variables. This is mainly because it is more challenging to optimize a larger number of variables and also because when we increase the number of variables evolution starts from an initial population that is farther away from the true Pareto front, as evidenced by the larger values of GD at generation $t = 0$. Comparing Expansion and Additive mappings, we can see that GD's transition over the generations looks similar for $N = 10$ variables, but GD by Expansion becomes worse than by Additive for $N = 30$ and $N = 50$ variables. Notice that GD by Expansion assumes worse values as the slope γ increases. In addition, note that for $N = 50$, after an initial stage in which GD improves (smaller values) we can see that GD starts to get worse (larger values). Comparing Contraction and Additive mappings, we can see that GD by Contraction with larger slope γ is worse than GD by Additive for small N. However, for $N = 50$ we can see that GD by Contraction becomes better than by Additive and also that larger values of the slope γ lead to better values of GD.

5.3 Convergence Reducing Recombination Rate per Variable

Recent work on combinatorial problems has shown that diversity in variable space of non-dominated solutions gets significantly larger with the number of objectives and recombination may become too disruptive, affecting its effectiveness to find

better solutions [6]. In this section we analyze the effects on convergence of a less explorative recombination by reducing the crossover rate per variable from $pcv = 0.5$ to $pcv = 0.1$. Our intention is to verify whether our findings on combinatorial problems hold on continuous problems and analyze the relationship between the effectiveness of the recombination operator and the distribution of solutions that selection tries to induce. Figure 4 shows the average GD by all algorithms with $pcv = 0.1$. In this figure, note the convergence trend by NSGA-II for $N = 30$ and $N = 50$ variables, which is not observed for $pcv = 0.5$. This is an indication that recombination with $pcv = 0.1$ is more effective than with $pcv = 0.5$. This also implies that ineffective recombination is also responsible for the divergence of the algorithm towards dominance-resistant solutions, and not only selection as suggested in the previous section. Looking at the AϵR algorithms, we can see that they can achieve better values of GD with $pcv = 0.1$ than with $pcv = 0.5$. Note also that the improvement of GD becomes more significant on problems with larger number of variables. The trends of GD by Expansion and Contraction mappings using $pcv = 0.1$ are similar to those observed for $pcv = 0.5$ for problems with up to 30 variables. However, in the case of $N = 50$ variables, note that Expansion achieves better GD than Additive and Contraction achieves worse GD than Additive. It is interesting to note that exactly the opposite trend is observed for $pcv = 0.5$ as shown in Fig. 3e,f. The overall better GD obtained by reducing recombination rate per variable from $pcv = 0.5$ to $pcv = 0.1$ suggests that the effectiveness of recombination is an issue that needs to be carefully considered in continuous problems.

5.4 Obtained Solutions and Their Distribution

In previous sections we analyzed the effects of AϵR algorithms and rate of recombination per variable on the convergence of the algorithms. In this section we focus our analysis on the distributions of the obtained solutions.

Figure 5 shows scatter plots of the Pareto optimal solutions found by Additive mapping for one run of the algorithm. Results are shown in the range $[0.0, 2.0]$ for the planes formed by f_1, f_3, and f_6 at generations 50 and 500 using $pcv = 0.5$ and $pcv = 0.1$. On the other hand, Figs. 6 and 7 show hexagon binning plots of solutions in the $f_1 - f_3$ plane at generation 500. A hexagon binning is a form of bivariate histogram. A grid of hexagons is formed in the plane $f_1 - f_3$ and the number of solutions falling in each hexagon are counted. The hexagons with count > 0 are plotted varying the radius of the hexagon in proportion to the counts. Results are shown in the range $[0.0, 1.25]$ for the three variants Additive, Contraction, and Expansion mappings using $pcv = 0.5$ in Fig. 6 and $pcv = 0.1$ in Fig. 7. In Fig. 5, note the different distributions of solutions obtained by using $pcv = 0.5$ and $pcv = 0.1$. At generation 50, we can observe that many solutions have become dominance-resistant, i.e. f_1 and (or) f_3 approach 0.0, when $pcv = 0.5$. On the other hand, many more solutions are observed in the central regions of objective

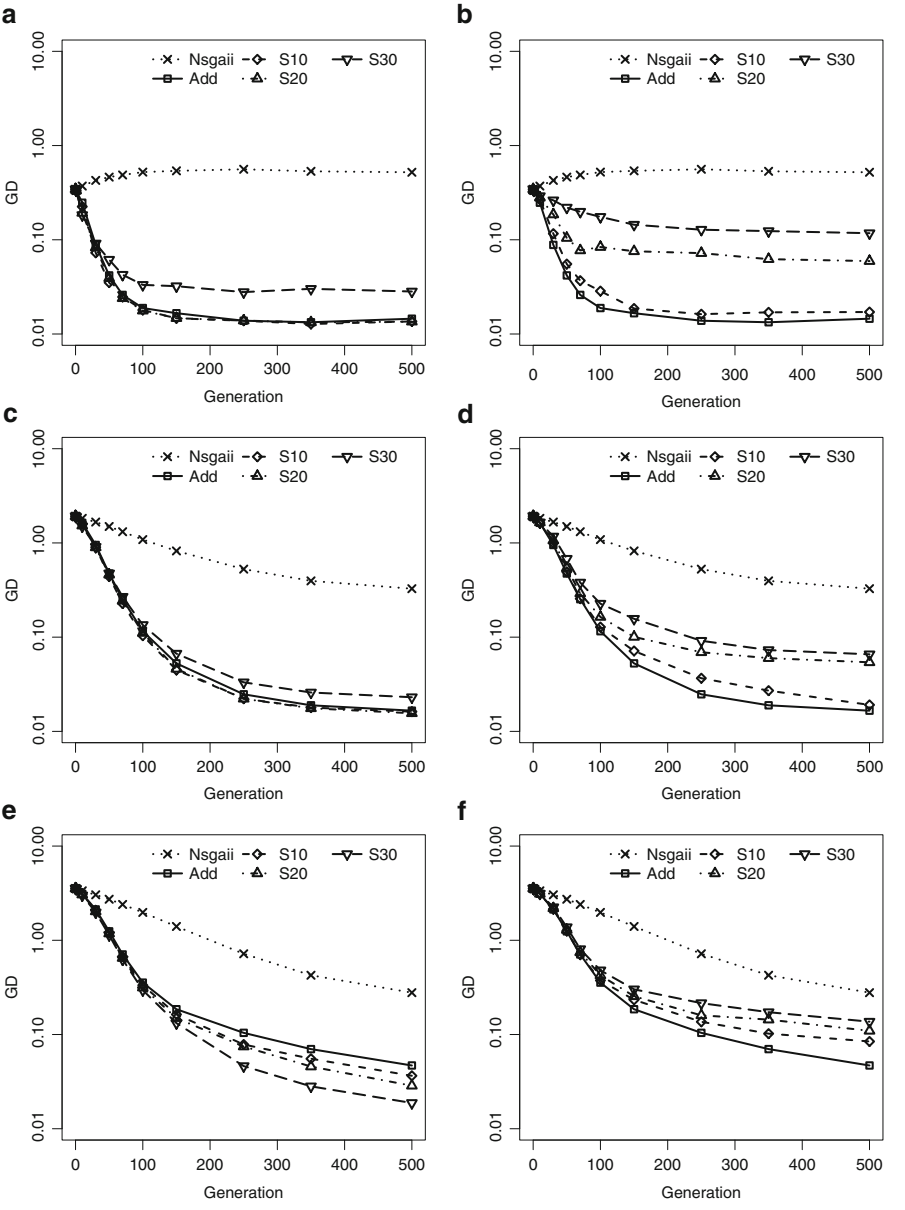

Fig. 4 Generational Distance by NSGA-II, Additive Epsilon, Linear Expansion, and Linear Contraction. $M = 6$, $N = \{10, 30, 50\}$, $pcv = 0.1$. (**a**) Expansion $N = 10$, (**b**) Contraction $N = 10$, (**c**) Expansion $N = 30$, (**d**) Contraction $N = 30$, (**e**) Expansion $N = 50$, and (**f**) Contraction $N = 50$

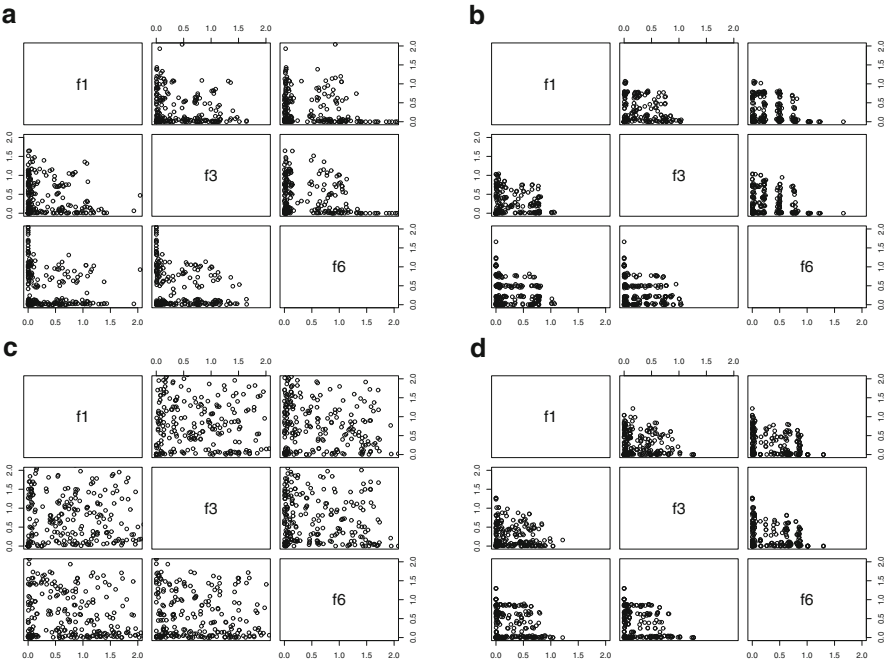

Fig. 5 Non-dominated solutions by Additive mapping, range [0.0, 2.0]. DTLZ2, $M = 6$, $N = 50$. (**a**) Generation 50, pcv=0.5, (**b**) generation 500, pcv=0.5, (**c**) generation 50, pcv=0.1, and (**d**) generation 500, pcv=0.1

space when $pcv = 0.1$. At generation 500, most solutions have converged close to the optimum hypersphere of radius 1. However, solutions obtained by $pcv = 0.1$ are more evenly distributed than solutions obtained by $pcv = 0.5$. This is better illustrated by hexagon binning in Figs. 6a and 7a. For example, note that in the case of $pcv = 0.5$ there are 47 solutions, out of a maximum of 300, located close to $f_1 = f_3 = 0.0$, whereas there are 34 in the case of $pcv = 0.1$. The distribution of solutions by Additive mapping with $pcv = 0.1$ reflects more precisely the kind of distribution we want to achieve with the additive mapping described in Sect. 3. Our analysis of GD shows that an appropriate recombination rate is important to improve convergence. The analysis of the distribution of solutions suggests that selection alone, without considering a proper recombination, cannot induce the distribution we aim for.

Looking at the hexagon binning plots by Expansion and Contraction mappings, the relevance of recombination to achieve the desired distribution of solutions becomes clearer. Note in Fig. 7b that in the case of $pcv = 0.1$ solutions tend to cluster around $f_1 = 0.5$, $f_3 = 0.5$, and $f_1 = f_3 = 0.5$, which is what should be expected for the Expansion mapping. However, that is not the case when $pcv = 0.5$, where a large number of solutions are clustered at $f_1 = f_3 = 0.0$, as shown in Fig. 6b. Similarly, note in Fig. 7c that in the case of $pcv = 0.1$

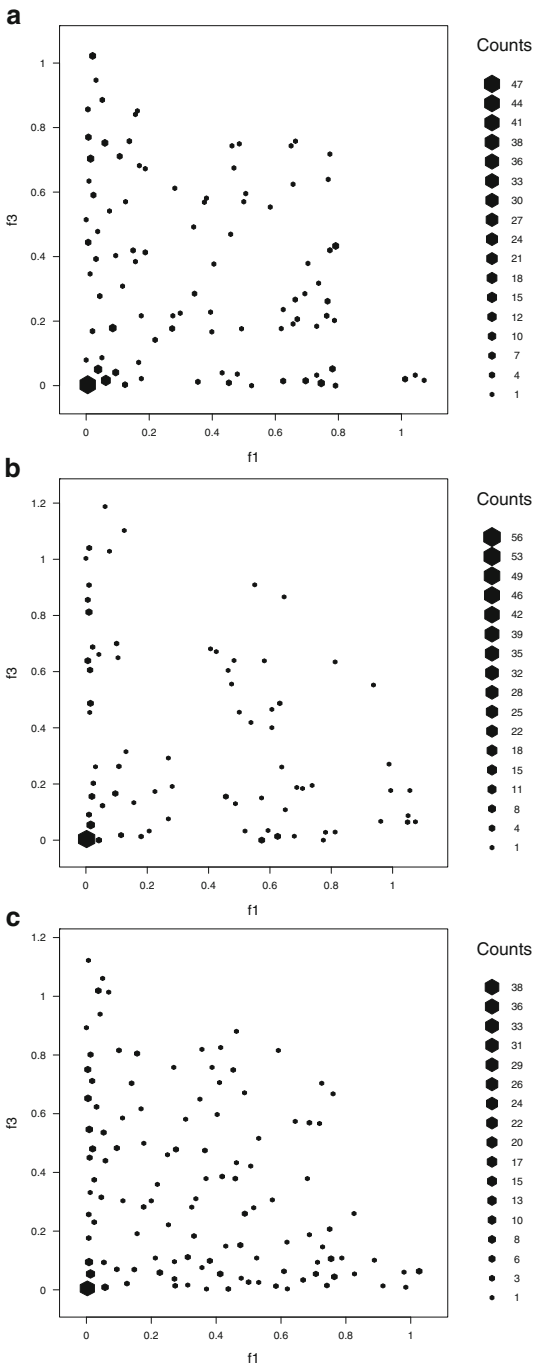

Fig. 6 Hexagon binning of non-dominated solutions in plane $f_1 - f_3$ by Additive, Expansion, and Contraction. DTLZ2, $M = 6$, $N = 50$, $pcv = 0.5$. (**a**) Additive $pcv = 0.5$, (**b**) Expansion $pcv = 0.5$, and (**c**) Contraction $pcv = 0.5$

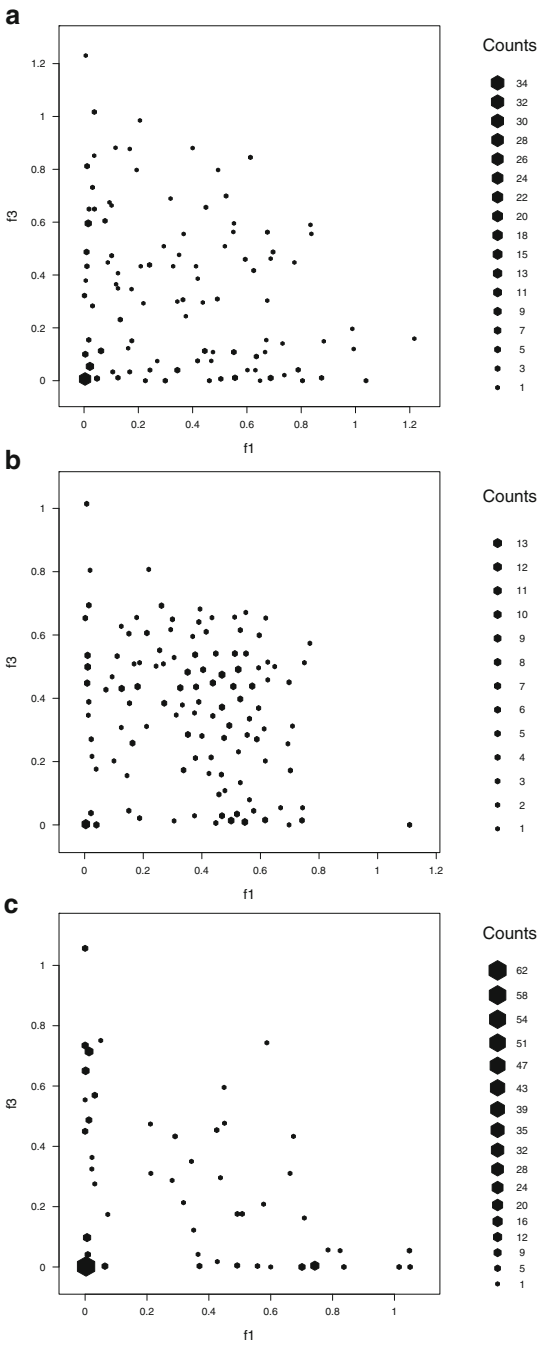

Fig. 7 Hexagon binning of non-dominated solutions in plane $f_1 - f_3$ by Additive, Expansion, and Contraction. DTLZ2, $M = 6$, $N = 50$, $pcv = 0.1$. (**a**) Additive $pcv = 0.1$, (**b**) Expansion $pcv = 0.1$, and (**c**) Contraction $pcv = 0.1$

there is a large number of solutions concentrated around $f_1 = f_3 = 0.0$ and along the axis, as expected for the Contraction mapping. On the other hand, in the case of $pcv = 0.5$ the distribution of solutions in the objective space looks more uniform, which is counterintuitive for the Contraction mapping, as shown in Fig. 6c. In fact, it is interesting to note that the distribution by Contraction mapping with $pcv = 0.5$ shown in Fig. 6c looks more uniform that the one by Additive mapping with $pcv = 0.5$ shown in Fig. 6a. Looking at Fig. 3f we can also see that Contraction mapping with $pcv = 0.5$ achieves better GD than Additive mapping with $pcv = 0.5$. If GD were our only criterion to evaluate the algorithms, it would seem that Contraction mapping works better than Additive mapping. However, the distribution generated by Contraction is not what should be expected from such a mapping. In the future we should analyze in more detail the reasons why Contraction mapping generates a more uniform distribution than Additive mapping when a large crossover rate is used.

6 Analysis on Combined Pareto Optimal Set

The mappings used for ϵ-dominance aim at generating different distributions of solutions. However, a particular distribution is meaningful if the solutions found have good convergence properties. In the following, we combine solutions found by all algorithms to analyze distribution and convergence of solutions that remain non-dominated relative to solutions found by the other algorithms. To achieve this, we extract the set PPOS of Pareto optimal solutions from the non-dominated solutions obtained independently by the four algorithms NSGA-II, Adaptive, Linear Expansion, and Linear Contraction. Each solution in PPOS is tagged with the algorithm that found it.

Figure 8a,c,e show the average fraction of solutions found by the algorithms that remain non-dominated in PPOS for $N = \{10, 30, 50\}$, $pcv = 0.1$, and $\gamma = 0.30$. Similarly, Fig. 8b,d,f show the average GD over the generations of the entire PPOS and the subsets of solutions in PPOS classified by their tag. First, note that at $t = 0$ the fraction of solutions in PPOS by all algorithms is 1.0. This is because all algorithms start with the same random initial population and thus at $t = 0$ render the same set of non-dominated solutions. Looking at results obtained for $t > 0$, it can be seen that the fraction of non-dominated solutions in PPOS remains close to 1.0 by Additive, Linear Expansion, and Linear Contraction regardless of the number of variables of the problem. That is, only few solutions found by these algorithms turn out to be dominated when PPOS is formed. On the other hand, note that the fraction of solutions in PPOS by NSGA-II falls rapidly to a value below 0.2 and remains low throughout the generations, especially for $N = 10$. Increasing the number of variables to $N = 30$ and $N = 50$, it can be seen that the fraction increases and reaches values between 0.3 and 0.4 at generation $t = 500$. A similar trend is observed varying γ, as shown in Fig. 9. The large fraction of solutions in PPOS by Additive, Linear Expansion, and Linear Contraction indicates that solutions found by these mappings are located in different areas of objective space as expected.

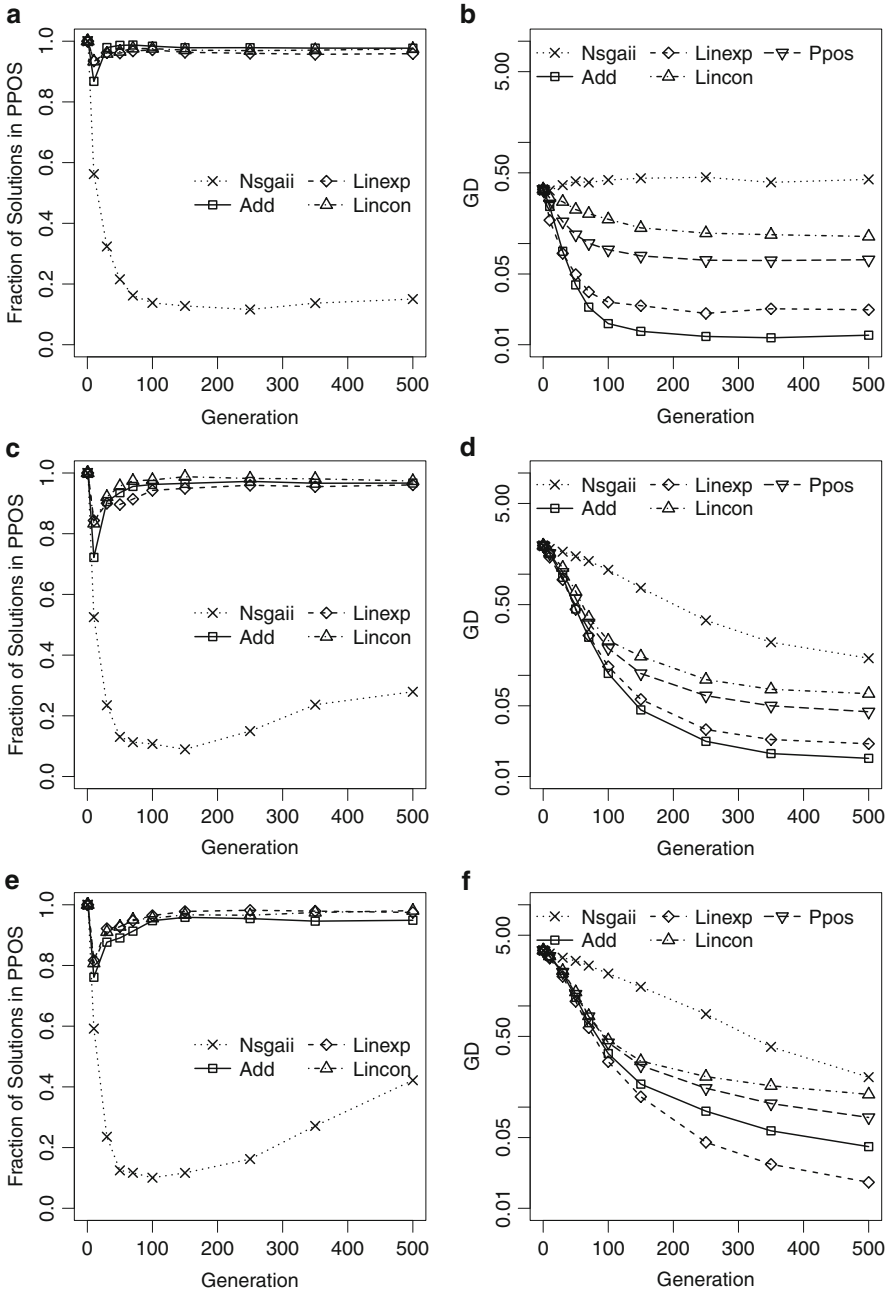

Fig. 8 Generational Distance and Fraction of Solutions in PPOS by NSGA-II, Additive, Linear Expansion, and Linear Contraction varying $N = \{10, 30, 50\}$. DTLZ2, $M = 6$, $pcv = 0.1$, $\gamma = 0.30$. (**a**) Fraction, $N = 10$, (**b**) GD, $N = 10$, (**c**) fraction, $N = 30$, (**d**) GD, $N = 30$, (**e**) fraction, $N = 50$, and (**f**) GD, $N = 50$

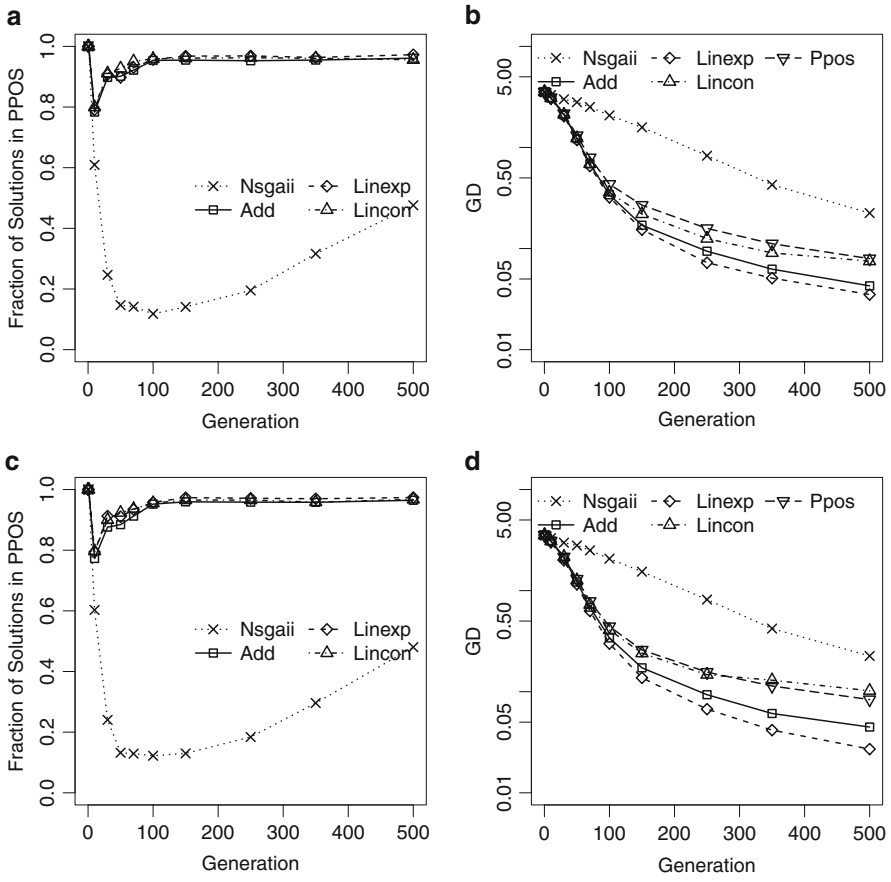

Fig. 9 Generational Distance and Fraction of Solutions in PPOS by NSGA-II, Additive, Linear Expansion, and Linear Contraction varying $\gamma = \{0.20, 0.30\}$. DTLZ2, $M = 6$, $N = \{50\}$, $pcv = 0.1$. (**a**) Fraction, $N = 50$, $\gamma = 0.10$, (**b**) GD, $N = 50$, $\gamma = 0.10$, (**c**) fraction, $N = 50$, $\gamma = 0.10$, and (**d**) GD, $N = 50$, $\gamma = 0.10$

Looking at the GD of solutions in PPOS, it can be seen that the subset of solutions corresponding to Additive renders the smallest GD for $N = 10$ and $N = 30$. However, the subset of solutions corresponding to Linear Expansion renders the smallest GD for $N = 50$. The subset of solutions in PPOS by Linear Contraction is the third best in terms of GD and the subset of solution in PPOS by NSGA-II is the worst. It should be noticed that all solutions in PPOS are non-dominated. However, some solutions with high GD are obtained by Linear Contraction and especially by NSGA-II. This suggests that some of these solutions, although non-dominated, are dominance resistant with very small values in some objective functions but with not so good values on the other objective functions.

Figure 10 shows fitness vectors in the plane $f_1 - f_3$ of the solutions in PPOS found by the algorithms in one of their runs. Non-dominated solutions

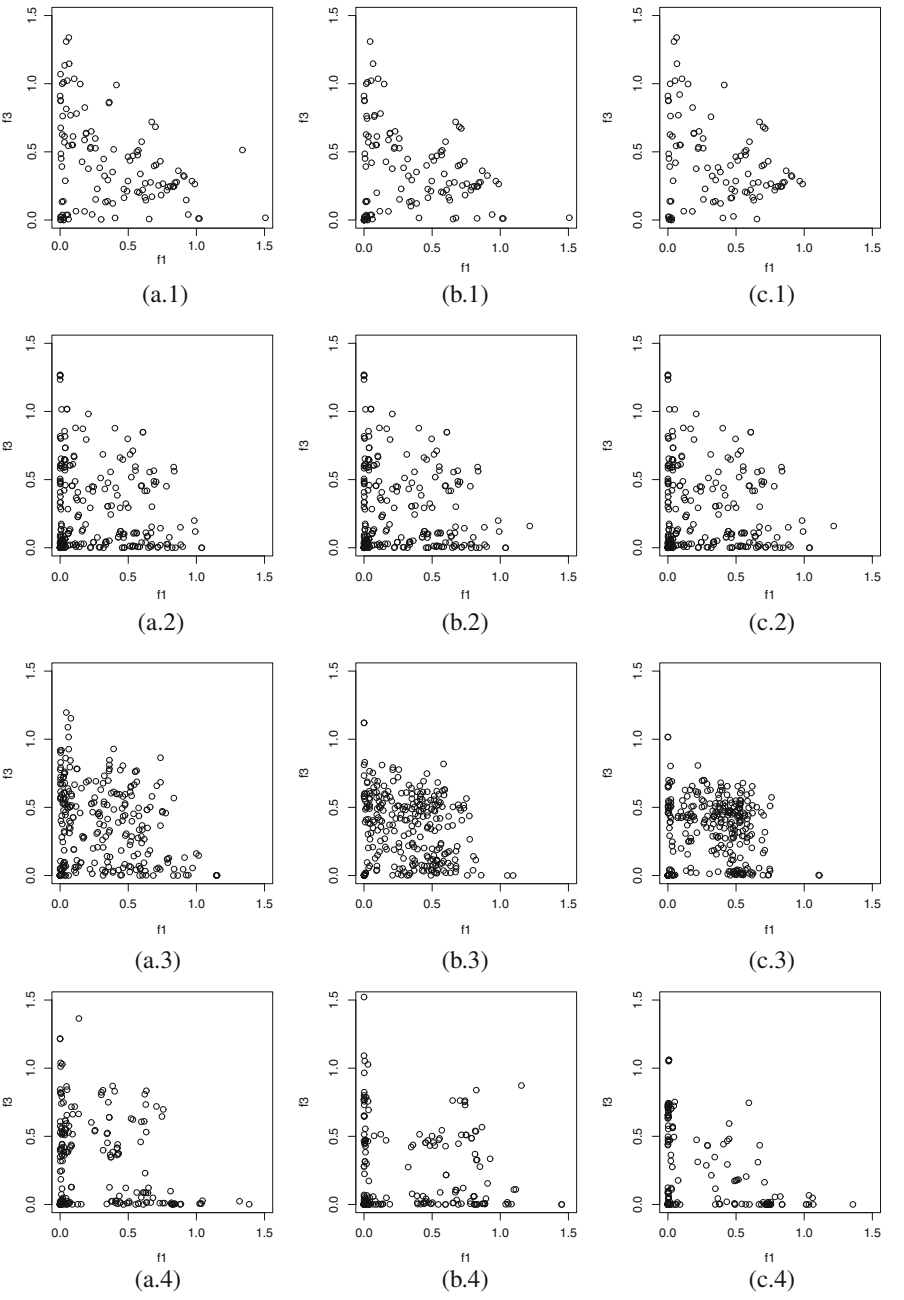

Fig. 10 Scatter Plots of Solutions in PPOS by NSGA-II, Additive Epsilon, Linear Expansion, and Linear Contraction. DTLZ2, $M = 6$, $N = \{50\}$, $pcv = 0.1$, $\gamma = \{0.10, 0.20, 0.30\}$. (**a.1**, **b.1**, **c.1**) NSGA-II, (**a.2**, **b.2**, **c.2**) Additive, (**a.3**) Exp. $\gamma = 0.10$, (**b.3**) Exp. $\gamma = 0.20$, (**c.3**) Exp. $\gamma = 0.30$, and (**a.4**) Cont. $\gamma = 0.10$, (**b.4**) Cont. $\gamma = 0.20$, (**c.4**) Cont. $\gamma = 0.30$

found by Linear Extension and Linear Contraction set to $\gamma = \{0.10, 0.20, 0.30\}$ are combined with results by NSGA-II and Additive to form one PPOS per γ. Figure 10a.1–a.4, b.1–b.4, and c.1–c.4 show solutions in PPOS for $\gamma = 0.10$, $\gamma = 0.20$, and $\gamma = 0.30$, respectively. From these figures note that the distributions by Additive look the most uniforms, as expected. Also, the distributions by Linear Expansion gradually cluster around $f_1 = 0.5$, $f_3 = 0.5$, $f_1 = f_3 = 0.5$ by increasing γ, whereas the distributions by Linear Contraction gradually cluster around the axis. Looking at results by NSGA-II, it is worth noting that the empty spaces observed in its distributions match with the regions where solutions by Additive, Linear Expansion, or Linear Contraction cluster.

7 Conclusions

In this work we have studied distribution search in the context of many-objective optimization focusing on the effectiveness of Adaptive ϵ-Ranking. We have analyzed three additive mapping functions for the ϵ-sampling procedure of Adaptive ϵ-Ranking in order to bias the search towards different distributions of solutions. We also analyzed the relationship between the effectiveness of the recombination operator and the distribution of solutions that selection tries to induce. We have verified that in many-objective continuous problems a less explorative recombination can increase substantially the convergence of solutions. Also, we verified that selection alone without considering a proper recombination rate cannot induce the distribution we seek to achieve. In the future, we would like to extend our analysis to larger populations and to other problems. Also, we would like to look deeper into the relationship between recombination and selection for distribution search.

References

1. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: a short review. In: Proceedings 2008 IEEE Congress on Evolutionary Computation, IEEE Press, pp. 2424–2431 (2008)
2. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley, Chichester (2001)
3. Coello, C., Van Veldhuizen, D., Lamont, G.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer, Boston (2002)
4. Aguirre, H., Tanaka, K.: Insights on properties of multi-objective MNK-landscapes. In: Proceedings 2004 IEEE Congress on Evolutionary Computation, IEEE Service Center, pp. 196–203 (2004)
5. Aguirre, H., Tanaka, K.: Working principles, behavior, and performance of MOEAs on MNK-landscapes. Eur. J. Oper. Res. **181**(3), 1670–1690 (2007)
6. Sato, H., Aguirre, H., Tanaka, K.: Genetic diversity and effective crossover in evolutionary many-objective optimization. In: Proceedings of Learning and Intelligent Optimization Conference (LION 5). Lecture Notes in Computer Science, vol. 6683, pp. 91–105. Springer, Berlin (2011)

7. Kowatari, N., Oyama, A., Aguirre, H., Tanaka, K.: A study on large population MOEA using adaptive epsilon-box dominance and neighborhood recombination for many-objective optimization. In: Proceedings of Learning and Intelligent Optimization Conference (LION 6). Lecture Notes in Computer Science, pp. 86–100. Springer, Berlin (2012)
8. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining convergence and diversity in evolutionary multi-objective optimization. Evol. Computation **10**(3), 263–282 (2002)
9. Aguirre, H., Tanaka, K.: Adaptive ε-ranking on many-objective problems. Evol. Intel. **2**(4), 183–206 (2009)
10. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. In: Proceedings 2002 Congress on Evolutionary Computation, IEEE Service Center, pp. 825–830 (2002)
11. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001 (2000)
12. Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. Complex Syst. **9**(2), 115–148 (1995)
13. Zitzler, E.: Evolutionary algorithms for multi-objective optimization: methods and applications. Ph.D. thesis, Swiss Federal Institute of Technology, Zurich (1999)

Concurrent Implementation Techniques Using Differential Evolution for Multi-Core CPUs: A Comparative Study Using Statistical Tests

Kiyoharu Tagawa

Abstract In order to utilize multi-core CPUs effectively, a concurrent version of a recently developed evolutionary algorithm, i.e., Differential Evolution (DE), is described. The concurrent version of DE is called Concurrent DE (CDE). CDE is designed based on a programming model known as "MapReduce" and implemented in Java. Two implementations of CDE, namely CDE/D and CDE/S, are proposed and compared from the viewpoint of both quality of solutions and execution time. Through the numerical experiments and the statistical tests conducted on two kinds of popular multi-core CPUs, it is shown that CDE/S uses multi-core CPUs more effectively than CDE/D. However, the quality of solutions obtained by CDE/S tends to fluctuate with the number of threads and the kind of benchmark problems.

1 Introduction

Differential Evolution (DE) proposed originally by Storn and Price [1] is arguably one of the most powerful stochastic real-parameter optimization algorithms in current use. DE is regarded as a kind of Evolutionary Algorithm (EA). However, comparing it with conventional EAs such as Genetic Algorithm (GA), Evolution Strategy (ES), and Particle Swarm Optimization (PSO), it has been reported that DE exhibits an overall excellent performance for a wide range of benchmark problems [2]. Furthermore, because of its simple but powerful searching capability, DE has been used in several scientific and engineering applications [2, 3].

Because EAs maintain a lot of individuals, or tentative solutions, manipulated competitively in the population, primary EAs have a parallel and distributed nature

K. Tagawa
School of Science and Engineering, Kinki University, Higashi-Osaka 577-8502, Japan
e-mail: tagawa@info.kindai.ac.jp

S. Cagnoni et al. (eds.), *Evolution, Complexity and Artificial Life*,
DOI 10.1007/978-3-642-37577-4_17, © Springer-Verlag Berlin Heidelberg 2014

intrinsically. Therefore, many parallelization techniques have been contrived for various EAs [4, 5]. These parallelization techniques of EAs can be extended easily to DE. Actually, parallel implementations of DE variants have been proposed for some platforms such as networked computers and computer clusters [6–9].

Recently, multi-core CPUs, which have more than one processor (core), have been introduced widely into commodity computers. Therefore, in order to utilize the additional cores to execute costly application programs, researchers' attention has been focused on their parallel and concurrent implementations [10]. Some tools are available today to parallelize application programs. Parallel compilers transform sequential programs into parallel ones automatically. However, due to complexity of automatically transforming sequential algorithms into parallel ones, the amount of parallelism reached using parallel compilers is low. In general, parallel programming obtains a higher performance than automatic parallelization. OpenMP (Open Multiprocessing; http://openmp.org/) is an Application Programming Interface (API) that supports parallel programming in C/C++ and Fortran on shared memory multi-platforms such as multi-core CPUs. Besides, the Java language provides high-level components and mechanisms to build concurrent application programs efficiently [11].

In addition to multi-core CPUs, Graphics Processing Units (GPUs) designed originally to accelerate graphics applications with several hundreds of simplified cores have also been used to run parallelized application programs. OpenCL (Open Computing Language; http://www.khronous.org/opencl/) is a low-level API for writing programs that execute across heterogeneous platforms consisting of multi-core CPUs and GPUs. A survey of parallel programming with OpenMP and OpenCL is provided in [12].

Multi-core CPUs and GPUs also offer a new paradigm of implementation to EAs. There have been several attempts to accelerate DE programs consisting of hundreds of threads executable in parallel by using GPUs [13, 14]. Certainly, GPUs are far faster than multi-core CPUs. However, GPU-based applications are limited by the architecture and memory model of GPU. GPU-based applications also require to build specific programs for the GPU, called "kernels," besides the main programs for the host CPU. Data transfers between CPU and GPU can easily become a bottleneck. On the other hand, CPU-based applications are not limited by the architecture of CPU such as the number of cores and much easier to implement than GPU-based ones [12]. Even though the number of available cores is not so large, the concurrent program executed on a multi-core CPU is the simplest and easiest way to implement a parallelized DE. Therefore, authors have proposed a concurrent version of DE called Concurrent DE (CDE) for multi-core CPUs [15]. Strictly speaking, the proposed CDE is a parallelized version of a novel DE based on the "steady-state model."

The procedure by which EAs update the individuals in the population is called "generation alternation model." Many EAs usually employ either of two types of generation alternation models [16]. The first one is called "generational model,"

while the second one is called "steady-state model." The original DE is based on the generational model [1]. According to the generational model, DE holds two populations, namely a primary one and a secondary one. After generating all individuals of the secondary population from those of the primary population, the primary one is replaced altogether by the secondary one. On the other hand, a new DE based on the steady-state model has been reported and studied lately [17–19]. The new DE is sometimes called Sequential DE (SDE) [17]. According to the steady-state model, SDE holds only one population. Then individuals in the population are updated one by one. Comparing it with the generational model, the steady-state model is usually suitable for parallelizing EAs [20]. That is because EAs based on the steady-state model need not synchronize the manipulations of all individuals for replacing the primary population by the secondary population at one time.

In order to realize the above CDE, two implementation techniques of CDE, namely CDE/D and CDE/S, are presented. CDE/D allocates the manipulations of individuals to the corresponding threads dynamically, while CDE/S allocates them to all threads statically. Both CDE/S and CDE/D are developed in Java. The multi-threading and the mutual exclusion provided by the Java language are indispensable for the two implementations of CDE. The performances of CDE/D and CDE/S are compared with respect to both the execution time and the quality of solutions using two kinds of popular multi-core CPUs: Intel(R) Core(TM) i7 and AMD Phenom(TM) II X6. Through the numerical experiments and the statistical tests conducted on six benchmark problems, it is shown that CDE/S uses multi-core CPUs more effectively than CDE/D. However, the quality of solutions obtained by CDE/S tends to fluctuate with the number of threads and the benchmark problems.

The remainder of this chapter is organized as follows. Section 2 formulates the real-parameter optimization and describes the algorithms of DE and SDE. Section 3 presents CDE with its two implementation techniques, namely CDE/D and CDE/S. Section 4 compares CDE/D and CDE/S through numerical experiments and statistical tests conducted on benchmark problems. Finally, Sect. 5 summarizes the chapter.

2 Differential Evolution

2.1 Representation

The real-parameter optimization problem is usually formulated as shown in (1). The global optimal solution of the optimization problem is a D-dimensional real-parameter vector $\mathbf{x} = (x_1, \cdots, x_j, \cdots x_D) \in \Re^D$ that minimizes the objective function value $f(\mathbf{x}) \in \Re$. Furthermore, the value of each decision variable $x_j \in \Re$ is limited to the range between the lower $\underline{x}_j$ and the upper $\overline{x}_j$ bounds as

$$\begin{cases} \text{minimize} & f(\mathbf{x}) = f(x_1, \cdots, x_j, \cdots, x_D), \\ \text{subject to} & \underline{x}_j \leq x_j \leq \overline{x}_j, \ j = 1, \cdots, D. \end{cases} \tag{1}$$

DE searches for a global optimal solution of the optimization problem in (1). Like other EAs, DE holds N_P tentative solutions of the optimization problem called individuals in the population $\mathbf{P}$. The i-th individual $\mathbf{x}_i \in \mathbf{P}$ is represented as

$$\mathbf{x}_i = (x_{1,i}, \cdots, x_{j,i}, \cdots, x_{D,i}), \tag{2}$$

where $x_{j,i} \in \Re$ and $\underline{x}_j \leq x_{j,i} \leq \overline{x}_j$, $j = 1, \cdots, D$, $i = 1, \cdots, N_P$.

2.2 DE Strategy

In order to generate a candidate for a new individual of the population $\mathbf{P}$, DE uses a unique reproduction strategy. The strategy of DE is defined by a series of three genetic operators, namely reproduction selection, differential mutation, and crossover. Even though various strategies have been proposed for DE [2, 17], a basic strategy named "DE/rand/1/exp" is described and used in this chapter. That is because our previous studies about SDE have shown that the basic strategy of DE is excellent and has relatively good compatibility with SDE too [19].

In the reproduction selection, each individual $\mathbf{x}_i \in \mathbf{P}$ is assigned to the "target vector" in turn. Besides the target vector $\mathbf{x}_i \in \mathbf{P}$, three other distinct individuals, say $\mathbf{x}_{i1}$, $\mathbf{x}_{i2}$ and $\mathbf{x}_{i3} \in \mathbf{P}$ ($i1 \neq i2 \neq i3$), are selected randomly from $\mathbf{P}$.

By using the above three individuals, the differential mutation generates a new real-parameter vector called "mutated vector" $\mathbf{v} = (v_1, \cdots, v_D) \in \Re^D$ as

$$\mathbf{v} = \mathbf{x}_{i1} + F\,(\mathbf{x}_{i2} - \mathbf{x}_{i3}), \tag{3}$$

where the scale factor $F \in \Re$ ($0 < F \leq 1$) is a control parameter.

The exponential crossover between the mutated vector $\mathbf{v}$ and the target vector $\mathbf{x}_i$ generates a candidate for a new individual $\mathbf{u} = (u_1, \cdots, u_j, \cdots, u_D)$ called "trial vector." Each component u_j of the trial vector $\mathbf{u}$ is inherited from either the mutated vector $\mathbf{v}$ or the target vector $\mathbf{x}_i$. The pseudo-code in (4) describes the procedure of the exponential crossover combined with the differential mutation shown in (3). The subscript $j_r \in [1,\ D]$ in (4) is selected randomly, which ensures that $\mathbf{u}$ differs from $\mathbf{x}_i \in \mathbf{P}$ for at least one component u_{j_r}. Furthermore, rand[0, 1] in (4) denotes a random number generator that returns a uniformly distributed random number in the range [0, 1]. As well as the scale factor F in (3), the crossover rate $C_R \in [0,\ 1]$ is a control parameter specified by the user in advance.

$$\left\lceil \begin{array}{l} j = j_r; \\ \text{do } \{ \\ \quad u_j = x_{j,i1} + F\,(x_{j,i2} - x_{j,i3}); \\ \quad j = j \,\%\, D + 1; \\ \} \text{ while}(\text{rand}[0,\ 1] < C_R \ \wedge\ j \neq j_r) \\ \text{while}(j \neq j_r)\ \{ \\ \quad u_j = x_{j,i}; \\ \quad j = j \,\%\, D + 1; \\ \} \end{array} \right. \tag{4}$$

If a component u_j of the trial vector $\mathbf{u}$ is out of the range $[\underline{x}_j,\ \overline{x}_j]$ as the result of the operation shown in (4), it is rescaled as

$$u_j = \begin{cases} x_{j,i1} + \text{rand}[0,\ 1]\,(\underline{x}_j - x_{j,i1}) & \text{if}(u_j < \underline{x}_j), \\ x_{j,i1} + \text{rand}[0,\ 1]\,(\overline{x}_j - x_{j,i1}) & \text{if}(u_j > \overline{x}_j). \end{cases} \tag{5}$$

2.3 DE Algorithm

The original DE proposed by Storn and Price [1] is based on the generational model. Therefore, DE uses two populations, the primary one $\mathbf{P}$ and the secondary one $\mathbf{Q}$. The individuals $\mathbf{z}_i \in \mathbf{Q}$ $(i = 1, \cdots, N_P)$ are generated from $\mathbf{x}_i \in \mathbf{P}$. After that, $\mathbf{P}$ is replaced by $\mathbf{Q}$ at one time. The DE algorithm works as follows:

1: Randomly generate N_P individuals $\mathbf{x}_i \in \mathbf{P}$ $(i = 1, \cdots, N_P)$.
2: For each individual $\mathbf{x}_i \in \mathbf{P}$, evaluate the objective function value $f(\mathbf{x}_i)$.
3: Set the generation as $g = 0$.
4: For each of the target vectors $\mathbf{x}_i \in \mathbf{P}$, execute from Step 4.1 to Step 4.4.

4.1: Randomly select $\mathbf{x}_{i1}, \mathbf{x}_{i2}$ and $\mathbf{x}_{i3} \in \mathbf{P}$ $(i1 \neq i2 \neq i3)$.
4.2: Generate the trial vector $\mathbf{u}$ from (4) and (5).
4.3: Evaluate the objective function value $f(\mathbf{u})$.
4.4: If $f(\mathbf{u}) \leq f(\mathbf{x}_i)$ holds then let $\mathbf{z}_i = \mathbf{u}$ and $f(\mathbf{z}_i) = f(\mathbf{u})$, otherwise let $\mathbf{z}_i = \mathbf{x}_i$ and $f(\mathbf{z}_i) = f(\mathbf{x}_i)$.

5: Replace $\mathbf{P}$ by $\mathbf{Q}$ such that $\mathbf{x}_i = \mathbf{z}_i$ and $f(\mathbf{x}_i) = f(\mathbf{z}_i)$ $(i = 1, \cdots, N_P)$.
6: If $g < G_M$ holds then update the generation as $g = g + 1$ and return to Step 4.
7: Output the best $\mathbf{x}_b \in \mathbf{P}$ with the minimum $f(\mathbf{x}_b)$ and terminate.

where the maximum number of generation G_M is also a control parameter.

In Step 5 of the above DE, the primary population $\mathbf{P}$ is replaced by the secondary population $\mathbf{Q}$ at one time. Therefore, even if every $\mathbf{z}_i \in \mathbf{Q}$ $(i = 1, \cdots, N_P)$ is generated in parallel, the procedure of the parallelized DE has to be synchronized in Step 5.

2.4 SDE Algorithm

SDE is an alternative DE based on the steady-state model. Therefore, SDE cleverly uses only one population **P**. As we can see in Step 4.4 of SDE, an excellent trial vector **u** is added in the population **P** instantly and used to generate new individuals descended from the trial vector. The SDE algorithm works as follows:

1: Randomly generate N_P individuals $\mathbf{x}_i \in \mathbf{P}$ $(i = 1, \cdots, N_P)$.
2: For each individual $\mathbf{x}_i \in \mathbf{P}$, evaluate the objective function value $f(\mathbf{x}_i)$.
3: Set the generation as $g = 0$.
4: For each of the target vectors $\mathbf{x}_i \in \mathbf{P}$, execute from Step 4.1 to Step 4.4.

4.1: Randomly select $\mathbf{x}_{i1}$, $\mathbf{x}_{i2}$ and $\mathbf{x}_{i3} \in \mathbf{P}$ $(i1 \neq i2 \neq i3)$.
4.2: Generate the trial vector **u** from (4) and (5).
4.3: Evaluate the objective function value $f(\mathbf{u})$.
4.4: If $f(\mathbf{u}) \leq f(\mathbf{x}_i)$ holds, then let $\mathbf{x}_i = \mathbf{u}$ and $f(\mathbf{x}_i) = f(\mathbf{u})$.

5: If $g < G_M$ holds, then update the generation as $g = g + 1$ and return to Step 4.
6: Output the best $\mathbf{x}_b \in \mathbf{P}$ with the minimum $f(\mathbf{x}_b)$ and terminate.

3 Concurrent Differential Evolution

3.1 Concurrent Implementation

A system is said to be concurrent if it can support two or more actions in progress at the same time. A system is said to be parallel if it can support two or more actions executing simultaneously. The key difference between these definitions is the phrase "in progress" [10]. A concurrent program consists of multiple processes, or threads, executable in parallel. If a multi-core CPU has N_T $(N_T \geq 1)$ cores, a maximum number of N_T threads can be run in parallel. How and when one of the threads is assigned to a core is dictated by the scheduler of the Operating System (OS).

Each thread has its own private working memory and no thread can access other threads' working memories. Besides, there is a common memory, which is shared between all threads. In order for multiple threads to use the common memory, mutual exclusion is needed, which will be discussed in Sect. 3.5. For accessing the common memory, we assume Concurrent Read and Exclusive Write (CREW) [10] in which multiple threads may read the same memory location at the same time but only one thread may write to a given memory location at any time. Of course, multiple threads can't read and write the same memory location at the same time.

3.2 CDE Main Routine

As stated above, CDE is a concurrent version of SDE. CDE consists of a main routine and N_T subroutines named `Worker(n)` ($n = 1, \cdots, N_T$). The main routine and each of the subroutines are realized as one thread, respectively. The population $\mathbf{P}$ is stored in the common memory, which is shared between all `Worker(n)` threads. Except the detail of `Worker(n)`, the main routine of CDE is provided as

1: Randomly generate N_P individuals $\mathbf{x}_i \in \mathbf{P}$ ($i = 1, \cdots, N_P$).
2: Evoke `Worker(n)` ($n = 1, \cdots, N_T$) in parallel.
3: Wait until every `Worker(n)` ($n = 1, \cdots, N_T$) is completed.
4: Output the best $\mathbf{x}_b \in \mathbf{P}$ with the minimum $f(\mathbf{x}_b)$ and terminate.

CDE is based on a programming model known as "MapReduce" [21], which has been employed widely to design various concurrent and parallel applications lately. The programming model consists of two phase, "Map-phase" and "Reduce-phase." In the main routine of CDE, Step 2 corresponds to "Map-phase," while Step 4 corresponds to "Reduce-phase." All `Worker(n)` threads are evoked and executed concurrently in Step 2. Each `Worker(n)` generates the trial vector $\mathbf{u}$ from several individuals $\mathbf{x}_i \in \mathbf{P}$ and evaluates its objective function value $f(\mathbf{u})$ by using its private working memory. The results of all `Worker(n)` threads are consolidated in Step 4. For assigning the target vector $\mathbf{x}_i \in \mathbf{P}$ ($i = 1, \cdots, N_P$) to `Worker(n)` ($n = 1, \cdots, N_T$), we propose two techniques, the dynamic allocation of tasks and the static allocation of tasks, which are described in the following subsections.

3.3 Dynamic Task Allocation

CDE with the dynamic allocation of tasks is named CDE/D. CDE/D allocates the target vector $\mathbf{x}_i \in \mathbf{P}$ to `Worker(n)` over time as CDE/D progresses. `GetIndex()` denotes an exclusive function that returns a unique index at a time in ascending order such as $t = 1, 2, \cdots$. By using `GetIndex()`, `Worker(n)` gets an index of the assigned target vector dynamically. Because each $\mathbf{x}_i \in \mathbf{P}$ is possibly updated by every `Worker(n)`, `Worker(n)` has to overwrite $\mathbf{x}_i \in \mathbf{P}$ and $f(\mathbf{x}_i)$ in Step 3.5 under the exclusion control. The procedure of `Worker(n)` is provided as

1: Get an index as $t =$ `GetIndex()`.
2: While $t \leq N_P$ holds, evaluate the objective function value $f(\mathbf{x}_t)$ and get the next index such as $t =$ `GetIndex()`.
3: While $t \leq (G_M + 1)\, N_P$ holds, execute from Step 3.1 to Step 3.6.

3.1: Designate an individual $\mathbf{x}_i$ ($i = t\%N_P + 1$) to the target vector.
3.2: Randomly select $\mathbf{x}_{i1}$, $\mathbf{x}_{i2}$ and $\mathbf{x}_{i3} \in \mathbf{P}$ ($i1 \neq i2 \neq i3$).

3.3: Generate the trial vector $\mathbf{u}$ from (4) and (5).
3.4: Evaluate the objective function value $f(\mathbf{u})$.
3.5: If $f(\mathbf{u}) \leq f(\mathbf{x}_i)$ holds, then let $\mathbf{x}_i = \mathbf{u}$ and $f(\mathbf{x}_i) = f(\mathbf{u})$.
3.6: Get the next index such as $t =$ `GetIndex()`.

3.4 Static Task Allocation

CDE with the static allocation of tasks is named CDE/S. First of all, the population $\mathbf{P}$ is divided into N_T sub-populations $\mathbf{P}_n$ ($n = 1, \cdots, N_T$) called "chunks" as

$$\mathbf{P} = \mathbf{P}_1 \cup \cdots \cup \mathbf{P}_n \cup \cdots \cup \mathbf{P}_{N_T}, \tag{6}$$

where an individual $\mathbf{x}_i \in \mathbf{P}$ ($i \% N_T = n$) is assigned to $\mathbf{P}_{n+1}$.

CDE/S allocates each chunk $\mathbf{P}_n$ to `Worker(n)` statically. `Worker(n)` can read any individual $\mathbf{x}_i \in \mathbf{P}$, but it may overwrite only the individuals $\mathbf{x}_i \in \mathbf{P}_n$. In other words, the task for updating the individuals in one chunk $\mathbf{P}_n$ is permitted only to `Worker(n)`. Therefore, the mutual exclusion among threads overwriting $\mathbf{x}_i \in \mathbf{P}_n$ is not necessary for CDE/S. The procedure of `Worker(n)` is provided as

1: For each individual $\mathbf{x}_i \in \mathbf{P}_n$, evaluate the objective function value $f(\mathbf{x}_i)$.
2: Set the generation as $g = 0$.
3: For each of the target vectors $\mathbf{x}_i \in \mathbf{P}_n$, execute from Step 3.1 to Step 3.4.

3.1: Randomly select $\mathbf{x}_{i1}$, $\mathbf{x}_{i2}$ and $\mathbf{x}_{i3} \in \mathbf{P}$ ($i1 \neq i2 \neq i3$).
3.2: Generate the trial vector $\mathbf{u}$ from (4) and (5).
3.3: Evaluate the objective function value $f(\mathbf{u})$.
3.4: If $f(\mathbf{u}) \leq f(\mathbf{x}_i)$ holds, then let $\mathbf{x}_i = \mathbf{u}$ and $f(\mathbf{x}_i) = f(\mathbf{u})$.

4: If $g < G_M$ holds, then update the generation as $g = g + 1$ and return to Step 3.

3.5 Implementation of CDE in Java

The SDE, CDE/D, and CDE/S algorithms are coded in Java, a very popular language supporting multi-threading and mutual exclusion between threads. In CDE, most threads are just reading the individuals in the population. Overwriting the target vector $\mathbf{x}_i \in \mathbf{P}$ seldom occurs. It is not necessary to exclusively lock access to $\mathbf{x}_i \in \mathbf{P}$ while reading because multiple read operations can be done in parallel unless there is an ongoing write operation. Read–write lock provided by Java enforces a multiple-reader and single-writer locking discipline: more than one reader can access the shared resource concurrently so long as none of them wants to modify it, but writers must acquire the lock exclusively [11].

By using the read–write lock, CWER assumed in Sect. 3.1 can be realized easily. In the implementation of CDE/D, the population $\mathbf{P}$ is equipped with two pairs of read and write locks. The first pair is used to access all individuals $\mathbf{x}_i \in \mathbf{P}$, while the second pair is used to access all objective function values $f(\mathbf{x}_i)$ ($\mathbf{x}_i \in \mathbf{P}$). That is because every `Worker(n)` of CDE/D is permitted to read and write the objective function values $f(\mathbf{x}_i)$ ($\mathbf{x}_i \in \mathbf{P}$). On the other hand, in the implementation of CDE/S, each chunk $\mathbf{P}_n$ ($n = 1, \cdots, N_T$) is equipped with a pair of read and write locks. Then the pair of read and write locks is used to access the individuals $\mathbf{x}_i \in \mathbf{P}_n$. Because a unique thread of CDE/S, i.e., `Worker(n)`, is permitted to read and write the objective function values $f(\mathbf{x}_i)$ ($\mathbf{x}_i \in \mathbf{P}_n$), no lock is necessary to access $f(\mathbf{x}_i)$.

4 Experimentation

4.1 Benchmark Problems

The following six test functions are used as the objective function $f(\mathbf{x})$ shown in (1). All benchmark problems are D-dimensional real functions with $D = 50$. The function values of the optimal solutions $\mathbf{x}^\star \in \Re^D$ are known to be $f_p(\mathbf{x}^\star) = 0$ ($p = 1, \cdots, 6$).

- Sphere function (unimodal function):

$$f_1(\mathbf{x}) = \sum_{j=1}^{D} x_j^2,$$
$$-100 \leq x_j \leq 100, \ j = 1, \cdots, D.$$

- Salomon function (multimodal function):

$$f_2(\mathbf{x}) = -\cos\left(2\pi \sqrt{\sum_{j=1}^{D} x_j^2}\right) + 0.1 \sqrt{\sum_{j=1}^{D} x_j^2} + 1,$$
$$-100 \leq x_j \leq 100, \ j = 1, \cdots, D.$$

- Rosenbrock function (multimodal function):

$$f_3(\mathbf{x}) = \sum_{j=1}^{D-1} (100\,(x_j^2 - x_{j+1})^2 + (1 - x_j)^2),$$
$$-2.048 \leq x_j \leq 2.048, \ j = 1, \cdots, D$$

Table 1 Specifications for Personal Computers (PCs)

PC	CPU	OS	Clock	Memory
PC1	Intel(R) Core(TM) i7	WindowsXP	3.34 GHz	2.99 GB
PC2	AMD Phenom(TM) II X6	Windows7	3.20 GHz	3.25 GB

- Rastrigin function (multimodal function):

$$f_4(\mathbf{x}) = \sum_{j=1}^{D} (x_j^2 - 10\cos(2\pi x_j) + 10),$$
$$-5.12 \leq x_j \leq 5.12,\ j = 1, \cdots, D.$$

- Ackley function (multimodal function):

$$f_5(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{j=1}^{D} x_j^2}\right) - \exp\left(\frac{1}{D}\sum_{j=1}^{D}\cos(2\pi x_j)\right) + 20 + e.$$
$$-32.768 \leq x_j \leq 32.768,\ j = 1, \cdots, D.$$

- Griewank function (multimodal function):

$$f_6(\mathbf{x}) = \frac{1}{4000}\sum_{j=1}^{D} x_j^2 - \prod_{j=1}^{D}\cos\left(\frac{x_j}{\sqrt{j}}\right) + 1,$$
$$-600 \leq x_j \leq 600,\ j = 1, \cdots, D.$$

4.2 Experimental Setup

The Java programs of SDE, CDE/D, and CDE/S were executed on two Personal Computers (PCs) equipped with different multi-core CPUs. Table 1 shows the specifications for the two PCs which are denoted by PC1 and PC2. The multi-core CPU in PC1 has four cores each of which manipulates two threads at the same time, while the multi-core CPU in PC2 has six cores each of which manipulates one thread. The control parameters of all algorithms were chosen as $F = 0.5$, $C_R = 0.9$, and $G_M = 1{,}000$. The population size N_P was set to either 100 or 500. Then the three algorithms were applied to the six benchmark problems 50 times, respectively.

Table 2 Execution time of SDE on PC1

N_P	f_1	f_2	f_3	f_4	f_5	f_6
100	129.06	130.94	138.76	402.50	355.32	394.38
	(7.59)	(8.32)	(8.14)	(9.81)	(10.27)	(8.15)
500	634.38	659.36	682.18	2021.86	1788.12	1974.06
	(12.69)	(13.80)	(12.40)	(11.37)	(13.83)	(13.34)

Table 3 Execution time of SDE on PC2

N_P	f_1	f_2	f_3	f_4	f_5	f_6
100	173.64	186.72	181.74	604.80	445.24	459.90
	(5.14)	(4.32)	(8.31)	(6.81)	(8.42)	(8.42)
500	898.44	916.20	913.42	3056.72	2293.78	2346.92
	(11.79)	(33.13)	(7.58)	(26.45)	(24.62)	(36.41)

4.3 CDE Execution Time

Table 2 shows the average execution time of SDE on PC1, where the standard deviation of the execution times also appears in parentheses. Similarly, Table 3 shows the execution time of SDE on PC2. The efficiency of CDE is evaluated by using the speedup $S_m(N_T)$ defined in (7), where T_m is the execution time of SDE and $T_m(N_T)$ is the execution time of CDE using N_T ($N_T \geq 1$) `Worker(n)` threads. Because SDE and CDE are stochastic algorithms, T_m and $T_m(N_T)$ are averaged over $m = 50$ runs.

$$S_m(N_T) = \frac{T_m}{T_m(N_T)} \tag{7}$$

Figure 1 compares the speedup curves achieved, respectively, by CDE/D (broken line) and CDE/S (solid line) on PC1. Figure 2 also compares the speedup curves achieved by CDE/D and CDE/S on PC2 in the same way. Because the maximum number of threads executable in parallel is different, the characteristics of the speedup curves differ between two PCs. For example, in case of PC1, the speedup curves of CDE/S are saturated at $N_T = 8$ threads. On the other hand, in case of PC2, the speedup curves of CDE/S are saturated at $N_T = 6$ threads. However, the speedup achieved by CDE/S is higher than the speedup achieved by CDE/D in every case. Therefore, from Figs. 1 and 2, it can be confirmed that CDE/S utilizes multi-core CPUs more efficiently than CDE/D on both PCs. That is because CDE/D spends a relevant overhead for the exclusion control between write operations to $\mathbf{x}_i \in \mathbf{P}$.

Incidentally, from Figs. 1 and 2, the characteristics of the speedup curves achieved by CDE (CDE/D and CDE/S) depend on the kind of benchmark problems rather than on population size. More specifically, we can observe that the higher speedup has been achieved by CDE in case of the more expensive objective

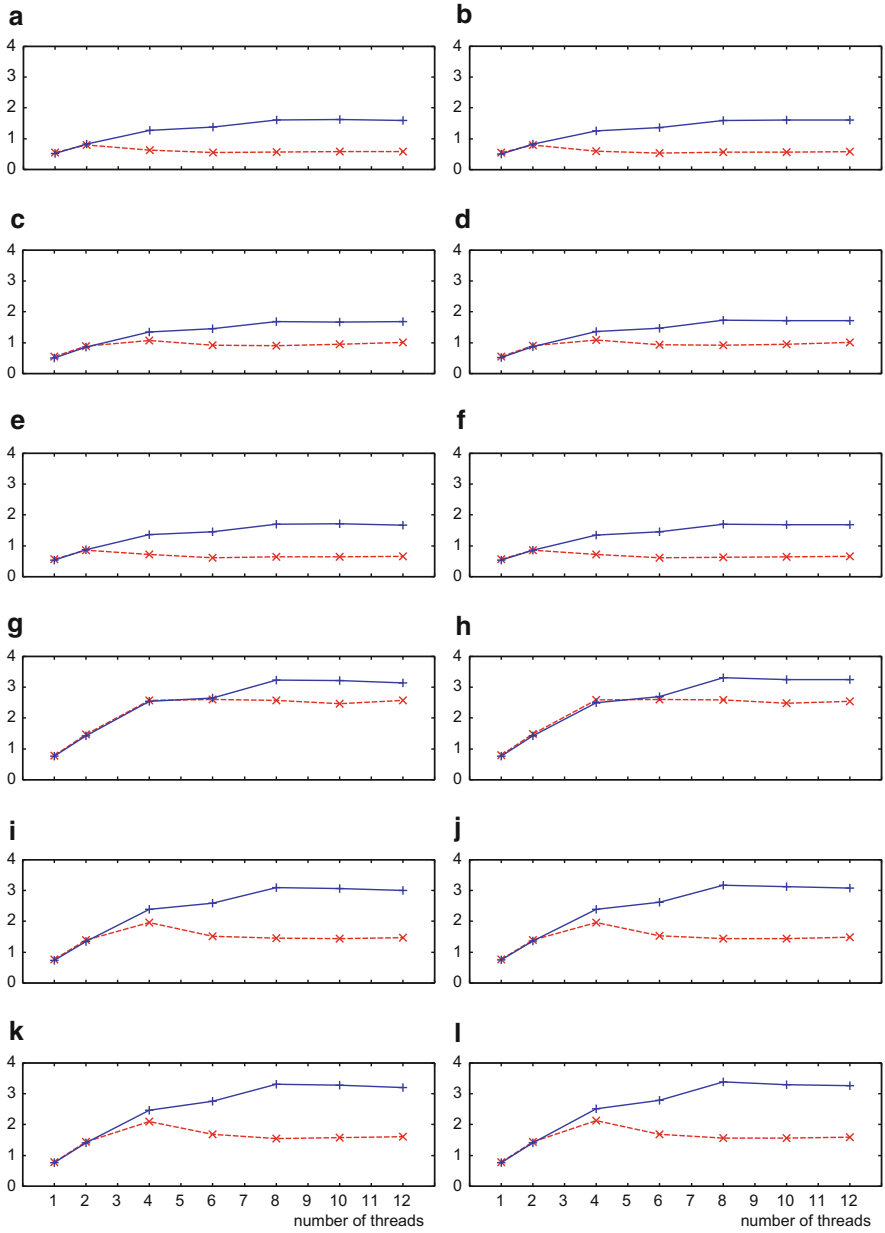

Fig. 1 Speedup curves achieved by CDE/D (*broken line*) and CDE/S (*solid line*) on PC1. (**a**) Sphere function: f_1 ($N_P = 100$). (**b**) Sphere function: f_1 ($N_P = 500$). (**c**) Salomon function: f_2 ($N_P = 100$). (**d**) Salomon function: f_2 ($N_P = 500$). (**e**) Rosenbrock function: f_3 ($N_P = 100$). (**f**) Rosenbrock function: f_3 ($N_P = 500$). (**g**) Rastrigin function: f_4 ($N_P = 100$). (**h**) Rastrigin function: f_4 ($N_P = 500$). (**i**) Ackley function: f_5 ($N_P = 100$). (**j**) Ackley function: f_5 ($N_P = 500$). (**k**) Griewank function: f_6 ($N_P = 100$). (**l**) Griewank function: f_6 ($N_P = 500$)

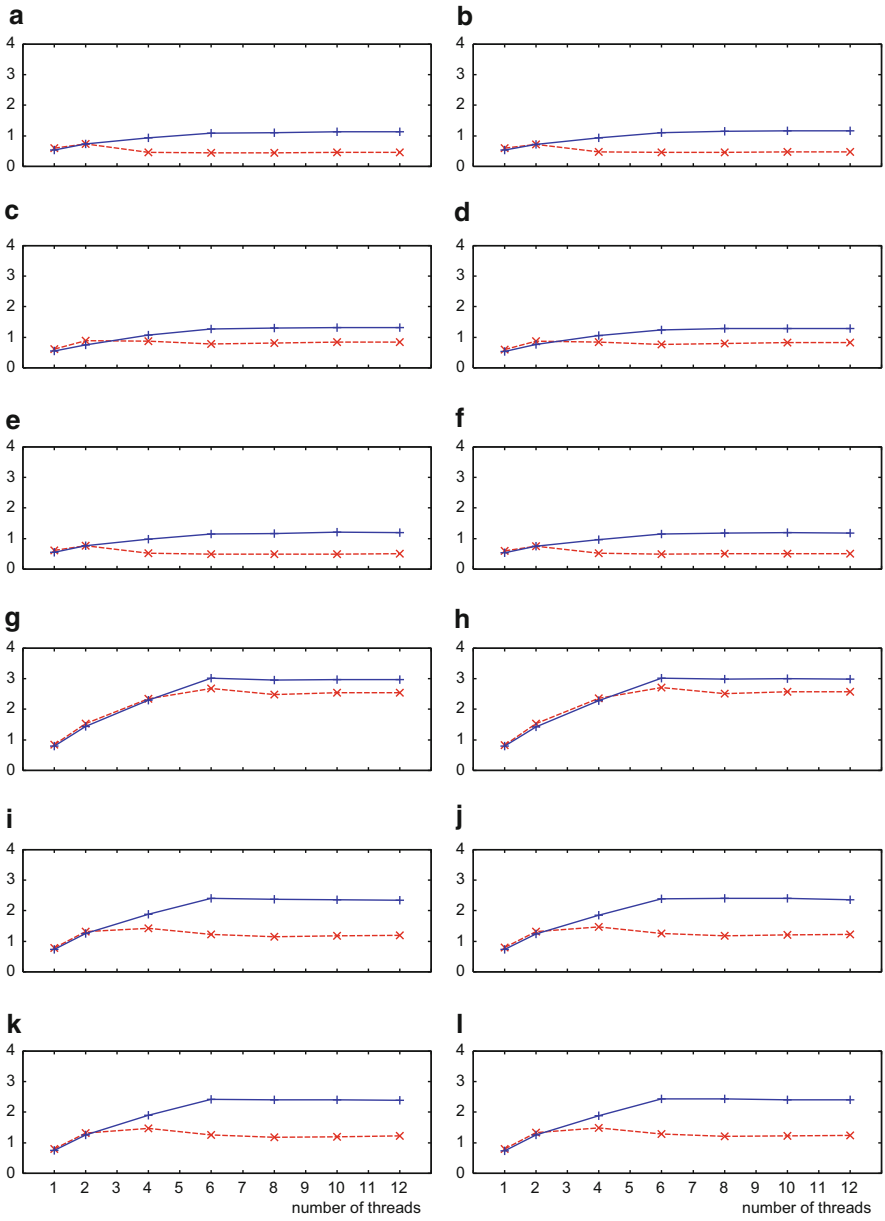

Fig. 2 Speedup curves achieved by CDE/D (*broken line*) and CDE/S (*solid line*) on PC2. (**a**) Sphere function: f_1 ($N_P = 100$). (**b**) Sphere function: f_1 ($N_P = 500$). (**c**) Salomon function: f_2 ($N_P = 100$). (**d**) Salomon function: f_2 ($N_P = 500$). (**e**) Rosenbrock function: f_3 ($N_P = 100$). (**f**) Rosenbrock function: f_3 ($N_P = 500$). (**g**) Rastrigin function: f_4 ($N_P = 100$). (**h**) Rastrigin function: f_4 ($N_P = 500$). (**i**) Ackley function: f_5 ($N_P = 100$). (**j**) Ackley function: f_5 ($N_P = 500$). (**k**) Griewank function: f_6 ($N_P = 100$). (**l**) Griewank function: f_6 ($N_P = 500$)

Table 4 Objective function value of the best solution obtained by SDE on PC1

N_P	f_1	f_2	f_3	f_4	f_5	f_6
100	3.28E−4	1.660	44.492	99.963	4.24E−3	1.24E−3
	(5.90E−5)	(0.108)	(0.523)	(6.297)	(5.45E−4)	(1.96E−3)
500	5.04E−4	1.677	44.320	93.686	5.35E−3	9.94E−4
	(4.97E−5)	(0.060)	(0.337)	(5.651)	(3.65E−4)	(2.63E−4)

Table 5 Objective function value of the best solution obtained by SDE on PC2

N_P	f_1	f_2	f_3	f_4	f_5	f_6
100	3.28E−4	1.660	44.492	99.963	4.24E−3	1.24E−3
	(5.90E−5)	(0.108)	(0.523)	(6.297)	(5.45E−4)	(1.96E−3)
500	5.04E−4	1.677	44.320	93.686	5.35E−3	9.94E−4
	(4.97E−5)	(0.060)	(0.337)	(5.651)	(3.65E−4)	(2.63E−4)

functions, namely f_4, f_5, and f_6, which are computed using a lot of trigonometric functions.

4.4 Quality of Solutions Obtained by CDE

Table 4 shows the average objective function values of the best solutions obtained by SDE on PC1, where the standard deviation of the objective function values also appears in parentheses. Similarly, Table 5 shows the objective function values found by SDE on PC2. Table 6 shows the average objective function values of the best solutions obtained by CDE (CDE/D and CDE/S) with $N_P = 100$ and 500 on PC1, where the standard deviation of the objective function values also appears in parentheses. Similarly, Table 7 shows the objective function values found by CDE on PC2. From Tables 6 and 7, the objective function values found by CDE, i.e., the quality of solutions, seem to change slightly with the number of threads in each of benchmark problems. Furthermore, the quality of solutions seems to be different in CDE/D and CDE/S. Therefore, we analyze the quality of solutions statistically by Wilcoxon test [22]. The null hypothesis is that there is no significant difference between two objective function values found, respectively, by SDE and CDE.

Table 8 compares CDE (CDE/D or CDE/S) with SDE on PC1 by using Wilcoxon test about the objective function values of the best solutions, in which $\Downarrow$ ($\Uparrow$) denotes CDE is significantly better (worse) than SDE with risk $\alpha = 0.01$; $\downarrow$ ($\uparrow$) denotes CDE is better (worse) than SDE with risk $\alpha = 0.05$; and "–" means that there is no significant difference between CDE and SDE in the objective function value. Similarly, Table 9 compares CDE (CDE/D or CDE/S) with SDE on PC2 by using Wilcoxon test about the objective function values of the best solutions.

Table 6 Objective function value of the best solution obtained by CDE on PC1

N_T	1	2	4	6	8	10	12
(a) CDE/D with $N_P = 100$							
f_1	3.28E−4 (5.90E−5)	3.39E−4 (8.35E−5)	3.51E−4 (7.08E−5)	3.63E−4 (7.20E−5)	3.56E−4 (7.47E−5)	3.53E−4 (7.22E−5)	3.89E−4 (8.29E−5)
f_2	1.660 (0.108)	1.660 (0.083)	1.672 (0.100)	1.678 (0.104)	1.665 (0.108)	1.667 (0.08)	1.679 (0.122)
f_3	44.492 (0.523)	44.571 (0.472)	44.486 (0.540)	44.582 (0.585)	44.508 (0.568)	44.526 (0.624)	44.574 (0.679)
f_4	99.963 (6.297)	98.128 (8.422)	99.423 (5.836)	99.089 (6.647)	101.059 (7.554)	101.010 (6.765)	100.808 (7.211)
f_5	4.24E−3 (5.45E−4)	4.22E−3 (4.98E−4)	4.30E−3 (4.56E−4)	4.39E−3 (5.25E−4)	4.33E−3 (5.30E−4)	4.44E−3 (4.97E−4)	4.54E−3 (4.63E−4)
f_6	1.24E−3 (1.96E−3)	1.03E−3 (1.30E−3)	1.08E−3 (1.89E−3)	1.14E−3 (1.16E−3)	8.38E−4 (6.14E−4)	1.16E−3 (1.25E−3)	1.02E−3 (1.26E−3)
(b) CDE/S with $N_P = 100$							
f_1	3.28E−4 (5.90E−5)	3.10E−4 (6.74E−5)	2.93E−4 (5.86E−5)	6.16E−4 (1.55E−4)	3.03E−4 (7.10E−5)	3.02E−4 (5.40E−5)	3.62E−4 (7.59E−5)
f_2	1.660 (0.108)	1.657 (0.083)	1.657 (0.094)	1.639 (0.099)	1.601 (0.101)	1.653 (0.102)	1.603 (0.100)
f_3	44.492 (0.523)	44.472 (0.565)	44.453 (0.605)	45.051 (0.580)	44.489 (0.571)	44.544 (0.553)	44.561 (0.540)
f_4	99.963 (6.297)	100.449 (6.406)	99.992 (6.400)	98.014 (7.730)	99.941 (6.484)	98.645 (8.234)	98.578 (8.394)
f_5	4.24E−3 (5.45E−4)	4.35E−3 (3.96E−4)	4.17E−3 (5.08E−4)	6.31E−3 (8.46E−4)	4.21E−3 (5.00E−4)	4.12E−3 (4.97E−4)	4.89E−3 (8.13E−4)
f_6	1.24E−3 (1.96E−3)	1.23E−3 (2.00E−3)	9.01E−4 (6.74E−4)	1.76E−3 (1.48E−3)	7.13E−4 (3.80E−4)	7.87E−4 (6.05E−4)	1.71E−3 (2.13E−3)
(c) CDE/D with $N_P = 500$							
f_1	5.04E−4 (4.97E−5)	5.17E−4 (6.55E−5)	5.15E−4 (7.29E−5)	5.13E−4 (5.67E−5)	5.06E−4 (5.72E−5)	5.32E−4 (6.40E−5)	4.99E−4 (4.93E−5)
f_2	1.677 (0.060)	1.667 (0.070)	1.671 (0.079)	1.677 (0.072)	1.672 (0.063)	1.668 (0.074)	1.644 (0.075)
f_3	44.320 (0.337)	44.272 (0.356)	44.180 (0.396)	44.313 (0.315)	44.349 (0.261)	44.226 (0.318)	44.278 (0.401)
f_4	93.686 (5.651)	93.357 (4.243)	93.765 (5.547)	94.748 (5.783)	92.623 (6.124)	92.829 (5.259)	94.493 (5.424)
f_5	5.35E−3 (3.65E−4)	5.41E−3 (3.40E−4)	5.42E−3 (3.67E−4)	5.32E−3 (3.45E−4)	5.45E−3 (3.83E−4)	5.49E−3 (3.09E−4)	5.45E−3 (3.17E−4)
f_6	9.94E−4 (2.63E−4)	1.09E−3 (3.72E−4)	1.11E−3 (2.65E−4)	1.07E−3 (3.33E−4)	1.14E−3 (4.05E−4)	1.04E−3 (3.42E−4)	1.04E−3 (2.78E−4)

(continued)

Table 6 (continued)

N_T	1	2	4	6	8	10	12
(d) CDE/S with $N_P = 500$							
f_1	4.96E−4 (8.28E−5)	5.04E−4 (4.64E−5)	4.38E−4 (6.52E−5)	1.11E−3 (1.70E−4)	4.46E−4 (5.69E−5)	4.20E−4 (6.26E−5)	4.29E−4 (6.29E−5)
f_2	1.677 (0.060)	1.666 (0.069)	1.644 (0.074)	1.620 (0.067)	1.652 (0.071)	1.638 (0.060)	1.642 (0.068)
f_3	44.320 (0.337)	44.214 (0.307)	44.182 (0.325)	45.035 (0.346)	44.192 (0.311)	44.299 (0.379)	44.168 (0.350)
f_4	93.686 (5.651)	92.221 (6.757)	92.367 (4.943)	90.986 (5.412)	90.654 (7.439)	94.420 (5.077)	94.505 (4.994)
f_5	5.35E−3 (3.65E−4)	5.33E−3 (3.47E−4)	5.00E−3 (3.87E−4)	8.50E−3 (6.44E−4)	5.11E−3 (2.85E−4)	4.95E−3 (3.52E−4)	4.92E−3 (3.97E−4)
f_6	9.94E−4 (2.63E−4)	1.00E−3 (2.27E−4)	1.01E−3 (3.67E−4)	2.13E−3 (6.74E−4)	1.01E−3 (2.93E−4)	9.22E−4 (3.26E−4)	9.23E−4 (3.71E−4)

From the results of Wilcoxon test shown in Tables 8 and 9, the null hypothesis is rejected with the risk less than $\alpha = 0.01$ in some cases. In other words, it can be confirmed that the objective function values found by CDE changes significantly with the number of threads in some benchmark problems. Even though the objective function values found by CDE are smaller than those values found by SDE in several cases, we privilege the robustness of solutions in this chapter. Compared to CDE/D, CDE/S is further different from SDE in the quality of solutions. In particular, the quality of solutions obtained by CDE/S becomes unstable on both PCs when the population size and the number of threads are large. Consequently, we can conclude that CDE/D is relatively superior to CDE/S in the quality of solutions.

5 Concluding Remarks

In order to utilize multi-core CPUs effectively, a concurrent version of DE called CDE was described. Strictly speaking, CDE is a concurrent version of SDE, that is a new DE based on the steady-state model. Then two implementation techniques of CDE, namely CDE/D and CDE/S, which differ in the allocation of tasks were compared with respect to both execution time and quality of solutions. CDE/D and CDE/S were implemented in Java and executed on two popular multi-core CPUs. Incidentally, the concurrent program of CDE run on a multi-core CPU could allocate tasks to threads directly but could not assign these threads to cores by itself. Instead of CDE, the scheduler of OS assigned those threads to cores. Therefore, CDE could run on various multi-core CPUs relying on different architectures.

In the numerical experiments conducted on six benchmark problems CDE/S was always faster than CDE/D. Furthermore, even though CDE/S was always faster than

Table 7 Objective function value of the best solution obtained by CDE on PC2

N_T	1	2	4	6	8	10	12
(a) CDE/D with $N_P = 100$							
f_1	3.28E−4 (5.90E−$y5$)	3.48E−4 (5.20E−5)	3.28E−4 (6.04E−$y5$)	3.38E−4 (6.50E−5)	3.49E−$y4$ (8.46E−5)	3.62E−4 (6.19E−$y5$)	3.60E−4 (8.71E−5)
f_2	1.660 (0.108)	1.663 (0.087)	1.649 (0.095)	1.675 (0.088)	1.694 (0.116)	1.692 (0.097)	1.665 (0.094)
f_3	44.492 (0.523)	44.459 (0.476)	44.511 (0.622)	44.545 (0.574)	44.599 (0.574)	44.574 (0.511)	44.699 (0.499)
f_4	99.963 (6.297)	99.926 (6.910)	100.198 (8.006)	100.112 (8.168)	102.090 (5.522)	101.626 (6.436)	100.633 (6.566)
f_5	4.24E−3 (5.45E−$y4$)	4.28E−3 (5.10E−4)	4.27E−3 (5.46E−$y4$)	4.37E−3 (5.42E−4)	4.38E−$y3$ (4.86E−4)	4.39E−3 (5.31E−4)	4.61E−3 (5.00E−4)
f_6	1.24E−3 (1.96E−3)	1.02E−3 (1.35E−3)	1.19E−3 (1.31E−3)	1.00E−3 (1.07E−3)	1.43E−3 (2.19E−3)	9.27E−4 (7.74E−4)	1.18E−3 (1.31E−3)
(b) CDE/S with $N_P = 100$							
f_1	3.28E−4 (5.90E−5)	3.09E−4 (6.45E−5)	2.92E−4 (6.19E−$y5$)	2.97E−4 (6.81E−5)	3.27E−$y4$ (8.27E−5)	3.59E−4 (8.87E−5)	6.51E−$y4$ (2.13E−4)
f_2	1.660 (0.108)	1.656 (0.104)	1.664 (0.083)	1.644 (0.079)	1.630 (0.104)	1.627 (0.095)	1.636 (0.129)
f_3	44.492 (0.523)	44.515 (0.563)	44.444 (0.611)	44.552 (0.594)	44.491 (0.620)	44.596 (0.607)	44.892 (0.604)
f_4	99.963 (6.297)	100.221 (7.424)	100.088 (6.129)	99.561 (7.097)	101.903 (7.338)	99.761 (6.631)	100.053 (6.470)
f_5	4.24E−3 (5.45E−4)	4.22E−$y3$ (4.68E−4)	4.21E−3 (4.62E−$y4$)	4.23E−3 (4.59E−4)	4.22E−3 (5.14E−4)	4.87E−3 (1.12E−3)	6.56E−$y3$ (1.52E−3)
f_6	1.24E−3 (1.96E−3)	1.09E−3 (8.90E−4)	1.22E−3 (1.55E−3)	8.40E−$y4$ (8.33E−4)	8.18E−4 (9.03E−4)	1.26E−3 (1.36E−3)	2.94E−3 (4.31E−3)
(c) CDE/D with $N_P = 500$							
f_1	5.04E−4 (4.97E−$y5$)	5.15E−4 (5.75E−5)	5.19E−4 (6.14E−5)	5.07E−4 (6.65E−5)	5.19E−4 (4.30E−5)	5.08E−4 (6.74E−$y5$)	5.07E−4 (6.02E−5)
f_2	1.677 (0.060)	1.671 (0.060)	1.649 (0.092)	1.648 (0.085)	1.669 (0.069)	1.659 (0.093)	1.675 (0.072)
f_3	44.320 (0.337)	44.384 (0.269)	44.154 (0.399)	44.279 (0.325)	44.258 (0.343)	44.282 (0.362)	44.227 (0.300)
f_4	93.686 (5.651)	91.505 (6.612)	94.256 (4.683)	92.879 (6.357)	93.983 (5.037)	92.570 (6.939)	92.670 (5.793)
f_5	5.35E−3 (3.65E−4)	5.40E−3 (3.50E−4)	5.36E−$y3$ (3.03E−4)	5.32E−3 (3.23E−4)	5.33E−3 (3.40E−4)	5.45E−3 (3.10E−4)	5.40E−3 (3.27E−4)
f_6	9.94E−4 (2.63E−4)	1.03E−3 (2.41E−4)	1.03E−3 (2.84E−4)	1.23E−3 (5.52E−4)	1.09E−3 (4.82E−4)	1.06E−3 (2.67E−$y4$)	1.07E−3 (2.96E−4)

(continued)

Table 7 (continued)

N_T	1	2	4	6	8	10	12
(d) CDE/S with $N_P = 500$							
f_1	5.04E−4 (4.97E−5)	4.83E−4 (5.78E−5)	4.69E−4 (7.26E−$y5$)	4.32E−4 (5.92E−5)	4.20E−$y4$ (6.28E−5)	4.38E−4 (6.47E−5)	5.00E−$y4$ (9.34E−5)
f_2	1.677 (0.060)	1.668 (0.064)	1.651 (0.082)	1.641 (0.084)	1.641 (0.080)	1.619 (0.084)	1.620 (0.064)
f_3	44.320 (0.337)	44.226 (0.355)	44.151 (0.359)	44.147 (0.408)	44.250 (0.345)	44.253 (0.365)	44.445 (0.349)
f_4	93.686 (5.651)	93.730 (4.793)	92.233 (6.178)	93.374 (6.018)	94.151 (5.522)	93.156 (5.473)	91.825 (6.138)
f_5	5.35E−3 (3.65E−$y4$)	5.37E−$y3$ (3.40E−4)	5.15E−3 (3.97E−$y4$)	5.00E−3 (3.80E−4)	4.81E−$y3$ (3.32E−4)	4.94E−3 (3.34E−4)	5.19E−3 (3.95E−4)
f_6	9.94E−$y4$ (2.63E−4)	1.02E−3 (3.74E−4)	9.88E−4 (2.99E−4)	8.96E−$y4$ (2.22E−4)	8.59E−4 (3.17E−4)	8.99E−4 (2.66E−$y4$)	1.32E−3 (6.11E−4)

Table 8 Wilcoxon test between SDE and CDE on PC1

N_P	100							500						
N_T	1	2	4	6	8	10	12	1	2	4	6	8	10	12
(a) CDE/D														
f_1	—	—	—	↑	—	—	⇑	—	—	—	—	—	↑	—
f_2	—	—	—	—	—	—	—	—	—	—	—	—	—	⇓
f_3	—	—	—	—	—	—	—	—	—	↓	—	—	—	—
f_4	—	—	—	—	—	—	—	—	—	—	—	—	—	—
f_5	—	—	—	—	—	↑	⇑	—	—	—	—	—	↑	—
f_6	—	—	—	—	—	↑	—	—	—	—	↑	—	—	—
(b) CDE/S														
f_1	—	—	⇓	⇑	↓	↓	↑	—	—	⇓	⇑	⇓	⇓	⇓
f_2	—	—	—	—	↑	—	⇓	—	↓	⇓	⇓	↓	⇓	⇓
f_3	—	—	—	⇑	—	—	—	—	—	—	⇑	—	—	↓
f_4	—	—	—	—	—	—	—	—	—	—	↓	↓	—	—
f_5	—	—	—	⇑	—	—	⇑	—	—	⇓	⇑	⇓	⇓	⇓
f_6	—	—	—	⇑	—	—	⇑	—	—	—	⇑	—	—	↓

SDE too, CDE/D was slower than SDE in several benchmark problems. Therefore, it could be concluded that CDE/S utilizes multi-core CPUs more effectively. On the other hand, the quality of solutions obtained by CDE/S is likely to change with the number of threads and the kind of benchmark problems. Actually, by using the statistical test, i.e., Wilcoxon test, it was confirmed that CDE/S was inferior

Table 9 Wilcoxon test between SDE and CDE on PC2

N_P	100							500						
N_T	1	2	4	6	8	10	12	1	2	4	6	8	10	12
(a) CDE/D														
f_1	—	—	—	—	—	⇑	—	—	—	—	—	—	—	—
f_2	—	—	—	—	—	—	—	—	—	—	↓	—	—	—
f_3	—	—	—	—	—	—	—	—	—	↓	—	—	—	—
f_4	—	—	—	—	—	—	—	—	—	—	—	—	—	—
f_5	—	—	—	—	—	—	⇑	—	—	—	—	—	—	—
f_6	—	—	—	—	—	—	↑	—	—	—	⇑	—	—	—
(b) CDE/S														
f_1	—	—	⇓	↓	—	—	⇑	—	—	↓	⇓	⇓	⇓	—
f_2	—	—	—	—	—	—	—	—	—	—	↓	↓	⇓	⇓
f_3	—	—	—	—	—	—	⇑	—	—	↓	↓	—	—	—
f_4	—	—	—	—	—	—	—	—	—	—	—	—	—	—
f_5	—	—	—	—	—	⇑	⇑	—	—	⇓	⇓	⇓	⇓	↓
f_6	—	—	—	—	—	⇑	⇑	—	—	—	—	⇓	↓	⇑

to CDE/D in the quality of solutions. Consequently, the current CDE/S leaves much to be desired.

Another implementation technique of CDE that enhances the performance of DE as regards both the execution time and the quality of solutions is a future challenge. Besides, CDE needs to be extended for structured population models [23].

References

1. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous space. J. Global Optim. **4**(11), 341–359 (1997)
2. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution - A Practical Approach to Global Optimization. Springer, Berlin (2005)
3. Das, S., Suganthan, P.N.: Differential evolution: a survey of the state-of-the art. IEEE Trans. Evolut. Comput. **15**(1), 4–31 (2011)
4. Cantú-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic, Boston (2001)
5. Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. IEEE Trans. Evolut. Comput. **5**(6), 443–462 (2002)
6. Tasoulis, D.K., Pavlidis, N.G., Plagianakos, V.P., Vrahatis, M.N.: Parallel differential evolution. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 2023–2029 (2004)
7. Zaharie, D., Petcu, D.: Parallel implementation of multi-population differential evolution. In: Nicolau, A., Grigoras, D. (eds.) Concurrent Information Processing and Computing, pp. 223–232. IOS Press, Amsterdam (2005)

8. Zhou, C.: Fast parallelization of differential evolution algorithm using MapReduce. In: Proceedings of Genetic and Evolutionary Computation Conference, pp. 1113–1114 (2010)
9. Ishimizu, T., Tagawa, K.: Experimental study of a structured differential evolution with mixed strategies. J. Adv. Comput. Intell. Intell. Inform. **15**(9), 1310–1319 (2011)
10. Breshears, C.: The Art of Concurrency - A Thread Monkey's Guide to Writing Parallel Applications. O'Reilly, Cambridge (2009)
11. Goetz, B., et al.: Java Concurrency in Practice. Addison-Wesley, Upper Saddle River (2006)
12. Diaz, J., Muñoz-Caro, C., Niño, A.: A survey of parallel programming models and tools in the multi and many-core era. IEEE Trans. Parall. Distr. Syst. **23**(8), 1369–1386 (2012)
13. de Veronese, L., Krohling, R.: Differential evolution algorithm on the GPU with C-CUDA. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 1–7 (2010)
14. Krömer, P., Snášel, V., Platoš, J.: Many-thread implementation of differential evolution for the CUDA platform. In: Proceedings of Genetic and Evolutionary Computation Conference, pp. 1595–1602 (2011)
15. Tagawa, K., Ishimizu, T.: Concurrent differential evolution based on MapReduce. Int. J. Comput. **4**(4), 161–168 (2010)
16. Syswerda, G.: A study of reproduction in generational and steady-state genetic algorithms. Foundations of Genetic Algorithms, vol. 2, pp. 94–101. Morgan Kaufmann, Los Altos (1991)
17. Feoktistov, V.: Differential Evolution in Search Solutions, chapter 6. Springer, New York (2006)
18. Tagawa, K.: A statistical study of the differential evolution based on continuous generation model. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 2614–2621 (2009)
19. Tagawa, K., Ishimizu, T.: A comparative study of distance dependent survival selection for sequential DE. In: Proceedings of IEEE International Conference on System, Man, and Cybernetics, pp. 3493–3500 (2010)
20. Davison, B.D., Rasheed, K.: Effect of global parallelism on a steady state GA. In: Proceedings of Genetic and Evolutionary Computation Conference Workshops, Evolutionary Computation and Parallel Processing Workshop, pp. 167–170 (1999)
21. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. In: Proceedings of 6th Symposium on Operating Systems Design and Implementation, pp. 137–149 (2010)
22. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures, 5th edn. CRC Press, Boca Raton (2011)
23. Dorronsoro, B., Bouvry, P.: Improving classical and decentralized differential evolution with new mutation operator and population topologies. IEEE Trans. Evolut. Comput. **15**(1), 67–98 (2011)

Erratum to:

Chapter 3
Models of Gene Regulation: Integrating Modern Knowledge into the Random Boolean Network Framework

Christian Darabos, Marco Giacobini, Jason H. Moore, and Marco Tomassini

Erratum to:
S. Cagnoni et al. (eds.), *Evolution, Complexity and Artificial Life*,
DOI 10.1007/978-3-642-37577-4_3, © Springer-Verlag Berlin Heidelberg 2013

The name of this chapter's author is wrongly updated as "Mario Giacobini"
The correct name is "Marco Giacobini".

The online version of the original chapter can be found under
DOI: DOI 10.1007/978-3-642-37577-4_3
