UNIVERSITEIT GENT

FACULTY OF SCIENCES

Using a Novel Bio-inspired Robotic Model to Study Artificial Evolution

Yao Yao

Promoter: Prof.Dr. Yves Van de Peer

Ghent University (UGent)/Flanders Institute for Biotechnology (VIB)

Faculty of Sciences

UGent Department of Plant Biotechnology and Bioinformatics

VIB Department of Plant Systems Biology

Bioinformatics & Evolutionary Genomics

This work was supported by European Commission within the Work Program "Future and Emergent Technologies Proactive" under the grant agreement no. 216342 (Symbrion) and Ghent University (Multidisciplinary Research Partnership 'Bioinformatics: from nucleotides to networks'.

Dissertation submitted in fulfillment of the requirement of the degree: Doctor in Sciences, Bioinformatics.

Academic year: 2014-2015

Yao,Y (2015) “Using a Novel Bio-inspired Robotic Model to Study Artificial Evolution”. PhD Thesis, Ghent University, Ghent, Belgium

The author and promoter give the authorization to consult and copy parts of this work for personal use only. Every other use is subject to the copyright laws. Permission to reproduce any material contained in this work should be obtained from the author.

Examination committee

Prof. Dr. Wout Boerjan (chair)

Faculty of Sciences, Department of Plant Biotechnology and Bioinformatics, Ghent University

Prof. Dr. Yves Van de Peer (promoter)

Faculty of Sciences, Department of Plant Biotechnology and Bioinformatics, Ghent University

Prof. Dr. Kathleen Marchal*

Faculty of Sciences, Department of Plant Biotechnology and Bioinformatics, Ghent University

Prof. Dr. Tom Dhaene*

Faculty of Engineering, Department of Information Technology, Ghent University

Prof. Dr. Steven Maere

Faculty of Sciences, Department of Plant Biotechnology and Bioinformatics, Ghent University

Dr. Evert Haaswijk*

Faculty of Sciences, Department of Computer Sciences, Vrije Universiteit Amsterdam

Dr. Guy Baele*

Faculty of Medicine, Department of Microbiology and Immunology, University of Leuven

*Member of the reading commission

Table of Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I would like to thank my promoter, Prof. Yves Van de Peer for his continuous support during these past six years. As a non-native English speaker, it was not always easy for me to carry out a PhD in English. Furthermore, starting off in the new interdisciplinary field of artificial life and evolutionary computation has additionally complicated things because there were many novel paradigms and terminologies that needed to be considered during my research. Therefore, it was quite a struggle for me at first but I have been fortunate to have such motivated and enthusiastic supervisor to guide me in my research. Your guidance with a great patience in both academic writing and bioinformatics aspect has been stimulating, educational and productive. Furthermore, I am also very grateful for your insightful comments, open attitude in the discussions, and immense knowledge in evolutionary biology that, taken together, made you a great mentor.

I would also like to thank Prof. Dr. Kathleen Marchal, my colleagues and former colleagues, both in the department of plant systems biology and in the Symbrion project, for their help in getting me started six years ago, for their helpful discussions and talks concerning my work and for the friendly atmosphere and support. I would like to extend special thanks to the people of VIB's administrative support team and IT support team - Sophie, Diane, Frederik, Dany – for their all support and help during my PhD study.

I would also like to thank my family and friends for the many good times during these past years. A special thank also to my wife Yi and my parents who have supported me all these years, in everything that I did.

Summary

Ever since Charles Darwin published “On the Origin of Species” in 1859, evolution as the essential concept for explaining the emergence of diversity and adaptability of life forms has been widely accepted and thoroughly investigated by the scientific community. At the same time, researchers from different fields have continuously expanded the connotations of evolution. For instance, evolutionary theory already has been separately introduced into the studies of genetics, artificial intelligence, economics, and psychology, to help explain certain phenomena or improving mechanisms of practical application [1-5]. These efforts certainly provided many novel and brilliant achievements in Science while at the same time they made the study of evolution becoming more and more interdisciplinary [6]. Today, people realize that evolutionary principles can actually be applied to various systems and these systems may not only include biological organisms but also market agents [7], cultural elements [2], or even computational programs [4]. Even in the physical world, some cutting edge research is exploring the possibility of connecting evolution with the concrete physical embodiment (i.e. [8]). Inspired by such ideas, one of the initial motivations of the research presented here is that it utilizes the principles of evolution in biological organisms to improve the adaptability of (simulated) robots (also referred as computational agents [9]) under unpredictable and changing environments. Furthermore, during this research, we realized that the self-organized collective behavior demonstrated by the swarm bio-inspired (simulated) robots could also spontaneously emerge at different levels and that this is also an inherent feature of all evolutionary processes in nature. The origin of emerging self-organized behavioral patterns at different levels has a great effect on evolution and studying this is helpful for providing a more comprehensive understanding of the evolutionary processes in nature.

The artificial genome and the bio-inspired model of gene regulatory networks

In terms of biological evolution, it is the genes on the genome that eventually carry all evolutionary information. Moreover, genes usually do not work in isolation, but form intricate gene regulatory networks (GRN). For example, interactions between genes have important impact on many aspects of evolution such as gene expression regulation, phenotypic plasticity, epigenetic inheritance, and genetic mutations. One view, originally developed by the works of W. D. Hamilton [10,11], Colin Pittendrigh [12] and George C. Williams [13] and later popularized by Richard Dawkins in his books ‘The Selfish Gene’ [14] and ‘The Extended Phenotype’ [15] considers the gene as the primary unit of selection. However, such gene-centered view of evolution also has been criticized as excessively "reductionist” by, for instance, Stephen Jay Gould [16]. Gould views selection as acting at many levels, and has called attention to a hierarchical perspective of selection. In fact, in this thesis, we try to combine these different kinds of views into one holistic view. On one side, we affirm the role of a gene as one of the evolutionary objects but on the other side, we agree that the gene is not the only evolutionary entity important for evolution. For instance, the organism is not simply a “vehicle” for genes but, in turn, acts also as an evolutionary object itself at a larger scale. In our model, evolution thus acts at multiple levels (see further). Back to the genetic level of evolution, due to the important

roles of genes and genomes, developing the appropriate artificial genome and gene regulatory network model is a critical and challenging task for mimicking the true biological genetic evolutionary system. Reading the current literature on hitherto developed models for artificial genes and genomes, it has become clear that this field still lacks a representation model that can represent genes as individual units of selection rather than passive genetic information recorders. Moreover, the interaction between the adaptation of a particular gene and the whole evolutionary process is ignored by many evolutionary models. Several principles and methods for improving previous gene and genome models are explored in this work. Furthermore, based on these principles, a novel artificial genome and a corresponding GRN framework have been developed. Through implementation of the principles of biological GRNs, our framework allows individual genes as independent evolutionary objects to interact with each other and to adapt to a changing environment. We believe that our novel model offers a more realistic and holistic view on gene evolution by including the dynamic interaction between genes and the phenotype of the organisms.

Using agent based simulation and swarm robots to study evolutionary systems as complex adaptive systems

The essential driving force of biological evolution is evolving the necessary adaptive traits in a(n) (complex) environment. Therefore, one of the primary reasons of mimicking biological evolution in computational programs is to obtain similar adaptability in digital organisms as in biological organisms by using (some of) the same principles of evolution. In general, the way adaptive phenotypes in biology evolve is completely different from the hitherto applied engineering approaches. Actually, the strong self-adaptability of biological organisms is based on a dynamic structure of interactions [17], which has been referred to as complex adaptive system (CAS) in complexity theory. In contrast to the traditional engineering system framework, CAS has a distributed structure and could be regarded as an aggregate of many independent interacting entities. The adaptability of CASs emerges from the collective behavior of these inner entities. Through the interaction of all inner entities and the environmental inputs, the collective behavioral pattern is continuously self-regulating to adapt to the current environment. Our approach described in this thesis aims to develop such CAS framework in computational systems and to develop an artificial system that also possesses the same self-adaptability capability as its biological counterpart. To achieve this goal, we adopted the principle of computational agents (see Introduction & Chapter 2) to represent the corresponding inner entities of a biological CAS and simulated the interaction of these inner entities in silico. Moreover, we also mimic the ‘nested’ structure of CASs. In nature, CASs rarely acts at one single level because, if well adapted, the CAS tends to reproduce many times in the environment. Since CASs can also interact with each other, a collection of CASs together in the common environment will, in turn, form a population, which could be seen as a CAS at a larger scale. Furthermore, the adaptation of the CAS not only means that the CAS needs to respond to environmental changes but, in turn, through its evolution and adaptation will also invoke some changes on the environment itself. As a result, other CASs need to change themselves (adapt) to adapt with the changed CAS. Such chain reaction will continue until all CASs reach a stable status. In terms of biological or gene evolution, for instance, we could consider the GRN in every individual organism as a CAS, but the collective behavioral patterns within the ecosystem actually forms another kind of CAS at a higher level. Due to the inseparable and nested relationships amongst CASs in nature and to better understand its consequences, we have tried to implement a similar nested structure in our simulations. First, we developed the computational CAS, which represents the gene regulatory model in biological organisms and then implemented this into each swarm robot as the controller. Multiple swarm robots in the simulation then constitute the collective behavioral pattern through their interaction. Such patterns are also evolving and adapt with the evolution of individual organisms. In turn, for each individual robot, the evolutionary processes playing at the population and/or ecosystem level will have, in turn, an effect on the adaptation of the individual organisms.

The adaptation of swarm robots under dynamically changing environments

In this study, we assumed no static environment and/or explicit fitness function for the robots in the simulation experiments. It is our conviction that, ideally, the fitness function or fitness evaluation can change as a result of the changing environment. Like a natural biological environment, in our simulation, the environment can change as a result of the interaction to adaptations and is potentially evolving on the shorter or longer term. In our simulations, all environmental conditions, as well as the adaptations of the robots, emerge based on the interaction between the swarm robots and between the swarm robots and the environment, so it is impossible to design a common explicit fitness function that applies to the behavior of all robots at all times. Instead of using an explicit fitness function, we chose to use the energy level of the robot as an estimation for the adaptation of the robots. The energy level of the robot is dynamically calculated based on the interactive behavior between the robot and its environment. For different time points and different robots, the energy level could be different but the survival and replication of all robots requires the energy to reach a certain level. Even with using a dynamic energy level as an indicator, identifying the particular adaptive behavior of each single robot in such changing environment is difficult because the adaptation only based on the energy level of the single robot ignores the environmental context. For example, increasing energy is easier in some environments than others for a certain robot. In our approach, we measure the adaptation of a group (swarm) of (simulated) robots instead of the individual robots and ensure that the initial environments are, on average, similar for different groups of robots (each group of robots start from independent simulation experiments in similar conditions). To examine the adaptability of the bio-inspired system and the functionalities of an artificial genome, we compared the different groups of robots in the same artificial life simulation scenario. The results show that, on average, our bio-inspired framework could improve the adaptability of robots more efficiently while the adaptive behavioral pattern of the bio-inspired systems also leads to higher diversity than more simple ANN based systems.

Application to evolutionary robotics and artificial intelligence

The application of the self-adaptive systems' framework to evolutionary robotics and artificial intelligence is one of the interesting and critical tasks of the work presented here. Three concrete applications that have been investigated and discussed in more detail in this work are collision avoidance, self-organizing robotic organism aggregation, and artificial life. The results of the corresponding experiments, which are based on robotic simulations, show that our system's framework carries the potential to self-adaption under various dynamically changing environments.

A novel bottom up holistic perspective on evolution

In our simulation experiments, we have simulated the process of self-organization and the emergence of certain behavioral patterns in populations and tried to connect the emergence of these patterns with particular genetic evolutionary processes. The results suggest that the emergence of collective patterns in a population has an effect on the local environments of individual organisms and in turn, the changed local environment can effect the evolutionary process in the individual organism. Therefore, evolution actually results from various evolutionary processes acting at multiple levels. In addition, through comparison with a more traditional evolutionary model, we show that our multiple level evolutionary model helps the simulation to produce more realistic outcomes. Using such a holistic perspective on evolution can provide us with a more comprehensive understanding of how evolutionary processes might act at different levels.

Glossary of terms and definitions used

In the interdisciplinary research described in this thesis, we have used a somewhat novel approach to look at and simulate artificial evolution. Due to the intersection of multiple disciplines, we here want to introduce some terms and concepts to describe our bio-inspired model for simulating artificial evolution that might be new or unfamiliar. In this section, we list and discuss those terms that are used throughout the thesis in a little more detail.

Evolutionary process/evolutionary context:

In our approach, evolution is considered a comprehensive process composed of many smaller-scale processes (see Fig 1). We refer to these as embedded evolutionary processes acting at different levels because all of them share the same evolutionary principles. In other words, they can be regarded as the miniatures or epitomes of overall evolution. For example, each gene evolves like an independent entity. Different genes interact with each other and they can all be under different selection pressures and have their own functional roles in Gene Regulatory Networks (GRNs), just like different organisms can have their own niche in an ecosystem. At a larger scale, this view is reminiscent of the Gaia hypothesis proposed by James Lovelock [18], where the whole biosphere is seen as some sort of super organism. We thus adopt the view that evolution forms a nested architecture, similar to the well-known Matryoshka dolls, however, of course much more complex. Every level at which evolution can act (a genome, an organism, a population, …) can include several independent evolutionary processes at the same time. Although one might argue about the differences in detail, such as the different rules of interaction between different evolutionary processes or the various features of different evolutionary components, we still can observe certain commonalities such as selection, variation, competition among these components and they all share the similar bottom-up self-organization processes. In addition, the rules of interaction at lower levels also apply at the rules of interaction at higher levels. Stripped from the different evolutionary contexts (context here refers to all historical changes that are relevant with the interaction in evolution), all these evolutionary processes can be described by a common model (complex adaptive system [19]). In conclusion, in our research an evolutionary process does not only refer to the overarching process of evolution but also to the subparts of evolution that can be considered different cases of the discussed common model. For instance, a sub-evolutionary process can be the evolutionary process acting on the gene, the genome, a population, or the whole ecosystem. In chapter 6, I will elaborate more on the reason why we (need to) distinguish between different evolutionary (sub)processes.

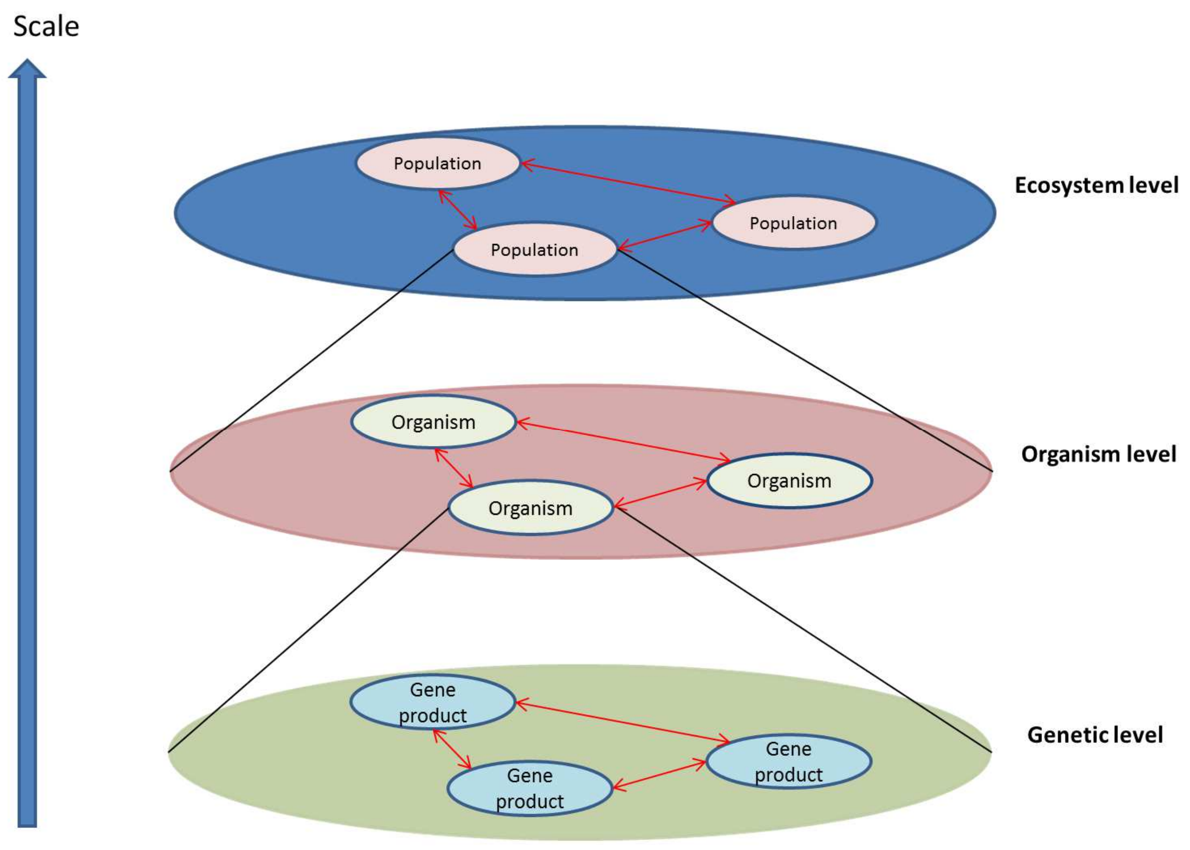


Figure 1 Multiple level evolutionary processes in our interaction-based evolutionary model.

Evolutionary components:

In this work, an evolutionary component refers to any functional entity (i.e. genes, proteins, organisms, and populations) that can interact with other functional entities (see Fig 1). Components are thus entities on which evolution acts but they can also independently interact with other components and they can accumulate changes (evolve). Actually, evolutionary components can be regarded as interactive modules in evolution. For example, in Fig 1, an organism is a component and the organism can interact with other organisms. Other organisms are external to this organism and therefore they are referred to as external components to this organism in our model. At the same time, this organism will also ‘interact’ with its internal entities (i.e. sleep behavior could affect the gene expression pattern and then the expression pattern will react on the behavior of organism later on [20,21]). We refer to these kinds of components as internal components to this organism. Furthermore, the corresponding environments of these two kinds of components are referred to as external or internal environments in our model.

Interaction and interaction-based evolution:

In our evolutionary model, a particular evolutionary event is actually defined by the interaction of all relevant components. Here, interaction refers to any causality on changes between two components. Changing one component can induce changes on another component, and such events between two components are referred to as interactions. All evolutionary contexts or events can be described as the consequences of such interactions. However, it should be noted that: 1) two interacting components can be derived from evolutionary events acting at different levels, i.e. a gene might have an effect on the phenotype of an organism, so gene and organism are from different 'evolutionary levels'; 2) interaction rules, i.e. the connection between two changes, are also part of the evolutionary context and can dynamically change during evolution (i.e. if a gene has lost its gene regulatory function, the interaction rules between this gene and others have changed); 3) interactions can have domino effects at later stages (i.e. one gene might have an effect on a GRN and the changed GRN can affect the behavior of an organism).

In general, we assume that all evolutionary contexts (evolutionary changes and events) are based on dynamically changing interactions that occur continuously during evolution.

The interactions between 'multiple level' components will determine the evolutionary context in our artificial simulations. We refer to this holistic model of evolution as interaction-based evolution. The simulation framework that was developed to implement this model thus aims to provide a more comprehensive evolutionary context where interactions between multi-level components (genes, genomes, organisms, populations, and ecosystems) are considered. In the new simulation model, interaction between different evolutionary components can dynamically affect different evolutionary processes and the relevant evolutionary context, such as fitness, selection pressure, gene mutation and so on.

Environment/selection:

In our multiple level model (see Fig 1), the selection pressure for a particular individual comes from two sides. On the one side, an individual has to adapt to its external environment. For instance, an organism (see Fig 1) has to adapt to other organisms and to local environmental changes. On the other side, the organism also has its own internal components (genes and gene products) and these internal components consist of an internal system that we can refer to as the internal environment of the organism. Adaptation of each of these internal components does not have to be beneficial to the individual organism explicitly (i.e. the "cheaters" may be harmful to the whole population but cheating behavior brings advantages to individuals temporarily [22]) and therefore the organism also has a selection pressure to have a more favorable internal environment (i.e. more mutations and novel traits arising in the population could increase the evolvability for adapting to changes of the external environment but the population also needs some kind of regulation mechanisms to avoid or remove possible mutations that may lead the cheating behaviors). Based on this model, adaptation is actually a trade-off between the external and internal environments of the system. In our study, we integrated both the external and internal environments into a common simulated evolution and evolve any novel trait in the simulation based on the balance between the both external and internal environments.

Artificial immune system:

The field of Artificial Immune Systems (AIS) is concerned with 'translating' the structure and function of the immune system into computational systems, and investigating the application of these systems towards solving computational problems in mathematics, engineering, and information technology. AIS is a sub-field of Biologically-inspired computing and natural computation, with interests in Machine Learning and belonging to the broader field of Artificial Intelligence. Based on [23], "Artificial Immune Systems (AIS) are adaptive systems, inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving".

Neuroevolution:

Neuroevolution is a form of machine learning that uses evolutionary algorithms to train artificial neural network.

Artificial life:

It is a field of study which seeks to synthesize the characteristics of life by artificial means, particularly employing computer simulation.

The evolutionary process acts at multiple levels in our simulation:

In our work, we assume that evolution acts at (at least) at three different levels. This framework is of course still a simplified model compared to biological evolution and the 'three level' architecture is just to demonstrate our 'multiple level' artificial evolution (the number of levels is initially specified but not necessary). In future research, we could increase the three level architecture to more or fewer levels depending on the research interests. Each particular level in our framework has at least one kind of interactive evolutionary component and can produce the self-organized structure through the interaction of these components. Since novel self-organized evolutionary components can emerge during evolution, new levels can emerge as well. For example, swarm robots can form groups (through aggregation or close cooperation patterns) and these groups may also interact with each other as individual modules. When we observe the self-organized pattern emerging from the interaction of groups, we will interpret this as the creation of a new level. In our simulation, we initially defined three levels and we use artificial genes, agents and swarm robots to represent the different evolutionary components at each level.

Gene level:

1. Selection is based on nucleotide substitution, gene deletion or gene duplication. If the fitness of the organism improves due to a mutation in a gene, the mutation rate of the gene becomes lower and the duplication rate higher.
2. Variation is based on the mutation operation and random genome initialization.
3. Competition is based on the feedback of agents. The gene that has higher gene expression will create more agents and will receive more feedback. The feedback can increase the expression level of the gene (inducer) but also can decrease the expression level (repressor).

GRN level:

1. Selection is determined by the agent's concentration decay rate, the lifetime of the agent and gene expression. Every time step, the concentration of the agents decreases while the age of agents increases. When the concentration level is lower than a certain threshold or the lifetime is higher than the maximum lifetime, the agent will be removed. Gene expression (caused by particular gene binding) will increase the concentration level and decrease the age of the agent encoded by the GRN or create a new agent through the GRN. Gene expression can be activated by regulatory agents or signaling agents
2. Variation is based on the different environmental inputs and the different genes in the genome.
3. Competition is based on the concentration levels of the agents. The concentration levels of the agents are a combination of gene expression, environmental inputs, agent decay and agent interaction.

Robot level:

1. Selection is based on the energy cost, lifetime of the robot and robot replication.
2. Variation is based on the different genomes and different GRNs. The compositions of GRNs depend on the environmental context and the environmental conditions.
3. Competition is based on the energy level of robot.

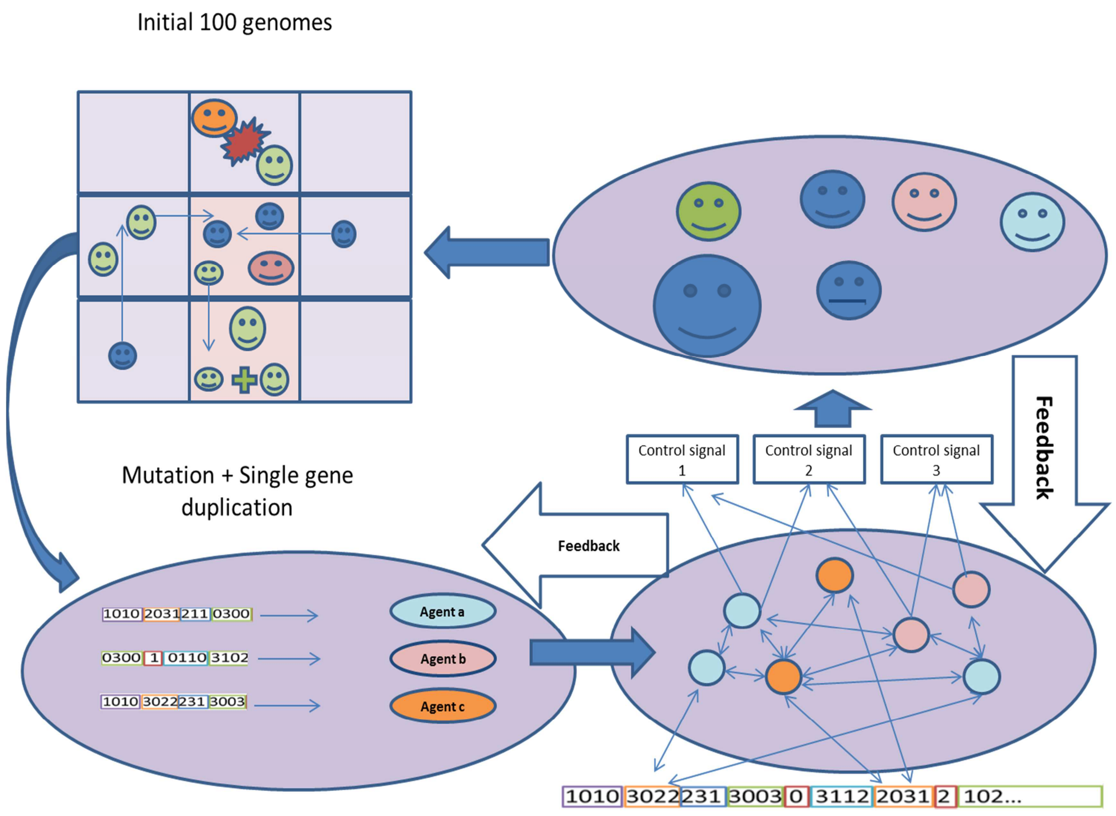


Figure 2 The evolutionary process acts at multiple levels. See text for details.

Equilibrium:

The condition of a system in which all competing influences are balanced. In our thesis, it refers to a variety based on the context. For example, it could means the balanced condition between food source growing and population extension or refers to the balanced focus on both food searching and robot prey behavior.

Chapter 1

Introduction

Author contribution

The content of this chapter was written by myself. It resulted from many fruitful discussions with both my promoters and all partners I had the chance to work with during my PhD studies.

The study of evolution has attracted great interest from biologists and, more recently, also computer scientists, albeit that the motivations of studying evolution might be different for the two research fields. In recent years, for biologists, one of the most interesting topics in evolution is the understanding of the evolutionary processes at the genome level. Understanding gene evolution helps the biologists to interpret the various phenomena and traits discovered in biological organisms. For instance, insights into the evolution of genes and genomes have helped explaining how species diversity has come about and how the tree of life could have been shaped[24-26]. On the other side, in terms of computer science, scientists have been fascinated with the inherent adaptability of natural evolutionary systems[27-29]. By mimicking the general mechanisms and principles of evolution, computer scientists aim to develop systems(i.e. [27,30,31]) that possess better adaptability in certain environments. Due to the different research interests in the two fields, people in each field have been separately investigating evolution for decades. However, through many evolutionary studies, more and more researchers realize that genetic evolutionary processes and adaptability are closely connected and evolutionary studies on both aspects require a close integration for extending our understanding of natural evolution. Moreover, scientists may have found biological evolution to be more complex than previously thought, because new evidence has shown for instance that biological evolution not only depends on the evolutionary processes at the genomic level but also is related with many other things such as the whole ecological environment [32], historical events [33], multiple level co-evolution processes [34] and so on. For example, researchers recently realized that, in nature, adaptability usually depends on the particular evolutionary context while the evolutionary process in turn can also be affected by particular adaptations [35,36]. For instance, Alon and colleagues [37] have stated that one trait of an organism could be adaptive in one environment but deleterious in another environment. In fact, such different environments usually can be encountered at different time stages so the adaptive value of a particular trait is based on the context. Moreover, regarding the niche and the collective behavioral patterns between organisms, these adaptations, in turn, can have an important impact on the evolutionary process through offering various selection patterns at different times [38]. As a result, evolution is based on the interaction of many aspects, like the genetic evolutionary process, ecology, the adaptation of the individual organisms, and the behavior of the whole population, rather than on a single independent process. To better understand the complexity and essential mechanisms of evolution, we need to integrate the knowledge about evolution from different aspects into a common model. Actually, today, we believe that a more comprehensive view on evolutionary research can promote studies on all related aspects of evolution. Based on such idea, this thesis will introduce a new approach for studying artificial evolution that combines gene evolution, evolution in individual phenotypes and evolution at the ecological level into one simulation model. For our research, we will use computational agent-based simulation to simulate gene evolution in virtual organisms with open-ended evolutionary models (see further). The evolutionary process in each organism is self-organized and such self-organization is based on the interaction of multiple agents. Such evolutionary process of the whole organism influences the gene regulation and the corresponding phenotype. Meanwhile, adaptation of organisms will influence the evolutionary process at a later stage as well. Furthermore, in our simulations, the organisms also interact with each other (as a population) and the interaction between organisms is recognized as part of the (changing) environment. With this approach, which is novel to the field of artificial evolution, we try to study evolutionary systems as dynamic developmental systems that include multiple interactive components. This way, both the evolutionary process and adaptation are integrated as two interactive parts into a common framework and the interaction among different components of biological evolution also are being seen as providing the essential context that that can have a significant influence on evolution.

The other sections of this chapter introduce the relevant background for this research. The first section briefly introduces some previous simulation work in evolutionary biology and discusses evolutionary models. The second section discusses some typical approaches in computer science for evolving adaptability of programs. In the third section, we introduce some recent interdisciplinary research in complexity theory, biology and computer science, which exemplifies the cross hybridization of evolutionary models from the different study domains.

1.1 Computational Evolutionary Biology

The study of evolution plays a critical role in biology because it reveals the diversity of life on Earth. Due to the importance of evolution ('Nothing in biology makes sense, except in the light of evolution' - Theodosius Dobzhansky [39]), there is a special sub-field of biology for investigating the evolutionary processes, which is referred to as evolutionary biology. To study the evolutionary process, the traditional approach in evolutionary biology usually is to compare phenotypes and genetic features among multiple samples from different time stages or species. Based on such comparisons, the evolutionary process and model can be inferred. The conclusions gained by such approach are convincing but the approach itself is usually limited by factors of time and sample sizes. For example, evolutionary processes in organisms usually develop over millions of years and are based on large population sizes. Using only a few samples to compare makes it difficult to infer the whole evolutionary process. In addition, some samples are difficult to retrieve or to identify at particular times. Such difficulties make that evolutionary studies usually depend on support from paleontology, molecular genetics, cell biology, ecology, and so on. Sometimes, experimental techniques and conditions critically influence the final conclusions and limit the applicability of results. For example, some have argued that the 'boring' experimental environment may bias the results and reduce variation [37]. Recently, computational simulations have started to provide new alternative solutions for solving problems in evolutionary biology that have been difficult to solve the more classical way.

During the last decade, more and more researchers in evolutionary biology have started to use computer simulations as a supplementary approach to study evolution. Computationally simulating the evolutionary process could extend the population size and time scale of the samples while reproducing the interactions and events during the evolutionary process. During simulation, these evolutionary trajectories can be easily recorded for further analyses. All these advantages make the use of computational simulation increasingly popular in modern evolutionary biology research [40,41]. Nowadays, most simulation experiments for evolutionary biology have three different foci.

The first focus is on genome evolution. Through applying artificial evolutionary operations and selection on an artificial sequence, one can replay evolutionary processes as observed in real genome sequences and store the evolutionary trajectories taken during the simulation. Antonio [42]divided such kinds of simulations into two subclasses according to the different platform types: those based on coalescent simulators [43,44] and those based on forward simulators [45,46]. Coalescent simulators are based on the coalescent theory, which is a retrospective model of population genetics. In contrast to forward simulators, coalescent simulators start the simulation from genome samples of the current population and trace back evolution until a single ancestral copy. Generally speaking, these kinds of simulations can both efficiently run a long evolutionary process over many generations but they often ignore the complexity of the mapping between the genotype and the phenotype and lack the interaction between organisms and the environment.

The second focus is on the dynamics of gene expression patterns. For compensating the lack of a sophisticated genotype-phenotype mapping and the influence of the environment, some researchers also pay specific attention to another kind of simulations that emphasize evolution at the gene regulation and gene expression level. Those new simulations can be recognized as the second kind of simulating approaches in evolutionary biology that we would like to discuss. Here, simulations are not only based on artificial genomes but also on the (encoded) gene regulatory network models. The representation of the Gene Regulatory Network (GRN) model can vary depending on the different research interests. As Hidde concluded [47], the GRN model representation could be based on many different formalisms like graphic [48], network (Bayesian [49], Boolean [50] or logical network [51]), differential equations [52] or predefined rules [53]. When considering gene regulation and expression, the simulations include the interaction between genes in evolution.

The third focus is on the integration between the Marco and Micro scale of evolution. Except for the simulations discussed above, many recent simulation experiments also tend to investigate the correlation between the gene evolution and the dynamics of populations. In this kind of studies, the interaction between individuals and their collective behavior also have been considered as part of evolution and this part could exert a great influence on the evolutionary processes. Based on such ideas, the evolutionary model also has been extended to whole populations and the interaction between individuals. Such simulators often adopt the cellular automaton [54] or artificial life approach [55] to simulate the dynamic interactions between individuals. By comparing the emergent collective behavioral patterns or the genetic features of different populations in the simulation, researchers can deduce the corresponding evolutionary process or identify the evolutionary trajectories of the particular phenotype. Some typical examples of these kinds of experiments can be found in [56,57], where scientists have shown that with a process of random mutations on the simulated organisms, natural selection could force the population to evolve some complex features. Natural selection in these simulations is based on the competition between simulated organisms rather than from a completely predefined model.

In terms of evolutionary simulations, each kind of simulation approach discussed above represents a certain perspective on evolution. However, natural Evolution is a complex combination of multiple aspects rather than a single one. Recently, more and more evolutionary studies attempt to synthesize the different kinds of simulating models described above into one integrated simulator. A typical example of such approach has been described by Dada and Mendes[58].In their paper, authors integrated evolutionary processes acting at multiple levels into one simulator and suggested that multi-scale methodologies are playing or will play important roles in multi-scale problems in systems biology. Such integration allows scientists to study the evolutionary processes with a more comprehensive view.

1.2 Evolutionary Computing and Bio-inspired Computing

In computer science, the use of evolutionary principles for automated problem solving started around 1950. There are several distinct pioneers of this field. Evolutionary programming was introduced by Lawrence J. Fogel [59], while John Henry Holland [60] called his method a genetic algorithm. In Germany, Ingo Rechenberg and Hans-Paul Schwefel introduced evolutionary strategies [61,62]. Later, these different terms and approaches were eventually unified as different representatives of one technology, called evolutionary computing. Simulation of artificial evolution using evolutionary algorithms and artificial life started with the work of Nils Aall Barricelli [63] in the 1960s, and was later extended by Alex Fraser, who published a series of papers on simulation of artificial selection [64].

Generally speaking, in contrast to biological evolutionary research, computer science is usually more interested in solving problems corresponding to concrete tasks rather than in unraveling the details of evolutionary processes. Therefore, in traditional evolutionary computing studies, the genome and the relevant genetic evolutionary models usually are much simplified while selection during the simulation is more or less predefined by a particular task description. However, in recent years these features have become more and more interwoven because scientists were hoping to assign more complicated tasks to artificial evolutionary programs.   Artificial intelligence (AI) systems are the main application of evolutionary computation. During the last century, general task requirements for AI systems have been relatively simple; for instance, performing a certain procedure based on a few possible inputs, which is a rather ordinary task paradigm for such systems. The main aim of artificial evolutionary experiments given such relatively simple tasks is to identify the best response procedure to the corresponding environmental conditions. In most cases, the environmental conditions are finite and identifiable to the developers. Through an appropriately simplified evolutionary algorithm, the program can effectively reach the best solution within a relatively short time as long as the environmental conditions are limited and in an identifiable search space. Therefore, early studies in evolutionary computation have usually focused on the design of the algorithm and the relevant parameters in the framework. The mechanisms of the evolutionary process are relatively simple compared to their biological counterparts. More recently however, with the further development of AI systems, scientists have started to assign more complex and sophisticated tasks to the AI systems like robot exploration in novel environments [65,66]. These new kinds of tasks often require implicit task description and self-adaptability within an unknown environmental context. Some typical cases are for instance Mars exploration, disaster rescue, and deep-sea exploration[67]. These scenarios all require the AI systems to accomplish some open-ended tasks under an unknown environment, independently. Without people's help and well pre-defined environmental knowledge, the systems have to 'learn' the environment during their adaptation process while the system also needs to evolve various mechanisms that could adapt with unexpected environmental changes. Such new changes on the task scenario of evolutionary computation require much more sophisticated evolutionary mechanisms; the traditional evolutionary framework discussed above becomes insufficient to cover these new requirements. Considering these new tasks of evolutionary computation, with a dynamic and greatly extended search space, it is very hard to reach a static and optimal solution by the current engineering algorithms.

To deal with these new challenges, a new field has been established that has been referred to as biologically inspired computing or bio-inspired computing[66,68]. The main aim of bio-inspired computing is to mimic the evolutionary processes of biological systems to improve the adaptability and evolvability of the computational program in complex changing environments. The motivation of using these kinds of approaches is based on how biological systems adapt to complex environments. In nature, through the evolutionary process, biological systems have been usually well-adapted to complex changing environments and are generally regarded much more adaptive than any kind of artificial system. Such self-adaptability in biological organisms is evolved in a changing environment but not determined by any specific algorithm. To achieve the same adaptability as biological systems, bio-inspired computational approaches not only have to adopt similar frameworks and evolutionary operators as biological organisms but also need to simulate the general evolutionary processes that formed the corresponding structures and the operations during evolution. Evolutionary processes in nature usually use a bottom-up, decentralized way to gradually evolve a self-organized system[69,70] rather than following an explicit predefined system paradigm. This is very different from traditional engineering algorithms. Some typical examples of bio-inspired computing approaches can be seen in the relevant research on neuro-evolution[71,72], swarm intelligence[73,74] and artificial embryogeny [75]. Although these approaches vary in the details of their implementation and are applied in different contexts, they all share the same way of simulating the corresponding natural evolutionary context and using the bottom-up self-organizing principle to construct the systems' structure. Neuro-evolution is based on the evolutionary model of the brain (evolution of the connections between nerve cells); swarm intelligence simulates the collective behavioral patterns evolving from the social interaction of the individuals in the population, while artificial embryogeny uses the developmental model of biological embryology, which is based on the evolution of cells and tissues. These novel approaches of bio-inspired computing give the artificial programs a certain self-adaptability in the experiment while at the same time sharing more principles with evolutionary biology.

1.3 Complex Adaptive Systems and Swarm Evolutionary Robotics

Nowadays, besides the evolutionary process itself, another common interest in biology and computer science is adaptation. The importance of adaptation in evolution has been addressed by the American biologist Niles Eldredge in his book 'Reinventing Darwin: the great evolutionary debate'[76], where he stated that "Adaptation is the heart and soul of evolution." However, adaptation still remains a hotly debated theme in evolutionary studies.

As is well known, the environmental condition is a crucial factor for adaptation and it eventually determines the features of organisms (the phenotype) in evolution through selection. In nature, the actual environment is very complex and dynamically changing all the time. However, the predefined fitness functions usually describe adaptation as a rather static process. To overcome the complexity of adaptation, many researchers tend to use a reductionist perspective to understand adaptation as the sum of many small and limited partitions. For example, in some evolutionary simulations, adaptation has been regarded as a collection of particular phenotypic features and the model of adaptation is represented as a network between all phenotypes and conditions. Such perspective tends to provide an accurately quantified network-based model to explain adaptation and it assumes that adaptation at the larger scale is just a collection of many sub-networks. However, others believe that adaptation is similar to many other complex phenomena in nature and follows a nonlinear style and such style is not supposed to be accurately represented by mathematical network-based models. This is because the interaction among molecules, genes and organisms in adaptation may produce a considerable domino effect through a chain reaction and such context-based domino effect could not be investigated separately part by part. For example, pleiotropy[77] may cause one gene to influence the expression of many other genes through a chain reaction process and such influence could be exaggerated on the phenotype in later stages. Such kinds of influences also may have an effect on the collective behavior of the organisms and eventually change the collective patterns of the ecological system. Because the collective patterns change, selection also can be altered. All these potential effects are hardly addressed if one only focuses on a particular aspect of the adaptation process. On the other hand, adaptation is a dynamic process. Through the collective interaction among different components (genes, organisms and the species), the adaptive status and relations among different components could be various based on the different context. Such facts make the static relations among finite conditions become insufficient to present adaptation over time. More details about the complexity of adaptation has been discussed in Wagner's recent book "Arrival of the fittest"[78]. In his book, Wagner emphasized that the adaptive innovation on phenotype is combinatorial and the origin of such innovation is depended on the interaction of many components. As the title shown, Wagner tried to explain the arrival of the fittest traits from a more holistic view in this book.

Due to the limitations of the reductionist approach, we have to approach adaptation by a more holistic view and investigate the effect of particular features in adaptation based on the dynamic interactions between all interacting components on which evolution can work. In terms of a holistic model of adaptation, the Abelian sandpile model[79] could be a good metaphor to explain it. For such dynamic system models, the effect of one single unit (units refer the basic interactive components in the system like the single grain of sand in a sand pile) could be too small to explain the self-organizing structure of the whole system but any such small effect also could totally change the structure and redefine the relation among all units later by a chain reaction. In evolution, every gene, organism and so on could be regarded as the single unit as well and these single units could also give a fundamental effect to whole evolution through a chain reaction. This phenomenon of such implicit effects has been referred to as the famous butterfly effect, which says that the movement of a butterfly wing can have effects thousands of miles further and these domino effects maybe enough to create a storm in the later time [80]. Adaptation is also a typical example of such dynamic system model but much more complex than the sand pile case. Here, a tiny change in the environment or in an individual organism, like a genetic mutation, a behavioral pattern switch, or even a small imperceptible weather change, could thoroughly change the evolutionary process and lead to unexpected effects on many other parts at later stages. This possibility makes any explicit local relationship based on the linear model of the adaptation become very vulnerable because such models lack the necessary context of evolution. Any missing information in the evolutionary process, even if it looks less important to us, could have a meaningful impact on the evolutionary process later. For example, let's assume that the concerned object in an evolutionary study is the correlation between a particular mutation and adaptability. As discussed previously, such correlation could be different under different environmental contexts and the context could be affected by many occasional events in history like famine, plague and so on. Although these occasional events may seem less relevant and it is not always possible to identify these during evolution, we could however not deny their domino effect in evolution (these domino effects may eventually change selection). Without knowing the environmental context, the observed correlation in simulation may be quite meaningless. In other words, the same mutation could variably affect adaptability depending on different context. Under such situation, we only can conclude that there are insufficient data to be able to precisely explain the phenomena that we observed. To more efficiently explain the complex phenomena that we discussed above, researchers have started to use a novel approach to investigate such kind of systems like the sand pile model or Lorenz weather model (see more examples in about Lorenz system in [81,82]) and referred to such systems as complex systems. Some of the complex systems like biological systems could self-adapt within their surrounding environment and these systems have been particularly referred to as complex adaptive systems (CAS) [83,84].

The modern scientific study of complex systems is relatively young in comparison to conventional fields of science but the development of the research field goes fast. In the early days of studies of complex systems, they have only been used to explain the nonlinearity and chaos in dynamic systems of physics and chemistry [85]. Later, people found that such systems also exist in many other domains and the study of CASs quickly extended to economics, artificial intelligence, biology and so on [86-88]. From the view of complex systems, adaptation is a kind of emerging equilibrium and it dynamically interacts with the environment during the evolutionary process[37,89]. In evolution, a particular environmental context determines adaptation while the adaptive behavior of all individuals, in turn, has an influence on the whole environment. All those interactions in evolution discussed above can be regarded as an entire complex system. The great advantage of this perspective is that it looks at evolution as a whole indispensable system and acknowledges the great complexity behind any phenomenon in such system. Deconstructing evolution could simplify such complexity but it will also make the eventual model become more limited because of lacking the necessary context[6,90]. Although studying the complex system is based on a holistic view, it also divides the system into many individual interactive components. The difference with other approaches though is that the new perspective does not emphasize the logic relations between these components nor does it tend to identify the sub-network based on the relations among components. Instead of studying the individual relationships, the complex system only defines the basic and common rules of interaction between all components and then simulates the interaction among the all components. By comparing the observed evolutionary patterns in one simulation with the results of other simulations, researchers can identify the common patterns of the collective behavior among these components. With the new approach, the behavior of the system emerges from the interaction during the simulation and the whole system is not divided into different sub networks that are studied separately. In fact, this holistic approach no longer wants to establish a precise model for representing the complex system before the simulation because the enormous complexity is too difficult to be predefined. On the other hand, it simulates the internal interactions of the system whose basic rules can be identified after which the program then creates the self-organizing complex system model through the simulated interaction. This way is not very good for presenting or predicting the concrete details of the system but it does provide a more convincing description about the systems' behavior in general because it uses a similar environmental and evolutionary context during the simulation. As we discussed above, complex system models literarily exist in many natural systems and it is very easy to integrate the knowledge from many different subjects into one common model because it’s inherent to the holistic view. This also renders the study of the complex system a real interdisciplinary field that allows investigating the common intersection of different study domains. Nowadays, evolutionary studies in both biology and computer science have shown a strong tendency to adopt the complex system concept and relevant examples can be seen in[91-95]. Furthermore, researchers from both different domains often cooperate to study evolution based on a common holistic perspective. In artificial life simulations, we can see many typical experiments based on such cooperation. By utilizing bio-inspired agents, researchers in artificial life mimic biological complex systems and study the emergent properties of societies of agents[96]. Platforms and approaches like Avida, EcoSim, Creatures Tierra, and others have been widely used in many studies in evolutionary biology and artificial intelligence recently[97-99]. Furthermore, with the aid of evolutionary swarm robotics, we can also use real robots instead of simulated agents to do similar experiments in the real world for practical scenarios (i.e. [100,101]).

To implement the holistic approach and the corresponding simulation, people need a distributed system framework and the system structure has to be flexible enough to allow bottom up emergence and dynamic interaction in the system. Agent based modeling and swarm robotic systems inherently fit these requirements and have therefore become the primary research methods for studying complex systems these days[96,102]. In both evolutionary biology and bio-inspired computing, more and more studies have adopted agent-based simulation and swarm robotics to study evolution. For example, Lenski [57] and Foster [12] have separately investigated the evolution of microorganisms by the simulation of swarm digit bacteria (agents). In the bio-inspired computing field, with using swarm robots or computational agents based simulation, Dorigo et al [33], Floreano et al [103] and al-Rifaie et al [74] also have done pioneering work in using the above-mentioned approach in evolutionary studies.

1.4 Artificial evolution based on computational simulation

Artificial evolution in this PhD thesis is interpreted and implemented as follows. The population of control systems is specified by an artificial genotype. This genotype evolves through various evolutionary approaches (i.e. genetic algorithm[104], Evolution strategy[61]). All individual (simulated) robots in the population follow Darwinian evolution in the sense that better adapted ones have a higher chance of survival and higher reproduction rates. Adaptability is based on the interaction of individual robots and the environment. All interaction follows certain interaction rules. A crucial question in designing the artificial evolution experiments is to what extent complexity, mimicking real biological evolutionary processes, can be implemented. Of course, completely simulating biological evolution is not feasible but we can at least try to adopt certain principles of biological evolution in the implementation of our artificial evolution. In our research, a main concern was to investigate interaction at different levels, i.e. among genes, GRNs and the environment and to test whether such interaction could help improving the adaptability of artificial control systems. In our artificial evolution experiments, artificial control systems have simplified genotypes, selection patterns and population context compared with real biological systems. Therefore, connecting to real, well studied models in biology should be done with caution. For biological studies, the framework that we discussed in this thesis is probably still insufficient to study concrete complex phenomena in evolutionary biology. However, due to the extensibility of agents and robots, it is always possible to add more sophisticated models or rules into the corresponding components in simulation (without reconstruct the whole framework). Nevertheless, this framework can simulate some general complex patterns (i.e 'prudent predator' [105,106]) in evolution which are based on the dynamic interaction of multiple levels.   we would hope that some of the observations made with our artificial 'digital life' system could be useful to help explaining certain biological observations. Furthermore, a second important application is to see whether we can use these principles to design better performing robots, which is interesting from an engineering point of view.

Based on the consideration above, we simplified the evolutionary mechanism in our experiment and using the interaction based on general interaction rules to determine the corresponding selection patterns and population size during the simulation rather than specifying a model. For example, the population size of robots in the simulation is based on the survive of every robots instead of a certain size and the selection of robot is based on the current energy level (we initialized 100 different genomes in the beginning of each simulation but only a very few genomes survive and develop their own population during the simulation). Under our simulation environment, the energy level of robot is not only depended on robot's phenotype but also highly influenced by other robot's response and many stochastic events (i.e. the distribution of food and robots). All these settings in this research tend to provide a simple example of general evolutionary scenario which is based on the interaction among genes, GRNs and environment. The principles in our scenario have universality in most of biological evolutionary processes although we did not adopt the particular parameters and models.

1.5 Holistic simulation framework and the interaction at multiple levels

In this research, we present a novel holistic simulation framework for artificial evolution. In previous evolutionary studies, scientists have shown that natural/biological evolution operates at multiple levels, from the genetic level to the organism level, to the population and even the ecological level [107-109] . Particular evolutionary processes usually act at a particular level after which they may sip through to other levels[110,111] . This implies that a process that is operational at a particular level can be affected by events that are operational at a different level, a different time or a different location. For example, an occasional change of an ecosystem long time ago could be responsible for the retention of a particular gene mutation that has become important much later [112] . All these effects from events that have occurred at other times or are regarded as a part of the evolutionary context. The evolutionary context actually determines all events in evolution (natural selection) and all of our research is focused on how to put the appropriate context into the common model for replaying the tape of evolution (life) computationally. Therefore, the next question is how to find the appropriate context for a particular evolutionary process. One way to retrieve the context is from previous data, like from experimental results. The disadvantage of this is that we usually do not have enough data to describe the context that we need in enough detail (more discussion about this further). Another way to retrieve the context is running a simulation and create the context from simulation. In most cases, we also do not have enough context data for running a comprehensive evolutionary simulation which means that we have to assume context from some previous experience (i.e. predefined fitness function, mutation rates, environmental changes and so on).

The caveat here is that small differences in context can accumulate and may cause bigger effects later on [113,114] . In addition, the processes that cause so-called catastrophes in evolution[115,116] are dynamic and too complex to model by any static model. Such processes usually involve many factors and variables, while the interaction between these factors and variables makes it very difficult to establish a realistic static model in simulation. In our new simulation framework, we try to utilize the idea of interaction (referring to any causality on changes) between evolutionary components (genes, organisms, populations, …) to improve the implementation of dynamic context in our artificial evolution experiments.

1.6 Chapter Overview

The chapters of this thesis summarize my research about modeling and simulation of evolution in the fields of biology and computer science. In this thesis, I have modeled and simulated biological evolutionary processes in a novel simulation platform and tried to identify the links between the principles of natural evolution and the adaptability of (artificial) organisms.

Through using a novel holistic simulating framework, descripted in chapters 2 and 3, I simulated the evolutionary processes at the genetic and population level simultaneously. While chapter 2 discusses the simulating models at the genetic level, chapter 3 introduces the simulation models at the population level. In addition, chapter 3 also briefly discusses the unique advantages of this new simulation framework in evolutionary studies.

Chapter 4 describes the improved adaptability of robotic organisms in our swarm robot simulation using bio-inspired principles of artificial evolution. These bio-inspired principles are inherent to natural evolution and they play an important role in enhancing the adaptability of organisms in a changing environment. Through some case studies, I illustrate the feasibility of using these principles in a computational system and show that such artificial evolution can improve the adaptability of artificial systems as well.

In chapter 5, I discuss the interactive behavior at multiple levels and the fact that it produces an effect in evolution and adaptation of individual organisms and populations. Based on the simulation experiments, I provide some proof for these claims and based on examples of complex adaptation in dynamic environments.

Finally, chapter 6 discusses the merging of the models for biological evolution, computer science and the complex system. The potential advantages and benefits of such merge are promising and exciting. By introducing bio-inspired principles and self-organizing models, computational systems can be based on more sophisticated artificial evolutionary processes and such processes can greatly enhance adaptability and evolvability in changing environments. On the other hand, by utilizing these bio-inspired computational systems, such as swarm evolutionary robots, researchers can simulate natural evolution in a novel bottom up way. In such simulations, after defining the basic interactive rules at a multi-level computational framework, the interaction among the swarm robots can self-organize the evolutionary process and the corresponding adaptive patterns. This means that selection is based on the interaction between robots and that no predefined evolutionary models are chosen in the simulation. The results of the simulations can also provide the relevant context and trajectories, which are not always available with other traditional simulation framework. Finally, some limitations and future work of this research are also discussed.

Chapter 2

Artificial Genome and GRN

Author contribution

The content of this chapter was written by myself. It resulted from many fruitful discussions with both my promoters and all partners I had the chance to work with during my PhD studies. Figures 2.1 and 2.2 are taken from the relevant papers (see references) and Figure 2.3 and 2.4 are from [117].

In this chapter, first a general introduction about the artificial genome and GRN in computational simulation is given. After the introduction, I will explain the development of the artificial genome and GRN in our simulation framework while briefly reviewing the relevant previous models in other studies. Furthermore, the details about our genome and GRN models will also be described in greater depth in the following sections. Finally, I will discuss the implementation of the artificial genome and GRN in our simulations.

2.1 Introduction

Genomes are the blueprints of all biological life. Every organism possesses a genome that contains the biological information needed to construct and maintain a living example of that organism. Most genomes, including those of all cellular life forms, are made of DNA (deoxyribonucleic acid) but a few viruses have RNA (ribonucleic acid) genomes. DNA and RNA are polymeric molecules made up of linear, unbranched chains of nucleotides (each nucleotide exist of one of four possible bases: A, C, G or T/U). Nucleotides are linked to one another by phosphodiester bonds to form a DNA polymer, or polynucleotide, which might be several million nucleotides in length. The biological information contained in a genome is encoded in the nucleotide sequence of its DNA or RNA molecules and is divided into discrete units called genes. A gene is a segment of the genome that is transcribed into RNA. If the RNA is a transcript of a protein-coding gene then it is called a messenger RNA (mRNA) and is translated into protein. If the RNA is non-coding, such as ribosomal RNA (rRNA), then it is not translated. Almost all genes contain coding regions known as exons, which are expressed, with intervening sequences, known as introns, which are not expressed. In addition to exons and introns, each gene contains a closely adjacent upstream (5') regulatory promoter region and other regulatory sequences including enhancers, silencers, and sometimes a locus control region. The promoter region contains specific sequences such as a TATA box, a CG box, and a CAAT box, which provide binding sites for transcription factors. The enhancer and silencer sequences fulfill a similar purpose, but are located at a greater distance from the coding sequences. The first and last exons contain untranslated regions, known as the 5' UTR and 3' UTR, respectively. The 5' UTR marks the start of transcription and contains an initiator codon that indicates the site of the start of translation. The 3' UTR contains a termination codon, which marks the end of translation [118]. The structure of a typical gene can be seen in Fig 2.1.

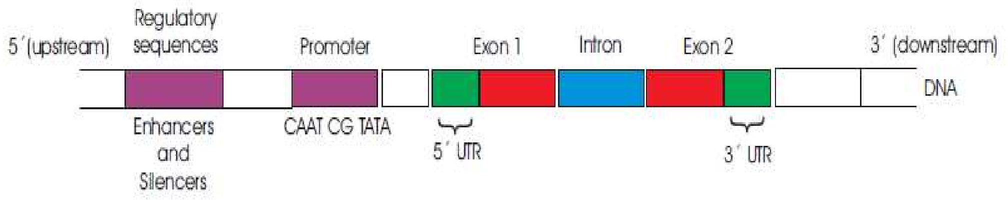


Figure 2.1 Graphical representation of the structure of a gene, after Young (*Cited from[118]*)

In biological organisms, the phenotype is, amongst other things, determined by gene regulation that is a process where a cell determines which genes it will express where and when. A gene regulatory network (GRN) depicts a collection of DNA segments in a cell that interact with each other (indirectly through their RNA and protein expression products) and with other substances in the cell, thereby governing the rates at which genes in the network are transcribed into messenger RNA (mRNA). In general, each mRNA molecule goes on to make a specific protein (or set of proteins). These proteins ultimately determine the phenotype. In fact, gene regulation is a general term for molecular processes that conduct the cellular control of the functional product of a gene, which may be an RNA or a protein. In single-celled organisms, regulatory networks respond to the external environment, optimizing the cell at a given time for survival in that particular environment. In multicellular animals the same principle has been put in the service of gene cascades that control body-shape. This fact means that the adaptability of the organisms in nature is mainly dependent on the corresponding GRN.

Due to the significant importance of the genome and GRN in evolution, researchers have paid considerable attention to develop suitable artificial genomes and GRNs in evolutionary simulations. The motivation for using artificial genomes and GRNs in simulations is twofold. On the one hand, artificial genomes and GRNs could help to simulate the corresponding genetic evolutionary process more realistically [119]. On the other hand, by using a bio-inspired genome and GRN, their inherent essential principles could help computational programs like robot controllers to adapt more efficiently to certain environments [103]. For these reasons, many versions of the artificial genome and GRN have been developed in previous research. In general, these artificial genomes can be divided into two classes, namely direct encoding and indirect encoding[120]. Direct encoding specifies all traits of the phenotype in the genome and the genes on the genome will directly determine the corresponding phenotypic traits. In contrast, indirect encoding only encodes the interactive rules between genes and the environment. Through the simulated gene regulation process, the phenotype will be self-organized during the simulation. Indirect encoding usually can hold more information (no specified details of phenotypic trait) and tends to be more flexible (allowing dynamic changes) than direct encoding[71,75]. On the other hand, direct encoding can efficiently simplify the complexity of the simulation and thereby reduces the chance of unpredictable behavior during evolution[72].

2.2 Artificial genome

In our work, we choose the indirect encoding scheme. The genome structure itself has been inspired by the model previously proposed by Reil [121] which consists of a randomly created string of digits (similar to DNA sequences which are essentially a string of nucleotides). Genes are not pre-specified, but identified in the randomly built genome as any occurrence of the sequence ’1010’, simulating the concept of a gene/transcript start site, followed by N digits that represent the actual coding gene). The region where no genes are present, i.e. the region between genes, is denoted as intergenic region.

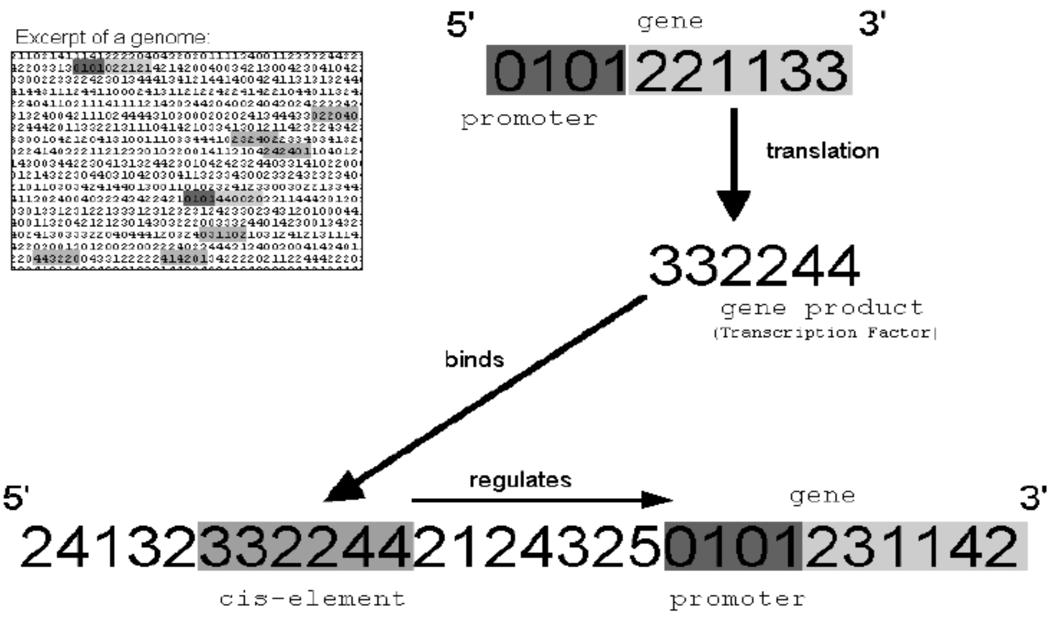


Figure 2.2 Pattern of gene expression and regulation in the artificial genome, as proposed by Reil (Cited from[121])

Compared to the initial model of Reil [121], our model (Fig. 2.3) was modified as follows: an explicit distinction is made between signaling, regulatory and structural genes. A gene indicator (one digit) indicates the gene type where 0 denotes that the respective gene has a regulatory function, 1 denotes a structural gene and 2 denotes a signaling gene. Any gene, irrespective of its type, consists of a transcript start site (1010), a sequence specifying gene length (which is intended to avoid overlap between neighboring genes in the genome) and three other regions that are a binding site region (4 digits), an expression level region (7 digits), which in turn consists of a 'default' and a 'gene-specific' expression region, and a gene content region. Structural, regulatory and signaling genes further differ in their gene content region. The content region of the regulatory genes has three parts: a target recognition site, which defines, in combination with the binding sites, which genes will be recognized and regulated by the regulatory gene, a region defining the regulatory type (being an activator (value 1) or a repressor (value 2) (Fig. 2.3) and an intensity region that defines the extent to which the regulator will activate/repress its targets genes. The content region of structural genes defines which actuators the structural gene will influence (for each structural gene this is predefined by means of a number that ranges from 0 to 7 defining the outcome) and also determines the extent to which the gene will influence the actuators (the output parameter). For signaling genes, the content region encodes an ANN structure that receives and integrates signals sensed by the robot (see Agent-based representation of the activated GRN). In the current study, the total genome size consists of 10 chromosomes of 10,000 characters.

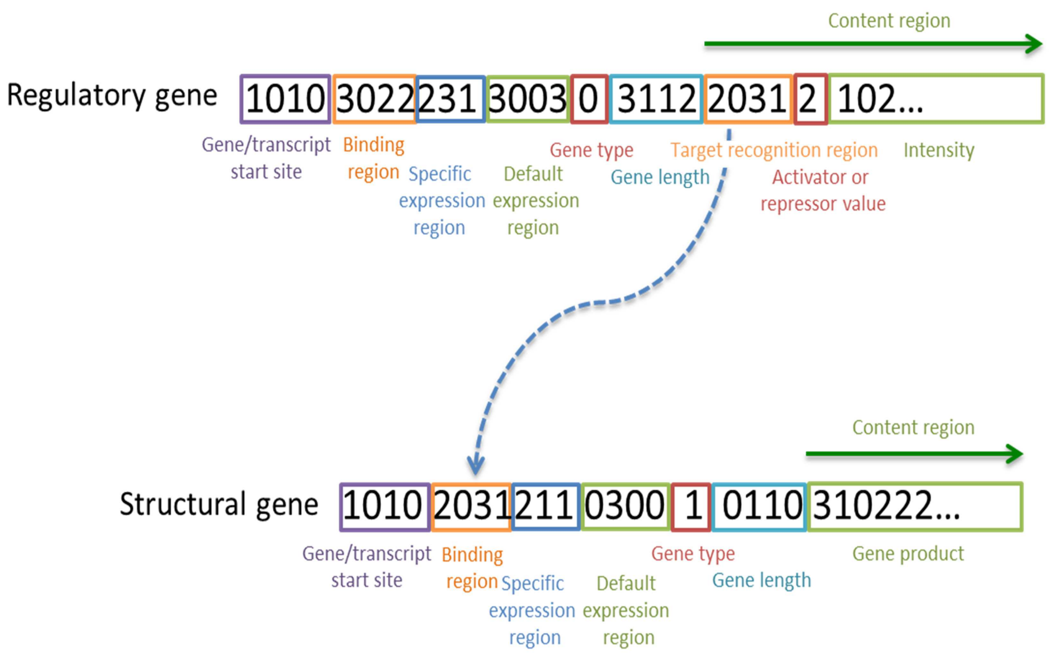


Figure 2.3 The artificial genome and gene regulatory model as used in the current study (Cited from[117])

The genome structure has been inspired by the model previously proposed by Reil [121] and consists of a randomly created string of digits. Genes are not pre-specified, but identified in the randomly built genome as any occurrence of the sequence ’1010’, simulating the concept of a gene/transcript start site, followed by N digits that represent the actual coding gene). Compared to the initial model of Reil, our model was modified as follows: an explicit distinction is made between regulatory and structural genes. A gene indicator (one digit) indicates the gene type where 0 denotes that the respective gene has a regulatory function and 1 denotes a structural gene. All genes consists of a transcript start site (1010), a sequence specifying gene length (which is intended to avoid overlap between neighboring genes in the genome) and three other regions that are a binding site region (4 digits), an expression level region (7 digits), which in turn consists of a ‘default’ and a ‘gene-specific’ expression region, and a gene content region. The content region of the regulatory genes has three parts: a target recognition site, which defines, in combination with the binding sites, which genes will be recognized and regulated by the regulatory gene, a region defining the regulatory type (being an activator (value 1) or a repressor (value 2)) and an intensity region that defines the extent to which the regulator will activate/repress its targets genes. The content region of structural genes defines which actuators the structural gene will influence and also determines the extent to which the gene will influence the actuators. The total genome size consists of 10 chromosomes of 10,000 characters.

2.2.1 Mutational events acting at the level of the artificial genome

As evolutionary forces (changing the genome), we implemented both substitutions and duplications. Regarding substitutions, the implementation is as follows:

In general, the intergenic part of the genome has a higher mutation rate than the 'coding' part. The mutation rates are gene specific and are dynamically determined by the fitness of the system: non-functional sequences and genes that have not yet contributed to the individual's fitness have a default mutation rate Nm ($3*10^{-5}$). However, a gene with a lower contribution to the fitness function will be assigned a higher mutation rate, whereas a gene with a higher contribution has a lower mutation rate. For each gene, the current mutation rate (Gm) is dependent on both the default non-coding sequence mutation rate (Nm) and the genes' adaptability value G1, as shown by the following equation:

$$Gm = (1 - \frac{G1}{Gmax}) * Nm$$

Where Gmax represents the maximum adaptability value (Constant).

These gene specific evolution models thus mimic the long-term effect of natural evolution in which genes that are under selection pressure tend to be maintained more than genes that are not.

Gene duplication is implemented as follows:

There is a software module that, at every time step, will check the expression of all genes and copies the 10 genes with the highest adaptability values. When a gene is on the list of genes with highest adaptability for more than 10 time steps, it will be regarded as a gene that qualifies to be duplicated. Like with mutations, the system also has a common background rate for duplications. Every time step, when the system searches for target genes, it also checks whether there is a gap or intergenic region between genes on the genome. If there is a gap (the minimum length being 100 bases), the program will check if there are any qualified genes with a length smaller than the gap and select this one for duplication. When all conditions have been satisfied, the candidate gene will be duplicated into the gap and removed from the list of candidates to be duplicated (the gene might get back on the list as long as it keeps its high adaptability value). When the environmental pressure increases, most genes will receive a negative feedback and as a result, the adaptability value of their corresponding agents will decrease. As discussed previously, this will result in an increased mutation rate, which may cause destruction of the promoter region, and consequently the gene downstream of it. Fewer genes on the genome will lead to more gaps and therefore higher duplication rates. More duplicates finally will introduce more variation to the genome, with the possibility of evolving novel GRNs.

2.2.2 Conversion of digital encoding of regions into corresponding values

Our artificial genome only includes digits but all sequence regions on the genome can be converted into numerical values. For all digital encodings, the following conversion into continuous numerical intervals is used: for a 'site' consisting of N digits (Equation 1):

(Digit at position 1)*$(4)^{(N-1)}$ + (Digit at position 2)*$(4)^{(N-2)}$ + …+ (Digit at position N)*$(4)^{0}$

So depending on N (the number of digits in a sequence site) the numerical ranges will differ. For example, the recognition between a binding site in a target gene and a TF recognition site is encoded as follows: both sites are converted into numerical values according to the equation described above. If the distance between the numerical value of the recognition site and the binding site is smaller than a predefined threshold, the gene is recognized by the TF, otherwise not. For example, the binding site 0120 is converted to 24 ($0*4^3+1*4^2+2*4+0=24$) whereas a recognition site of 0111 is converted to 21. The distance both sites equals 24-21=3. When the threshold is equal then or greater than 3, the binding between the TF and its target will be regarded as successful and the gene will be switched on (expressed). The different regions on the gene have their fixed length and their corresponding agent (see further in 2.3) could read all these sequence regions. Through the same decoding method discussed above, the agents could retrieve the numeric information from various regions on the gene. These numeric information could be used as the inputs or parameter values by agents.

2.3 Agent-based representation of the activated GRN

2.3.1 Agent based modeling

Agent-based modeling (ABM), also termed individual based modeling (IBM), is a relatively new approach to modeling systems comprised of autonomous, interacting agents. Computational advances have led to a growing number of agent-based applications in a variety of fields. For example, agent-based models have been used to simulate the electric power market designed to investigate market restructuring and deregulation and to understand implications of a competitive market on electricity prices, availability, and reliability[54]. Agent-based models have also been used to study biological tissue patterning events, which have implications for both physiological and pathological function, arise from a cascade of complex processes and rely on interactions between cells, genomic information, and intra-cellular signaling[122]. The benefits of agent-based modeling (ABM) over other modeling techniques are threefold[123]: (1) ABM captures emergent phenomena; (2) ABM provides a natural description of a system; and (3) ABM is flexible. Essential to ABM is its ability to capture emergent phenomena, which result from the interactions of individual entities. By definition, they cannot be reduced to the system’s parts: the whole is more than the sum of its parts because of the interactions between those parts. Emergent phenomena can have properties that are decoupled from the properties of the part. ABM is, by its very nature, the canonical approach to modeling emergent phenomena[124]: in ABM, one models and simulates the behavior of the system’s constituent units (the agents) and their interactions, capturing emergence from the bottom up when the simulation is run. Further, ABM provides a natural description of a system. Finally, the flexibility of ABM can be observed along multiple dimensions. For example, it is easy to add more agents to an agent-based model. ABM also provides a natural framework for tuning the complexity of the agents: behavior, degree of rationality, ability to learn and evolve, and rules of interactions [123].

Different modelers of agent-based systems have differing opinions on what constitutes an agent (see[125] for an overview). Macal and North [125]consider agents to have (amongst others) the following characteristics, which we adopt in our research: An agent is an identifiable, discrete individual with a set of characteristics and rules governing its behaviors and decision-making capability.

1. Agents have protocols for interaction with other agents, such as communication protocols, and the capability to respond to the environment.
2. An agent is flexible, and has the ability to learn and adapt its behaviors over time based on experience.

2.3.2 Gene regulation and gene expression

In terms of biology, gene regulation is a process in which a cell determines which genes it will express where and when. One of the easiest ways to illustrate gene regulation is to talk about gene regulation in humans. Every cell in the human body contains a complete copy of that person's DNA, with tens of thousands of potentially viable genes. Obviously, all of these genes cannot be expressed at once. Hence, cells must decide which genes to turn on and which genes to turn off. For example, a skin cell turns on the genes that make it a skin cell, while a bone cell would leave these genes turned off. Neither of these cells would need the genes that allow a cell to differentiate into a neuron, so these genes would be left of as well. In addition to being useful for cell differentiation, gene regulation is also valuable for cell function. As a cell moves through its life, it has different needs and functions, which can be addressed with the use of gene regulation to determine which genes are expressed and when. Likewise, cells can adapt to environmental changes such as an injury, which requires repair by activating new genes. For the cell, gene regulation can be accomplished in a number of different ways, with one of the most common simply being regulation of the rate at which RNA transcription occurs. Genes can also be deactivated by changing the structure of the DNA in an individual cell to turn them off or on. Unicellular organisms also utilize gene regulation to regulate their functions and activity. These organisms must be able to adapt genetic material quickly to adjust to changing circumstances and new environments. Failure to do so will cause not only death of the cell, but death of the organism itself. Gene regulation allows such organisms to do things which will allow them to fit into hostile and extreme environments and to adapt to changes such as the introduction of antibiotics into their environment.

In our simulation framework, we try to mimic the same gene regulation and expression principles of true biological genomes. The artificial genome is used to build the corresponding GRN system and the outputs of the GRN system eventually control the actual behavior the virtual organism. From the artificial genome to the GRN, gene expression plays a key process. In biological organisms, gene expression is the process by which information from a gene is used in the synthesis of a functional gene product. In our artificial genome, gene expression means that each gene on the genome can define a corresponding agent in the system based on its content information. When the individual gene is expressed (activated), its corresponding agent will be created in system as the gene product. These agents have their functions and could interact with each other. The whole GRN system is made up of these agents and the collective behavior of these agents determinate the eventual output of the GRN system. The biological gene expression underlies several control mechanisms in biological organism, whereas the regulation of the transcription machinery constitutes the most important gene-regulatory mechanism. Regulators of transcription are mainly proteins, called transcription factors (TFs). However, the overall gene regulation is much more complex and includes processes such as transcript degradation, translational control, and post-translational modification of proteins. Thereby, apart from proteins, also other molecules like RNAs and metabolites participate in a regulatory manner. Finally, the genes, regulators, and the regulatory connections between them form a gene regulatory network. In single-celled organisms, regulatory networks respond to the external environment, optimizing the cell at a given time for survival in this environment.

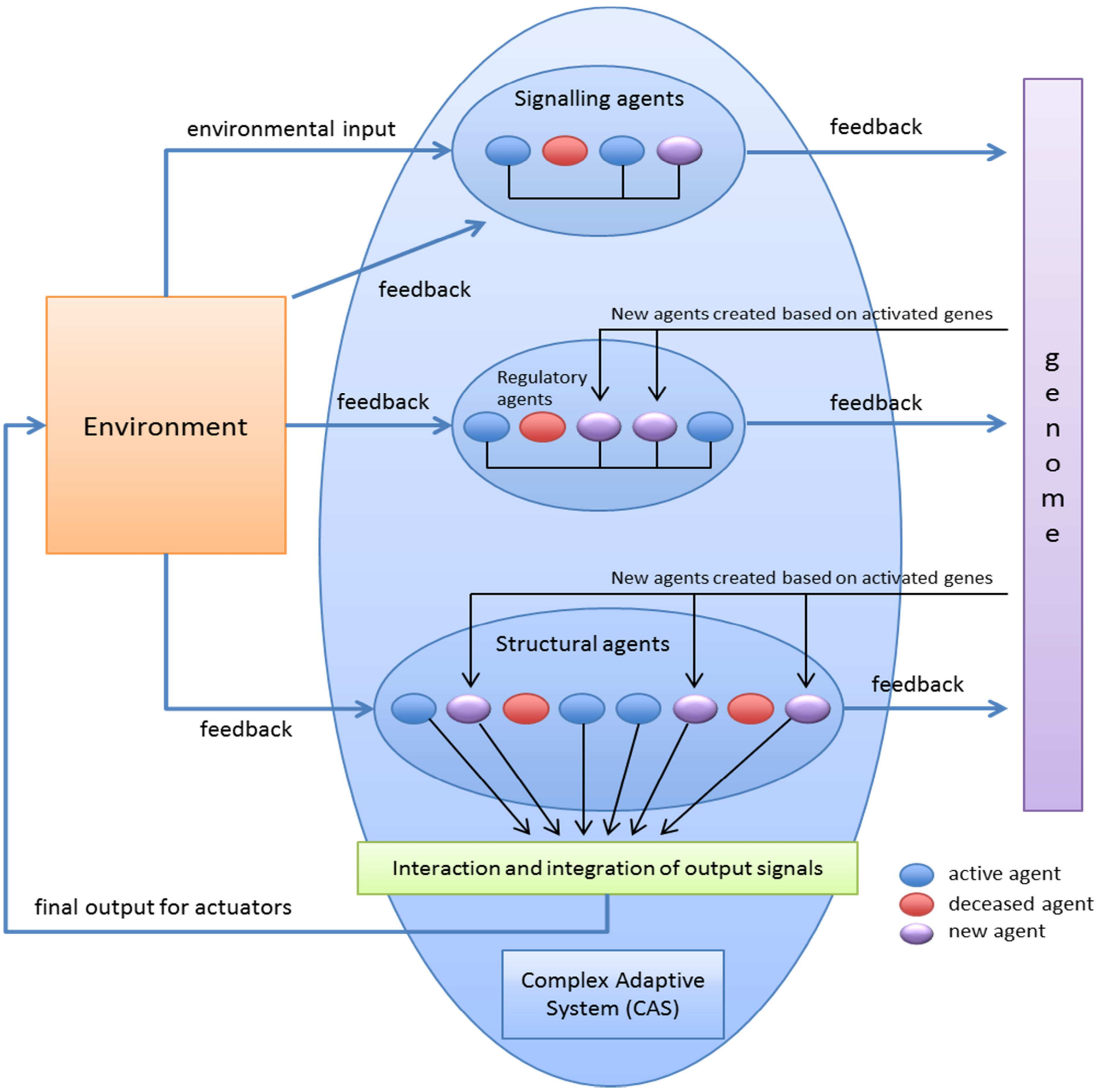
On the genome, some regulatory genes create the regulator agents rather than the TF agents (represent the RNA polymerase) when they are expressed. The regulator agents also check the target genes like the TF agents but change the expression level of the target gene instead of trigging the gene expression. Due to that, matches between the target recognition regions in these regulatory genes and the binding site regions in other target genes define which regulatory genes will recognize which targets and thus define the connections in the GRN (matches are defined as described above). If a regulatory gene recognizes the binding site in a target gene, the expression level of this target gene will be redefined. The degree to which a regulator agent enhances/represses its target genes is determined by the intensity region of regulatory genes (which is converted into an intensity value according to equation 1). The actual expression level of a target gene is determined by a default and a gene-specific expression level (gene expression level = default expression level * gene specific expression level) and the default expression level of a gene is derived from its default expression region using equation 1. The default expression region can only be altered through mutational events. The gene-specific expression level is a sum of a contribution of the degree to which the gene is activated/repressed by its regulators (derived from the intensity values of its respective regulators) and a value derived from its gene specific-expression region using equation 1.This latter region also allows for the accommodation of feedback from the gene's agent and changes dynamically according to the adaptability value of the gene's corresponding agent (see the details in the later section).

2.3.3 Different agents in the artificial GRN model

In our set up, the agent-based model is a representation of the condition-dependent instantiation of the GRN encoded by the AG (see Fig. 2.2 and Fig. 2.3). More specifically, three types of agents have been defined, each corresponding to a specific gene type. The signaling (also call transcription factor agents or TF agents) agents are corresponding to the signaling genes and these agents mimic the way biological systems integrate environmental stimuli. These agents can activate the particular encoded GRN based on the specific environmental inputs. Each signaling agent includes a uniquely embedded Artificial Neural Network (ANN), which reads the sensor input values and establish combinations of sensor values in the (simulated) robot. In other words, a single signaling agent does not correspond to a single sensor input, or a transformation thereof, but to a combination of different sensor inputs. The signaling agents channel the integrated sensor signals to the GRN by converting them into a ‘binding value’. This ‘binding value’ is used to activate other genes in the network (using the same principle that discussed at transcriptional regulation and gene expression section). There is a diagram in appendix C (fig S.3) shown a concrete example how one signaling agent works in the GRN framework.

Regulatory agents mediate signal transduction in the network by activating/repressing other regulatory or structural agents. In other words, every expressed regulatory gene invokes a regulatory agent. The regulatory agent identifies the target genes of its corresponding regulatory gene (determined by the match between the binding site region in the target and the recognition region in the regulator) and sends an instruction to its target genes to change their expression status (repress or enhance). The degree to which the expression of the target gene will change depends on the interpretation of the information encoded in its corresponding gene (expression region) and that of its regulator (intensity region). On the other hand, some regulatory agents directly activated the gene expression of other gene instead of giving the regulation.

A structural agent will translate the encoded information of a structural gene to an output parameter, which drives the actuator (e.g. wheel) of the robot. Each actuator usually receives many parameter values from different structural agents and will average these into one final value that will then be used as the control parameter (output value) for this particular actuator (see also Robot functionalities).



Signalling agents
environmental input
feedback
feedback
New agents created based on activated genes
Regulatory agents
feedback
feedback
Environment
g e n o m e
New agents created based on activated genes
Structural agents
feedback
feedback
Interaction and integration of output signals
final output for actuators
active agent
deceased agent
new agent
Complex Adaptive System (CAS)

Figure 2.4 Agent based system modeling the condition dependent instantiation of the GRN encoded by the artificial genome (Cited from [117])

The agents based level is also essential in establishing the feedback from the environment to the system through the agents adaptability values which affect both the agents lifetime, genomic encoding, and mutations rate.

2.3.4 Translation of the AG encoded GRN into the agent-based activated GRN

When a gene becomes expressed in the artificial genome, a corresponding agent will be created. Genes with a minimal expression level will be translated into agents with a probability that depends on the value of the gene-specific expression region. If a gene is switched on, the concentration of its corresponding agent depends on its total expression level. The concentration of an agent decays with time, mimicking protein degradation according to the following equation (2):

$C_i=(C_{i-1}*r*AV)/100$

where C_i represents the current concentration and C_{i-1} the concentration value at a previous time step, r the decay rate for the agent (set at 20% decay rate of the concentration for structural agents and 30% for regulatory agents), and AV represents the adaptability of the agent. Initially, agents take the value derived from the gene-specific expression region as the adaptability value (AV). However, the adaptability value will be altered depending on the current fitness value (see the adaptation on the individual gene section). During its lifetime, the agents' concentration and survival time will increase with an enhanced adaptability value. When the agent expires, its adaptability value will be used to alter the gene-specific expression region of the agent's corresponding gene (through the gene specific value using the reverse of equation 1). In general, the more the gene will be expressed, the higher the concentration of its corresponding agent and the higher the influence of the agent on the final output. If the concentration of the agent drops below a pre-set minimal level, the agent will be deleted. Signaling agents are treated differently and are initialized by activating 10 signaling genes. The system always ensures that during simulation always 10 signaling agents are active simultaneously by randomly activating a new signaling gene upon deletion of an expired signaling agent. To ensure signaling agents have a relatively long average lifetime, they are not decaying and every signaling agent will at least last for 50 time steps. They may be deleted after these 50 time steps when their AV drops below a predefined threshold.

All existing agents interact with each other to constitute a dynamic complex system in the simulation, which can be considered the active part of the regulatory network encoded by the genome. For each gene, it has its own corresponding agents and adaptation and thus selection can also vary for different genes on the genome.

2.3.5 Adaptation at the gene level

In our dynamic GRN, the agents produced by the expression of genes need to interact with each other and adapt to their own local environments. This design aims to simulate the formation of GRNs in the biological organisms. The developmental process of the biological GRNs is a self-organizing process and such process is based on the interaction of multiple genes and their products. During this process, each gene can have a different evolutionary context and can evolve different features or collective patterns with other genes. Here, we used a simplified model to represent such interaction and adaptation of the genes. In the simulation, we use the overall energy level of each simulated swarm robot (see more details in chapter 4) as a measure of its global fitness, which is used to define the feedback from the environment to the agents and via the agents to the genes. This feedback is defined through the 'adaptability value'. For each agent, the adaptability value (AV) is defined as a combination of the global fitness of the robot and additional values that express the dependence of the observed fitness on the specificities of a particular set of agents present in the robot at the time its fitness is evaluated. For instance, in our simulations, the adaptability value of a regulatory agent is determined by the global fitness (50%), by the overall average lifetime of the agents (assuming that an 'agent-set' with longer average lifetimes will have a greater long-lasting effect on the fitness) (30%), and by the number of agents active in the system, if this number ranges between 30 and 100 (20%). If the number of agents is smaller than 30, we judge the network too small to be viable. If the number of agents is larger than 100, we assume that it is hard to judge on the specificity of each of the agents. Consequently, in both cases we will decrease the contribution of the number of agents to the fitness.

The details of the relevant calculation can be found in the following equations:

The adaptability value on step i = V_i

Adaptability value at the previous time step = V_{i-1}

Energy level at step i = E_i

E_{i-1} the energy level of the robot at the previous time step;

Fitness value at step i = F_i

With Fi being equal to the average normalized change in energy of the robot between step i and i-1:

$F_i = \frac{E_i - E_{i-1}}{(E_i + E_{i-1})} * 100$

The number of agents at time step i = A_i

The average life time of all agents at time step i = L_i

And the average life time value at the last time step is L_{i-1}

With Lf being equal to the average normalized change in lifetime of the agents present in the robot between time step i and i-1:

$\mathrm{Lf} = \frac{L_i - L_{i-1}}{(L_i + L_{i-1})} * 100$

The concentration level (mimicking amount of protein product) of this agent = Ca

The adaptability value of an agent present in a robot at time step i (in case the number of agents ranges between 30 and 100):

$V_i = (F_i * 50\% + A_i * 20\% + Lf * 30\%) * \frac{Ca}{Cmax}$

With Ca = the concentration level (mimicking the amount of protein product) of this agent and Cmax

The final adaptability value at time step i: $\mathrm{AV} = \frac{V_{i-1} + V_i}{2}$

With V_i being equal to the adaptability value at time step i and V_{i-1} being equal to the adaptability value at the previous time step.

Assuming the adaptability value of the agent is AV, the feedback effect from that agent on the gene specific expression is at a time step i: Gene specific expression (i) = $10 * \frac{AV-20}{100}$ + Gene specific expression (i-1).

The Decay rate of the concentration level is connected with the AV of the agent and the higher AV gives the agent the lower decay rates. Eventually the AV of the agent will be regarded as the feedback its corresponding gene and gives the influence to change the gene specific expression level of that gene.

2.4 Comparison between our GRN framework and other bio-inspired computing frameworks

2.4.1 The features of our GRN framework

The design of this framework has a few new features compared with other bio-inspired computing frameworks.

First, we adopted a GRN based indirect encoding approach to construct the controller. As many recent indirect encoding Neuroevolution (see definition in Glossary) frameworks (i.e. hyperNeat, AGE-ANN), the framework in our research also uses the encoded genotype to produce the phenotype which refers to a corresponding control system. The main advantages of indirect encoding are ([126,127]) : 1) Allow recurring structures or features in the network to form modularity; 2) rearranging and specifying the search space based on different tasks.

Second, the main difference between our framework and many other indirect encoding network based frameworks is that our framework use genes to produce the individual agents instead of using genes to construct the particular network structure. This difference is reflected in the following aspects:

- An agent based GRN framework breaks down the whole network structure into granular pellets. By this, the explicit connections between network nodes are replaced by dynamic interactions of agents. The real interactions between agents are not only based on the interaction rules (evolved by genes) but also based on environmental conditions. For example, an agent interacting with another agent is frequency and concentration based. The interaction events sometimes depend on the competition among agents (i.e. one unit - based on the concentration - agent A can only interact with other agents once at one step. When the concentration of agent A is not sufficient to interact with all of other agents, agent like B or C may compete the opportunity to interact with that particular agent A). For network based frameworks, as long as the structure or genes do not change the connection will be fixed. In addition, an agent that changes may have a domino effect on others through interaction and such effects can be exaggerated through time.
- Network based frameworks can be formed based on the interaction of genes, but for agent based GRNs, the system is formed through the interaction between genes, agents and the environment. For instance, a change on the strength of an input signal will not alter the network structure but it can affect gene expression and consequently the composition of agents.
- In agent based GRNs, the encoded information in genes form the interaction rules for agent behavior and some of the agent behavior (structural agents) will determine the behavior of the robot directly. In network-based frameworks, the genetic information is

only used to form the network structure and it does not give direct effects to the robot's behavior. For network based framework, the development of the control system and the control signal produce are based on two separate processes but agent based GRN integrate these two processes into one.

2.4.2 The motivation of our GRN framework

The reason of using agents instead of networks is based on the following concerns:

Under unknown and changing environments, the system needs to be able to dynamically change to respond to a particular context. Furthermore, in our research, we assume that the individual control system has no knowledge about the environmental situation at the initial stage and the individual controller also lacks the knowledge about the global task. Under such situation, evolving the particular solution schema (represented by the network structures) is not efficient because the optimal solution can be different at different times. Considering a changing environment, the ability to 'know' the current situation and providing a quick response is important for adaptation. The agent-based model could separate the complex environment information into variable features which are corresponding to the particular agents in the model. Considering possible interaction between agents, the interaction rules of agents will evolve by selecting the suitable agents through the GRNs. Such agent-based model has been utilized by studies of artificial immune systems (AIS) (see Glossary for the definitions) and it has been shown to perform well in a variable environment based on its inherent degeneracy ([128]).

In complex adaptation, quickly responding to the particular environment might still not be sufficient. Proficient adaptation may need systems also to consider the context and have some mechanisms to regulate its responsive behavior based on different context. In our framework, we have extended the artificial immune system framework by encoding the artificial gene regulatory networks in a genome and the context based information (for example, agent A activates gene B and then agent A, B working together. Here, agent A represents the context of agent A and B working together) will be saved as the part of GRNs in each individual gene. Different to other artificial GRNs in bio-computing, I use the encoded genetic information to create the agents one by one (developmentally). This design gives the system a template in the genome to evolve while it also allows the system to flexibly rearrange such template during runtime based on the environmental and context feedback (i.e. agent A could activate gene B or gene C, that is the description in template; however if gene B in the genome has been repressed as a consequence of previous events or a current environmental condition, agent A will then only be able to activate gene C).

Because our framework has been specifically designed for a dynamically changing environment, some of its implementations could lead to poorer performance in a stable environment or for performing tasks that could be done based on certain predefined rules. Under stable conditions, network-based approaches can probably reach the optimum faster because of their more stable behavior (as sometimes observed for ANNs at the start of our simulations, see further). However, the potential more unstable behavior caused by the more variable performance of our agent-based implementation can be coped with under the swarm robots scenario because, from the whole population, only the most suitable robots are selected to survive and propagate. However, in a single robot scenario, this might pose problems, especially when the task is not very complex. Therefore, this framework mainly aims to help swarm robots adapt to a changing environment (without a well-defined task). Under a complex and dynamic environment, the variable self-organized behavior patterns can increase the adaptability of swarm robots (see details in chapters 4 and 5).

2.4.3 The comparison in our research scenario

To study the effect of the interaction between agents, we chose more classical network-based controllers as a comparison. In our research, first we aim to test the potential of considering interaction and gene regulation in evolution and then compare the efficiency of particular approaches in certain tasks. Based on these considerations, we firstly chose to develop a comparatively simpler evolutionary ANN-based controller instead of directly using other well-known frameworks such as those used in Neuroevolution (i.e. hyperNeat[129], AGE-ANN[71],H-GRN[130]). The reasons of doing this are the following:

First, our framework is a first implementation and we feel there is still much room for improvement. For instance, if the results show that the interaction in GRN can indeed improve the adaptability, we still could continue to improve the performance of robots. So far, the GRN framework and the interaction rules are both comparatively simple. Therefore, the first comparison should focus on observing the effects of the interaction at multiple levels (i.e. gene level, GRN level and organism level) rather than making a comparison based on the practical task with other well tested network based approaches.

Second, the efficiency of different controllers highly depends on the task and scenario. The well-known frameworks (i.e. hyperNeat, AGE-ANN,H-GRN) in Neuroevolution usually focus on constructing the network topology and these methods study for a well-defined task scenario. In fact, as already stated, with a fixed task, network based controllers may have better results than our GRN controller. However, this does not mean that network based controllers are better for all scenarios. In our experiment, we emphasize the unknown tasks (‘unknown’ means the environment, methodology in the task is unknown or dynamically changing. i.e Mars exploration task) and changes in the scenario and this does not always fit the complex network based structure because more complex networks may need much time to adapt to a new environment or new methodology. Without an extra dynamic regulation (learning), more complex network topologies could not improve their performance under such scenarios (this has been clearly proved by randomly created the complex network based controller in our previous experiments). For example, when the environment suddenly become simpler (i.e. more food and fewer predators), robots may need to quickly change strategies like defense or hibernation and then focus on the simpler food searching task, because otherwise the robots will lose the opportunity to gain energy. With a dynamic regulation (like in our tested ANN, the learning program could tune the weight matrix of ANN based on the feedback of robot performance), network based controllers can also possess the ability for rewiring, however whether the complexity of the network structure make the rewiring more efficient is still an unknown question. We assume that the complexity of the efficient network needs a balance between the task and environmental context. However, the learning program in ANN can only interact with the feedback value individually and they cannot interact with other components within the system. Lacking such interaction between agents renders the ANN fragile for network rewiring, especially when the network is complex. How to manage such a balance in ANN (like using agent based GRN to regulate such network rewiring instead of regulating gene expression) could be an interesting topic for future work. Since we do not know whether more complex networks will fit the dynamic regulation yet and removing the dynamic regulation will cause an unfair comparison between network based controller and our GRN controller, we choose a certain fully connected network in the ANN based controller. Of course, we aware that the indict encoding in GRN framework may give the system an advantage on scalability and variability but more variable and complex network structure also may limited the efficiency of network rewiring. For making a fair comparison, we always ensure the maximum complexity of the GRN controller is less than the complexity of the ANN controller (by limited the number of the agent and the interaction frequency between agents in the GRN controller). By this, we ensure the GRN framework cannot have the direct advantage from its indirect encoding during the simulation. In future, we will continue to investigate the potentials of the indirect encoding ANN and make more comparisons between indirect encoding ANN and GRN.

Third, comparing the efficiency of different controllers is related to many factors and the complexity on mechanisms is only one factor here. For launching a convinced benchmarking test scenario with a more complicated network based approach, it will require lots discussion and work to do. For the first test of the framework, we didn’t have enough time to cover this yet, however this is a necessary and interesting work in the future study. At the moment, this is work in progress. In our previous work, I try to make a proof of concept (identify the effect of the interaction at multiple levels) and compare with a simpler ANN in this research. For future work, I’m planning to compare with more complex ANNs.

	Direct encoding Neuroevolution Network	**Indirect encoding Neuroevolution network**	**AIS**	**ANN in the experiments**	**Agent based GRN**
Artificial genome and gene evolution	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Complex Gene Regulation	**No**	**Yes**	**No**	**No**	**Yes**
Agent based model	**No**	**No**	**Yes**	**No**	**Yes**
Interaction between genes and environment	**No**	**Yes/No**	**No**	**Yes**	**Yes**
Interaction between agents	**No**	**No**	**Yes**	**No**	**Yes**
Network structure	**Yes**	**Yes**	**No**	**Yes**	**No**

Table 1:Comparison between different bio-inspired approaches

The table above shows the different approaches (see them on each column) with different features (see them on each row).

2.5 Conclusion

Through gene regulation, biological organisms can present multiple phenotypes based on the same genotype in different environmental conditions. This feature gives biological organisms an advantage to adapt to a fluctuating environment. During the adaptation of the organisms, self-organized gene regulatory networks play an essential and critical role in the organisms and they are closely linked with the adaptability of the organisms. To achieve similar adaptability on simulated virtual (or digital) organisms (swarm robots), we mimic the same principles, which are inherent to the biological genome and GRNs. The following diagram (Fig 2.5) illustrates the general structure of the whole framework.

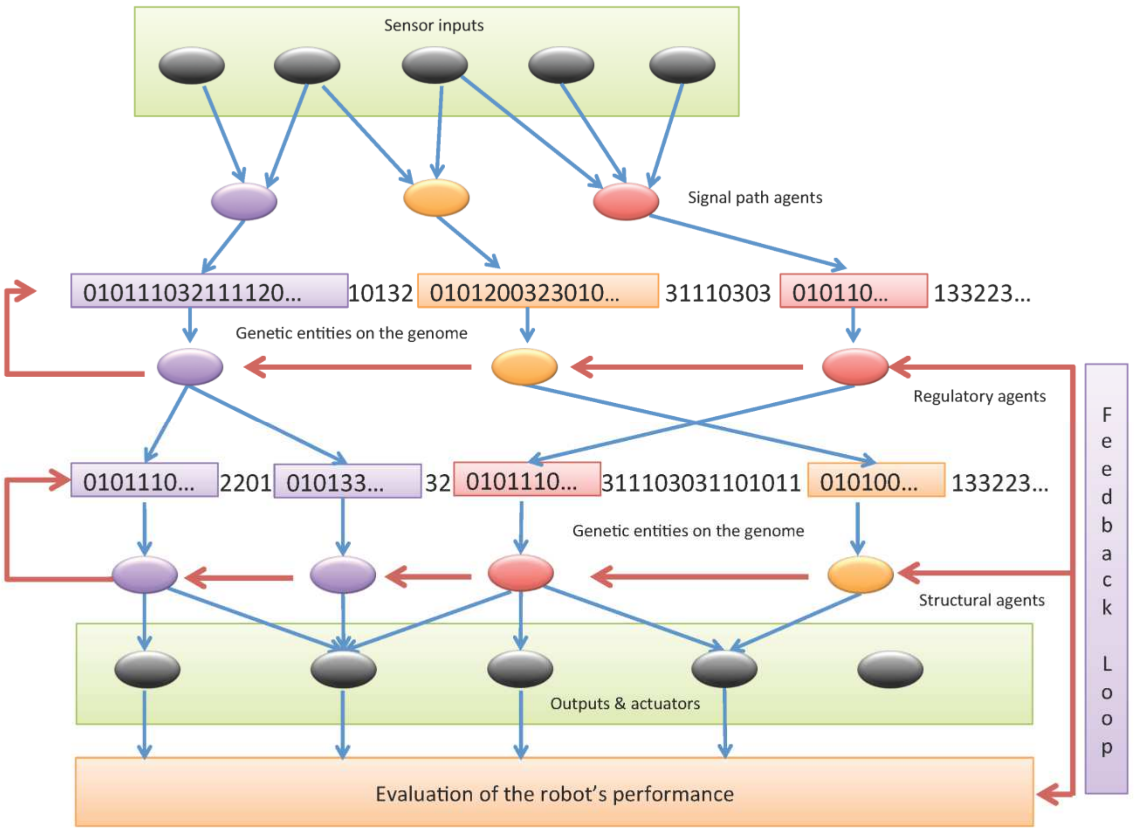


Figure 2.5 The dynamics of the GRN, encoded by the artificial genome and its interaction with the environment, are modeled by an agent-based system.

The red arrows represent the feedback flow from the environment to the agents and from the agents to the genome. The blue arrows represent the information (signal) flow in the agent-based system. Boxed sequences represent genes on the genome that are turned into agents (mimicking the active part of the GRN). At the signaling level, signaling agents are a special type of agents that activate other genes, based on the integrated signals they receive from the environment. At the regulatory level, the information flow mimics transcriptional regulation (from genes to agents to genes, …). Finally, structural genes encoding structural agents invoke a phenotypic behavior.

With such artificial genome and GRN, the virtual organisms could alter their phenotype based on the current environment. In the virtual organism, different environmental inputs will activate the corresponding genes and produce different agents in the GRN. During this process, each selected gene will reach its own adaptation within a certain evolutionary context. Through the interaction between the genes and agents, the GRN model and the phenotype of the organism will emerge as a collective behavioral pattern of agents (gene products).

On the other hand, for longer term changes in the environment, the selection pattern on different genes could vary during evolution and the particular environments may give a particular selection on its relevant genes (the relevant gene here means that the gene could be activated by the signals of this particular environment) on the genome. Furthermore, such influence of genetic regulation could be implicit and the gene activation could be presented as a chain reaction (interaction) in the GRN. Our simulation framework provides an efficient approach to address these kinds of far-reaching implicit effects on genome evolution by replaying the similar context at the short term and long term levels of artificial evolution.

The motivation of developing this framework is based on three aspects. First, in our simulation, we believe that a more realistic and dynamic evolutionary context should be based on a more fine-grained interaction. Of course, there is no way to know the details of all interactions (over time) during the evolutionary process but some known interaction rules might offer a clue for extending the relevant context in simulations. Through imposing stable and well-identified rules of interaction between well-known components in simulation, we can, in our simulations, dynamically implement the possible interactions between different components or at different levels (i.e. based on gene regulation and the interaction between genes, we could infer the behavior of organisms and the interaction between organisms). These will create a more holistic context that is based on a few simple but well-identified interactive rules of evolution. Although such simulated context may still not really compare to real biological contexts due to necessary oversimplification, it can offer a more complex dynamic model that considers domino effects caused by interactions between genes, GRNS, organisms and the environment.

Second, multiple level interaction in our simulation could provide a more interesting evolutionary process trajectory. Based on catastrophe theory [131], Small changes in a nonlinear complex system can cause equilibrium to appear or disappear, or to change from attracting to repelling and vice versa, leading to large and sudden changes of the behavior of the system. Such catastrophic changes also frequently happen during evolution and are regarded as evolutionary transitions. These evolutionary transitions are relevant in many evolutionary studies. However, simulating such transitions is a great challenge because the whole system needs to go through numerous unstable but necessary states before transition happen and each of these intermediate states may be difficult to identify. In mathematics, people use potential functions (A mathematical function whose values are a physical potential) to describe a catastrophe dynamically but such functions only can include a limited number of variables. In a similar philosophy, our simulations use the interaction of individual components to describe catastrophe in evolution. In our simulations, every next time step is the derivative of the interaction at the current time and every local bifurcations (local bifurcation means when a parameter change causes the stability of an equilibrium to change [132]) in the process is determined by the particular interaction events). Based on a few interaction rules, we can simulate the effect of different environmental conditions in evolution to individual interaction and for each single interaction event, its relevant conditions will be modular and identifiable. After we identified all interactions at one time step, we can simulate possible domino effect, interactions and transitions at the next step in evolution(based on the interaction rules).

Third, the influence of interactions could entail different time stages and levels of evolution, so it is also a good indicator to find correlations between events over different time stages or between different levels. Nature forms a nested architecture where evolutionary processes act at different levels and therefore we refer to biological systems as a nested architecture. Evolutionary systems are constituted by interactive modules that, in turn, are also a kind of evolutionary system at a lower level. For instance, Sloan et al. [133] suggested that even social groups could act as 'singletons' in evolution. The following figs 2.6 give some examples about multiple level evolution and nested system structure.

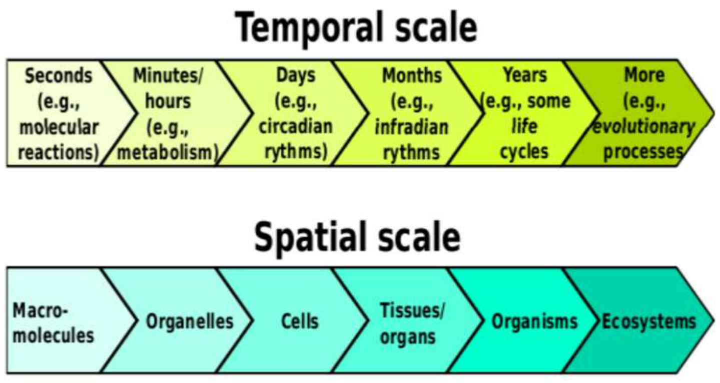

Temporal scale
Seconds (e.g., molecular reactions)
Minutes/ hours (e.g., metabolism)
Days (e.g., circadian rythms)
Months (e.g., infradian rythms
Years (e.g., some life cycles
More (e.g., evolutionary processes
Spatial scale
Macro-molecules
Organelles
Cells
Tissues/ organs
Organisms
Ecosystems

Figure 2.6 An example of the different levels of evolution at the temporal and spatial scale. *(Cited from[134])*

Based on previous research in evolution, we already know a great deal about how evolution happens at different levels. However, we still know little in general about how different evolutionary processes that play at different levels co-evolve in a common, complex and changing environment (like in the natural environment). In this case, the corresponding evolutionary context is unknown or dynamically changing. If we only consider evolution within a certain context only at one evolutionary level, many interactions (i.e. interaction between different evolutionary levels) may be greatly simplified or ignored. Under natural evolution, the context (like a particular selection pressure) actually always evolves so using a predefined context in simulation can definitely harm the reality of the evolutionary model. The contributed fitness effect on particular traits for adaptation is also dynamically changing and such changing is based on the evolutionary context which is determined by interaction of evolution. Acknowledging such multiple level interactions inevitable increases the complexity of the evolutionary model but this complexity is inherent to natural evolution and closely connected with the evolutionary process. Denied or over simplified complexity may lead to serious limitations in the interpretation of our simulation studies.

To better understand how the evolutionary processes behave at the different levels, we need to know how they work together and we need to simulate the corresponding dynamic context during evolution with a holistic view. The traditional simulating framework usually is not sufficient to perform such simulation because they often lack implementation of multiple evolutionary processes occurring at different levels of evolution. In addition, these traditional frameworks also tend to ignore the interaction between the different evolutionary processes and predefine the context (such as selection pressures, fitness functions etc.). Compared with using fixed selection based on certain fitness measurements, using the multiple level interaction concept can help to create a more realistic and dynamic selection and fitness evaluation framework. Our recent simulations, taking into account interaction at multiple levels, have shown the emergence of complex adaptive patterns, where in experiments not considering these multiple level interactions, such complex adaptive patterns have not been observed (see the details in chapter 5). Such results could be a clue and imply the interaction in evolution actually could accelerate the emergence of complex adaptation.

The particular genome and gene design in this research is for supporting the variability of the agents in GRN. For example, the regulatory agents in this system could have more than 1000 different binding motifs and various regulation behaviors. This requires a corresponding extension on the gene structure. Such diversity inevitably brings extra complexity into the genome but it is also necessary for agents to respond to the different environmental input patterns. Insufficient variability will lead to different environmental inputs actually giving the same influence to GRNs. In fact, for adaptation in a simple or static environment, such complexity and variation could be reduced (i.e. by reducing the length of genome, extending the binding range and so on) because the comparatively simple response behavior may be enough to deal with all possible situations. However, an over-simplified system and insufficient variability on behavior will be deleterious and unrealistic in a more the complex environment. In general, monotonous environmental inputs will activate fewer agents than complex environments and this corresponds to a small GRN. This way, the complexity of the real system is self-regulated based on the environment and can evolve during the simulation.

Current limitations of the framework are pattern recognizing functions and a tracking system. For analyzing the influence of interactions more precisely, the simulation needs an efficient pattern recognizing function that could measure emergent behavior patterns during the simulation. The development of such functions is planned for future work.

Chapter 3

Artificial life simulations and virtual robots

Author contribution

The content of this chapter was written by myself. It resulted from many fruitful discussions with both my promoters and all partners I had the chance to work with during my PhD studies. Figure 3.1 are cited from my previous paper.

The environment plays a critical role in natural evolution. Natural selection and adaptation are both greatly influenced by the interaction between organisms [135] and by occasional events occurring in the environment [136]. Therefore, the environment has a fundamentally important impact on the evolutionary processes. To better understand evolution in nature, we need to investigate the corresponding environmental context as well [137-140]. Artificial life simulation is one effective approach to provide such environmental context by which we can study the interaction between environment and organisms (i.e. [133,141]).

In this chapter, I introduce the implementation of an artificial life simulation. In our artificial life simulation, we define the whole environment as a complex adaptive system and use virtual robots to represent the individual organisms in the environment.

3.1 Introduction

The phrase 'artificial life' was coined by Christopher Langton, who envisioned a study of life as it could be in any possible setting [142]. As Mark A. Bedau defined in [143], "the contemporary artificial life (also known as "ALife") is an interdisciplinary study of life and life-like processes that uses a synthetic methodology". Therefore, these kinds of studies mainly focus on imitating some aspects of biological phenomena through the use of simulations with computer models, robotics, and biochemistry. Despite the fact that there have been various implementations of the different ALife studies, all these studies share three common features. The first is that the simulations in ALife always involve complex adaptive systems and emerging behavior. The second is that the emerging models in the simulation are self-organized and the result of the interaction of the components at the lower level. The third is that the evolutionary processes in the ALife simulation are based on open-end evolution[144], meaning that the evolutionary models are not predefined and they actually could evolve themselves during the simulation. In fact, these features form also the common principles of biological systems and they are actually the main difference between biological organisms and traditional engineering systems. ALife researchers use such bio-inspired principles to investigate biological phenomena and evolution in a different way than with more traditional approaches. The results of ALife research have been very promising and the artificial life techniques are becoming more and more accepted in mainstream evolutionary studies (see the examples in [143,145,146]). Tierra[99] and Avida [97] frameworks are the two typical and popular ALife platforms in this field. In both simulations, researchers use independent programs (with own memory and CPU time) to represent the particular virtual organisms. Each virtual organism has a piece of executable code as its genome. These genomes can be changed or reassembled during the replication of the organism and the different genomes could give the organism different functions during the simulation. The various organisms can adapt during the simulation and interact with each other based on their own functionality. The results of such simulation could show a complex collective behavioral pattern evolved in the population through the interaction among the individual organisms. A few examples and advantages of these kinds of simulations have been elaborated by by Bill O'Neill in [33]. In our simulation, we use a similar approach to simulate complexity in the environment. However, for identifying the correlation between gene evolution and the evolution in an ecosystem, we extended the above framework with an embedded GRN simulation. In our framework, each virtual organism, instead of just executing some code, undergoes an independent simulation based on gene regulation (see more details in the chapter artificial genome and GRN). The output pattern of the GRN simulation will decide the different functions of the virtual organisms in our simulation. Hereby, the changes on the genome will not directly have an explicit effect on the phenotype, but implicitly influence the corresponding GRN. This setting gives us a chance to combine the evolutionary context at the genetic level and the ecosystem level into one holistic model and allow people to identify the possible correlations between these evolutionary processes.

3.2 Methodology

In our framework, the main aim of the artificial life simulation is to mimic the dynamics and complexity of natural environments to study evolution. To implement this, we adopted the principle of swarm evolutionary robots to present the organisms. On the one hand, each of these robots has a GRN and genome. On the other hand, these robots form interactive elements in the whole environment. Based on the collective behavior of these robots, the ecosystem and various local niches can emerge from a self-organization process. The whole environment in our artificial life simulation is regarded as a complex adaptive system and is constituted by the local niches of the robot organisms. Through the interactive behavior of all robotic organisms, this environment can dynamically change. For our simulations, we design the scenarios based on evolutionary game theory[147,148] and the dynamics of strategy change on one robot organism may have an effect on the strategies of other robot organisms. Such dynamic evolutionary context (the strategy changes and relationships of organisms in history) in a changing environment gives the artificial gene evolution a more realistic background. By comparing with the evolutionary trajectories in gene evolution, we can also identify the correlation between the evolution of the ecosystem and the genome.

3.2.1 General scenario

The simulation map is similar to the cellular automaton model, which is a discrete model studied in computability theory [149], mathematics [150], physics [151], complexity science [152], theoretical biology [54] and microstructure modeling [153]. Like the cellular automaton, the map in our simulation also consists of a regular grid of cells (90*90), each in one of a finite number of states. The possible states for these cells could be empty, occupied by food or occupied by the digital organisms. In fact, each cell here represents one of the basic spatial locations in the simulation. Swarm robots are distributed over the cells of the map during the simulation and each single robot occupies one cell at a certain time. Different from the cellular automaton is that the next state of each cell is not only based on the neighborhood but also on the behavior of the robots. Robots can move around and can also eat the discovered food sources. The subsequent effects of these behaviors change the states of the corresponding cell. Selection and fitness of the robots are all based on energy (in the form of food resources). For surviving during the simulation, robots have to consume a certain energy at every time step. Different robots can have different energy consumption styles, each of which comes at a cost. For instance, maintaining aggregation with other robots will cost extra energy, but comes at the benefit of being able to acquire more costly food (see further).

During every time step, the robots receive new sensor inputs. More specifically, they will sense the number of surrounding robots and food sources. The robot then determines its next action based on its input and controller output signal values. Different robots will choose different actions and consume different amounts of energy, depending on their individual activated GRNs. As the environment changes dynamically, the activated GRNs could also change based on different environmental inputs and then the corresponding phenotype (behavioral patterns) may change through time.

During the simulation, robots can choose different behaviours to obtain energy. The decision of choosing the appropriate behaviour is crucial for the adaptation of the robot. The available options include searching food for energy, attacking other robots for stealing energy or aggregating with other robots to share energy. For instance, sharing can improve defence and food searching and preying ability.

If a robot does not have enough energy to cover its basic living energy requirements, it will be regarded as dead and removed from the simulation. Detailed parameters that govern the simulation setup are given in Table 2.

3.2.2 The models in the artificial life simulation

As discussed above, the motivation of our artificial life simulation is to provide a realistic and dynamically changing environment for the evolutionary robots, very much like natural ecosystems. Based on this, we use the essential common model that can be found in most of ecosystems of nature. The model includes three important aspects such as organisms, food and environmental conditions. In this section, I will explain each these aspects separately in the following part.

Organisms

All organisms in the simulation are represented by the same kind of virtual robot. These robots all have their own independent controller although they all have the same sensors and actuators. By having different controllers and genomes, every robotic organism actually can respond to the environment through a different behavior. Similar approaches also have been used in many previous simulations such as [57,154,155]. In our bio-inspired framework, the controller of the robot usually is a dynamic GRN while the output of the GRN simulation determines the behavior of the robot. In the control experiments, we also use an evolutionary neural network (ANN) as the controller of the robot for comparing the differences between the behaviors of these two different frameworks.

All resources of the robotic organisms have been abstracted as the energy of the robots. In every time step of the simulation, the robot status and all activities depend on the consumption of energy. If the energy of the robot drops below a certain threshold, the robotic organism will die. If the energy of the robot becomes too small for a particular function, the robotic organism will not be able to perform the corresponding behavior.

The common functionalities all robotic organisms

Each robot has seven possible functionalities, each of which comes with a different energy cost (energy consumption style). Some functionalities need the controller to give the control signals when they are performed.

1. The following functionalities are performed by default and not controlled by the controllers. Performing these functions don't consume any energy:

Sensing: At every time step, each robot can sense the number of other robots or food sources within a two-cell distance.

The map or grid is divided into 8100 cells. During every time step, each robot can sense the surrounding regions and its located region. The sensing would tell the robot how many food resources and how many other robots exist in the sensing range.

Preying: the robots can increase their energy level by consuming the food sources located in the same cell as the robot. For different sources of food, the preying might require the robot to possesses different amounts of energy in advance (see Table 3). If preying is successful (i.e. the robot has enough energy to take the food), the food will be removed and the energy content of the food will be added to that of the robot. As long as there is a food source in the same location as the robot, the robot will automatically try to eat the food.

2. The following functions are driven by the signals of responsible controllers. Performing these functions require extra energy:

Moving: a robot can move to the surrounding cells in the two-dimensional matrix.

Energy cost: 5

This function is controlled by two kinds of signals

Controlling signals:

Signal 1: Value > 0 means forward; Value < 0 means backward; 0 value = no movement

Signal 2: Value > 0 means go right; Value < 0 mean go left; 0 value = no movement

Attacking: Every time step, the robot could choose to attack one other robot, which occupies the same cell. If the attack is successful, the attacking robot inherits the energy of the robot that has been attacked and the latter will be removed from the simulation.

Energy cost: 2

This function is controlled by one kind of signals

Controlling signals:

Signal value >= 0 means the robot will try to attack surrounding robots

Signal value < 0 means the robot will not try to attack surrounding robots

Defending: robots can defend against an attack by another robot. This is simulated by investing a certain amount of energy (referred to as defense value). When the robot i is attacked by another robot, its defense value (def_i) plus its energy (Re_i) will be compared with the energy level (Re_h) of the attacking robot. If the attacking robot's energy is higher (Re_h > def_i+Re_i), the defense will be broken and the attacked robot's energy will be transferred to that of the robot who attacks. Otherwise, the defense is successful and costs nothing. Defense values can also be accumulated during the lifespan of a robot.

Energy cost: depends on the controlling signal (see further)

This function is controlled by two kinds of signals

Controlling signals:

Signal 1 decides on defending behavior:

Value >= 0 means the robot will increase the defense value

Value < 0 means not increase the defense value

Signal 2 decides on how much energy is used for defending (only available when Signal 1 decided on defending):

Replication: when the energy level of robot i exceeds a minimal threshold ($minR_i$), the robot can choose to replicate. For every time step, the chance of replication (Rc) for a given robot i is based on the following equation:

Rc = (Re_i-$minR_i$)/500; (CurR<MaxR) (Table 2).

After replication, the residual energy of the replicating robot will be divided equally over the parent and daughter robot. New robots will have the same characteristics as the parental robot except for the defense value. When the maximum robot population size (it refers to the total population which is for all robots in the simulation) has been reached, the replication function will be disabled until the population has reduced again.

Energy cost: 1 (and each robot will have the half of the rest energy)

This function is controlled by one kind of signals

Controlling signals:

Signal 1: This value determines the threshold for replication (provided the population size has not reached a maximal level)

Aggregating: At each time step, a robot can send an invitation to another robot. If the invited robot accepts the invitation, both robots will aggregate, merge their energy levels and form a new robot (referred to as robotic organism). The GRN controllers will be integrated into the new robot by fusing their Signal signals. At any point, one of the joined robots can stop the aggregation and become single again. Subsequently, the separated robots will receive their part of the total energy (total energy divided by the number of aggregated robots in the organism) of the previously joined robot. The advantage of aggregation is the joined 'robotic organism' will have more energy and greater defense capabilities. For example, the maximum energy of a robotic organism (including n robots) will be equal to $\sum_{n=1}^{n} Maxe(n)$ (Table 2). Such robotic organism will perform better on preying, defense and attacking. The disadvantage of aggregation is that, for every time step, it comes with an energy cost (see Table 2).

Energy cost: 1

This function is controlled by two kinds of signals

Controlling signals:

Signal 1 decides whether to aggregate with another robot

Signal 1 Value >= 0: the robot will aggregate with the surrounding robot value

Signal 1 Value < 0: the robot will not aggregate with the surrounding robot

Signal 2 Value: determines whether to disassemble (only available when the robot is part of a robotic organism):

Signal 2 Value >= 0 the robot will stay aggregated

Signal 2 Value < 0 the robot will disaggregate

Table 2:Parameters used in the artificial life robot simulations

Symbol	Explanation	Default setup value
X	horizontal position in the matrix	90
Y	vertical position in the matrix	90
MaxR	The maximum number of robots in the simulation	200
CurR	The current number of robots in the simulation	The default initial total population is 100
Fnum1	The number of food sources of type 1 (see Table 3)	500
Fnum2	The number of food sources of type 2 (see Table 3)	100
Fnum3	The number of food sources of type 3 (see Table 3)	50
Fr	The default rate for food increase	30% novel food sources/20 time steps
Re	The robot current energy level	The default initial value is 500
Be	The basic energy consumption required for each time step	7
Ae	The energy consumptions for actions	based on the specific action (see Materials and Methods, main manuscript)
Ee	The extra energy consumption for aggregation during each time step	1
Maxe	The maximum energy for a single robot in the simulation	1200
minR	The minimal energy threshold for replication	Given by the genome of the robot. The range of this value is from 500 to 1200

Robotic organism energy consumption and behavior model

During the simulation, at every time step each robotic organism checks its energy. If the energy level is above 0, the organism will sense the environment and update the controller. Based on the different environmental inputs, the controller will give a corresponding response. As discussed above, these responses could mean that the robot performs a particular function from the functionality lists. However, before performing any functions, the robot will check its energy level again and update the new energy level. Insufficient energy of a robot will stop its intended function performance.

The total energy consumption for one robot during one time step is described by the following equation (with n corresponding to the number of functionalities in time step i):

For a single robot:

Total energy consumption = $\sum_{i=1}^{n} Ae(i) + Be$

For a robot that is part of a robotic organism (aggregate of robots):

Total energy consumption = $\sum_{i=1}^{n} Ae(i) + Be + Ee$ (see Table 3)

Each function of the robot can only be performed once at one time step, but the effect of one particular behavior may be able to remain in the simulation for multiple time steps. For example, the sensing inputs at one time point can trigger the new agents in the corresponding embedded GRN and these new agents will have effects on the behaviors of the robot for many time steps.

Food

As stated, the robotic organisms live in a two-dimensional 90 by 90 matrix or grid in which a number of energy sources (e.g. food) are distributed. Several types of food source exist that differ from each other in the minimal amount of energy required to access the food source (see Table 3). Initially, food sources are randomly distributed over the 90 x 90 grid and each cell can only contain one food source. After a pre-set number of time steps, the system will add new food sources with a certain replication rate (Pr) according to the following function:

$$\text{Pr} = \left(\frac{500 - Fnum\_x}{500}\right) * Fr$$

With Fnum_x being the current total number of food sources of type x that are already present in the simulation and Fr being the default food increase rate. In the simulation, the maximum energy level that a single robot can achieve is restricted to a value of 1200.

The recovery policy of the food in this simulation imitates the growth of the producers in the food chain and we assume the food does not have the ability to move. In addition, robots need to choose the suitable kind of food based on its own status. For example, a single robot cannot take the type 3 food alone because it requires the robot organism to have at least an energy level of 2000 to take the type 3 food while a single robot can only gain an energy level of 1200 at most. Through this implementation, we force single robots to aggregates if they need to get access to food of type 3.

Table 3:Different types of food sources used in the simulation

Food type	Food energy	Distribution range (by coordinate value)	Requirement
Type 1	300	X:0-60;Y:0-90	No requirement
Type 2	1000	X:61-80;Y:0-90	Can only be eaten by robots that have energy levels >= 800
Type 3	3000	X:81-90;Y:0-90	Can only be eaten by robots that have energy levels >= 2000

3.2.3 Environmental conditions

Except the interaction between organisms and food, this simulation also considers the impact of the change on the general environmental conditions to adaptation and evolution. We have therefore simulated two kinds of common conditional changes in the simulated environment.

Geographical diversity

In initializing the simulation, we impose the tendency that food of the same kind prefers to be located close to each other. Therefore, in the different regions of the environment (the grid), the food distribution pattern will display significant diversity. Due to the food replication policy used in the simulation (see the last food section in 3.2.2 for more detail), one kind of food of will only produce the same kind of food in the surrounding area (In our simulation, food is implicitly regarded as plant and this rule mimics the plant community in nature), therefore such geographical diversity in the environment could be kept during later stages and can profoundly influence the adaption of the robots. Robots in the nearby vicinity could use their behaviors to construct their own local niches and then develop different surviving strategies.

Climate

In the natural environment, the adaption of organisms depends on many environmental factors like the temperature, the oxygen concentration, altitude and so on. These factors actually frequently change during the life of organisms and such fluctuations are closely linked to the dynamics of adaptation. In our artificial life simulation, we provide a similar fluctuation in the environment by imposing artificial climate change. During the simulation, at certain times, all robotic organisms have to consume extra energy for movement and survival. The values of these extra energy costs differ at different times during the simulation. Every 100 time steps, these values change for all robotic organisms. For every robot, these values are same at the same time point. During the simulation, there are four different extra energy cost rules that separately dominate the actual extra energy consumption of the robotic organisms in turn. Like seasonal changes in nature, the general conditions of the environment thus also change in our simulation. The Climate environmental condition is circular changing based on the time and it does not interact with the robot organisms. In our simulation, the climate refers to a null model of environmental circular changes in evolution.

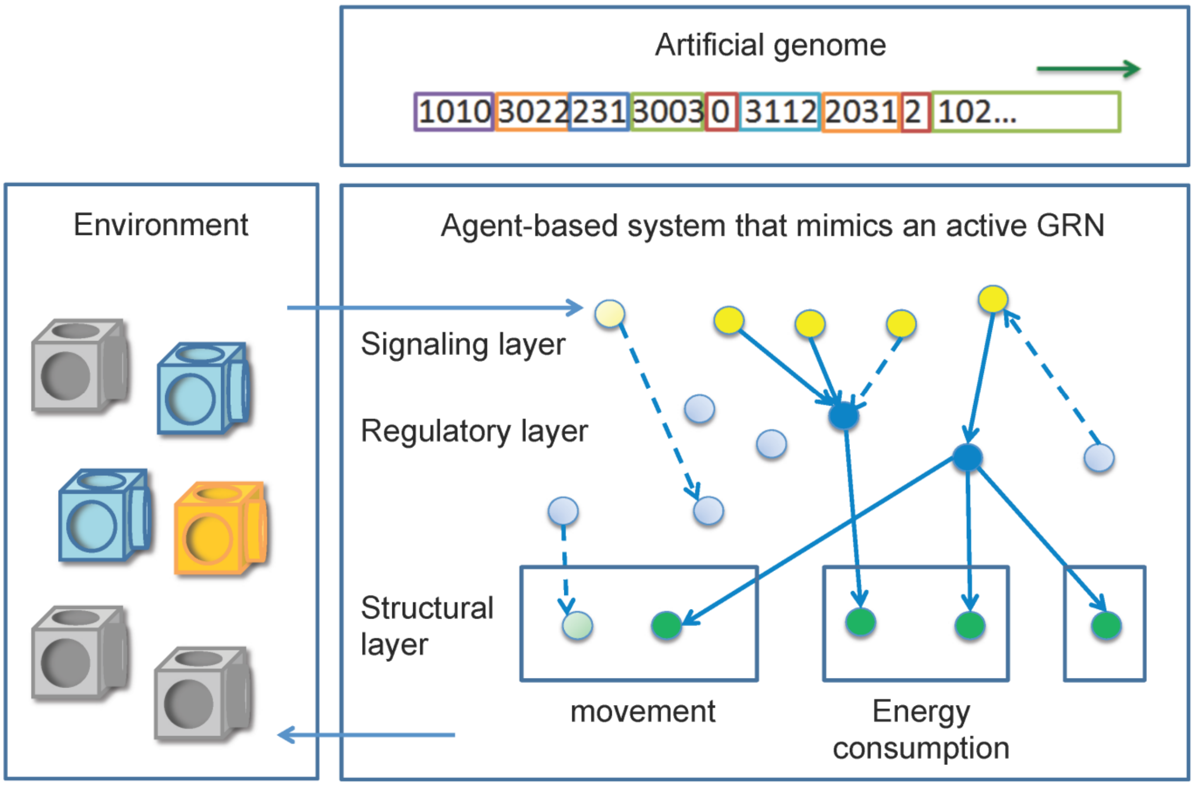


Figure 3.1 Overview of the bio-inspired framework in the simulation.(Cited from [117])

The GRN-based controller actually consists of two separate layers. First, an artificial genome (AG) (top panel) encodes the full (core) regulatory network (lower panel, all nodes and edges), i.e. all potential interactions that can take place between signaling (yellow), regulatory (blue) and structural genes (green 'nodes'). Evolutionary forces act at the level of this genome. Second, an agent-based layer (lower panel) that corresponds to the 'activated' regulatory network (colored nodes and full lines). The agent based layer mimics the translation of the core regulatory network into an activated network, following the rules embedded in the AG. Agents thus correspond to activated genes. The agent-based layer constitutes the active controller of the system and drives the behavior of the robots (left panel). Key to our approach is the condition-dependent activation of the core genome encoded by the AG into an activated network modeled by the agent based layer resulting in the fact that only the translated part of the core network will affect the robots behavior.

3.3 Conclusion

The natural environment is complex and there is interaction with the organisms that live in the environment. In addition, there are many trade-off situations that require different adaptive strategies. For example, under starvation (i.e. few available food sources), saving energy could become more important than the search for new energy, but priorities may change when food becomes abundant again. These complex and dynamic interactions make adaptation and evolution in nature very different from what would happen in a simple and static environment. The main differences are that 1) there is no explicit best adaptive solution in a complex environment and thus adaptation in a complex environment will need to be a dynamic equilibrium rather than a certain optimization process, and 2) the previous evolutionary context will have an essential impact in the later environment and selection. The interaction between the evolutionary systems and the environment is ubiquitous in nature, therefore, the adaptability and fitness measurement in a complex environment are also implicit and are based on the context. Back to a static unchanged environment, the selection pattern could be always the same and the influence of the context could be ignored during the simulation because the previous behaviors cannot change the environment thoroughly.

To simulate natural evolution and adaptation, our new simulating model particularly mimics specific features of natural environments. By imposing for evolutionary trade-off scenarios (e.g. aggregating other robots for sharing energy or preying on other robots for stealing energy) and considering the interaction between the robots and the environment, our artificial life simulation also can simulate dynamics and contexts observed in natural environments. Furthermore, the selection of the robotic organisms is dynamically changing during the simulation and is affected by the context as natural selection is.

In conclusion, with this simulating environment, our framework separately runs the independent gene evolution and regulation in each individual robotic organism and then gives it the collective behavior of the robotic organisms based on the internal simulated GRNs.

As can be observed from Fig 3.1, the different evolutionary processes at different levels (genetic level, phenotypic level and ecosystem level) can be regarded as one integrated complex system and the interaction among all components of this complex system will determine evolution and adaptation. Within this framework, adaptation of the different components and the corresponding selection are dynamically self-organized during the evolutionary process. As a result, the current simulator allows the evolutionary processes at multiple levels to interact with each other and whole evolution can be influenced by the corresponding evolutionary context. For evolutionary studies, the benefit of this new simulator is that it in particular can help to identify the effect of the interaction among different levels in evolution and provide the convinced trajectories or context for the simulated evolutionary processes(see more examples and details in the chapter 5).

Chapter 4

Bio-inspired GRNs improve the adaptability of virtual robotic systems

Author contribution

I performed all the development, experiment implementation, result collection and computational analysis described in this chapter. All these works were done under supervision and significant contributions of Yves Van de Peer, Kathleen Marchal and Guy Baele.

The inherent principles of GRNs that we learn from biological organisms play an important role in the adaptation of organisms. Especially for adaptation in a changing environment, gene regulation through the action of GRNs provides phenotypic plasticity providing an advantage to organisms in response to variation in the environment. One of the main aims of our research is to investigate whether utilizing the principles of GRNs also improves the adaptability of artificial systems. For demonstrating the feasibility and potential benefits of using these bio-inspired principles, we used our artificial genome and GRNs as robot controllers in different robotic simulation experiments and observed the effects on the adaptability of the robots. Based on the different functions and tasks of the robots in the simulation, these previous experiments could be summarized into three different classes:

1. Collision avoidance and exploration

   In this kind of experiments, robots need to avoid obstacles while exploring the map as widely as possible. Every single robot in the simulation has the necessary sensors to detect the obstacles. The input signals of the sensors will feed into the movement controller, which is developed based on our artificial GRNs. The output signals of the controller will determine the moving behavior of the robot. In this thesis, we test the feasibility of using our GRN controller in the real time movement control and test the adaptability of our robots in an unpredictable environment.

2. Self-organizing robotic organisms

   This experiment adopts a simplified version of the GRN controller as in the simulated ‘symbrion’ robots (see the detail in the section ‘the self-organizing robotic organisms’) to control robot aggregation. The task requires that the swarm robots need to aggregate and form a robotic organism to perform a certain task. In this experiment, the GRN controller can self-organize (based on the environmental inputs) the adaptive shape of the organism. Furthermore, the GRN controller will also control each swarm robot to complete the aggregation during the experiment.

3. The virtual ecosystem

   The simulation platform and aims of this experiment will be discussed in greater detail in the last chapter. Here, we will focus on the experimental results and on the comparison of the adaptability between the GRN controller robots and the ANN controller robots.

All above experiments have their own particular scenario and use different kinds of robots (see the detail in each corresponding section below). However, they all share the same bio-inspired GRN controller, which is used to provide artificial gene regulation to (the simulated) robots.

4.1 Collision avoidance and area exploration

This section 4.1 is partly reproduced from the following publication:

Yao, Y., Baele, G., Van de Peer, Y. (2011) A bio-inspired agent-based system for controlling robot behaviour. Proceedings of the IA - 2011 IEEE Symposium on Intelligent Agents organized in IEEE Symposium Series in Computational Intelligence 2011 Paris, France.

4.1.1 Abstract

In this section, we present an agent-based GRN controller to control a single robot's moving behavior. This work is the early work of my research so the agent-based GRN system in here is a simpler version than the default version discussed in other place of the thesis. Both versions share the same principles. In this version GRN system, we also use agent-based modeling (ABM) to simulate a bio-inspired GRN based on the artificial genome, with the ultimate goal of providing phenotypic information for a simulated robot. We show that the presence of a feedback loop in the agent based system, along with the corresponding agent replacements, is essential to allow the robot to perform its tasks.

4.1.2 Introduction

The field of evolutionary dynamic optimization deals with the application of evolutionary algorithms to dynamic optimization problems (DOP)[156]. In these problems, the environment changes frequently or is completely unknown and the optimization methods need to adapt their proposed aim to time-dependent contexts. Previous research has seen different approaches to address such problems, such as a bio-inspired agent-based framework, which can be adapted to a highly changing environment with good scalability and flexibility[157]. Other research has shown that degeneracy of solution representation will improve the robustness and adaptiveness of dynamic optimization, aside from mentioning that degeneracy is the main feature of bio-system in natural evolution[158]. In short, these studies show that evolutionary algorithms (EA) have the possibility to solve the DOP by introducing bio-inspired principles. Further, the flexible agent-based structure is inherently suitable to implement such bio-inspired system. In this research, we present a layered structure with a dynamic feedback loop which is inherent properties of gene regulatory networks (GRNs), in order to develop a practical framework for achieving a self-adaptive robot in a simulated scenario. By mimicking the principles and features of GRN in our framework, we expect the agent-based system to efficiently deal with a highly changeable or unknown environment.

4.1.3 Materials and Methods

The GRN controller presented in this work has a little difference with the general GRN framework that I have discussed in the previous chapter. The GRN controller in this work is for supporting a real time control and it is only in charge of the movement of robots. Based on the task, this GRN controller is comparatively simpler than the general framework although it also shares the same principles and models of biological GRN.

Layered approach

In order to simplify the complexity of our agent-based modeling approach, we distinguish three different layers in this agent-based system, which are shown in Fig 4.1. Separating the agent-based system into different layers results in agents that only have connections with the other agents in the same layer. This way, changes to a particular agent in a given (sub)network will only affect that same (sub)network. Further, interactions between agents in different layers can then be replaced by interactions between different layers, thereby simplifying the agent-based system's design.

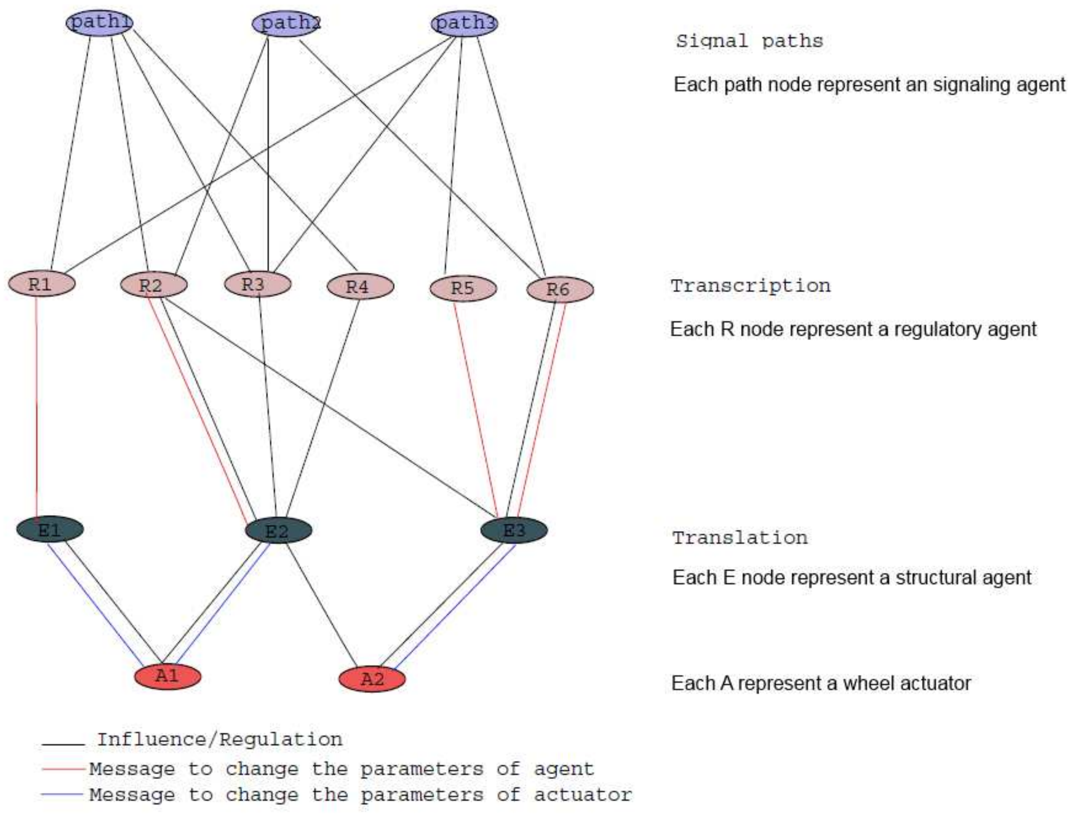

path1
path2
path3
Signal paths
Each path node represent an signaling agent
R1
R2
R3
R4
R5
R6
Transcription
Each R node represent a regulatory agent
E1
E2
E3
Translation
Each E node represent a structural agent
A1
A2
Each A represent a wheel actuator
Influence/Regulation
Message to change the parameters of agent
Message to change the parameters of actuator

Figure 4.1 The different layers in our developed agent-based system[159]

The following layers can be distinguished in our agent-based system:

- Signal path layer: the main functions of this layer are receiving the sensor values and transforming this sensory information to usable values/signals for the agent-based system. This layer can be regarded as a hardware abstraction layer, so that different types of robots (providing sensor inputs of different magnitudes) can be used in the agent-based model. This layer is hence the bottom layer which is connected to the actual sensor of the robots. Each signal path consists of a combination of sensory inputs, determined by a corresponding gene in the artificial genome, and is allowed to evolve through time. The loss of certain sensors (for example due to damaged hardware) or other kinds of unexpected changes in the sensory inputs will be taken care of by this layer.

- Transcription layer: this layer contains the agent-based equivalent of a gene regulatory network (GRN) in that it consists of a network of regulatory agents. This layer is the middle layer in our layered approach and does not directly interface with any hardware components of the robots. The main tasks of this layer is the optimization / emergence of the robot's behavioral / movement patterns, with the biological counterpart being the production of novel functions and patterns for the actuators.

- Translation layer: this layer is the top layer in our system and interacts with the actuators of the robot. The translation layer consists of structural agents that take their information from the network of regulatory agents in the transcription layer. This layer then provides values for the actuators of the robots (in this case, the wheels of the robots). The main task in this layer is finding suitable strategies for each actuator in order to provide an adequate output pattern for the robot.

The essential aim of our agent-based system is to optimize its structure to adapt with a changing environment. In order to efficiently do so, we adopt a layered approach in our agent-based system. While a layered approach has the same components and relationships as a non-layered approach (not shown), the layered structure has a better flexibility for coping with changes than the non-layered structure, as it allows the components in each layer to be adapted independently from one another, without affecting the whole network at once. This opens up various possible approaches to deal with the information retrieved from the feedback loop, as it allows for each layer to react to this information in a specific way.

Runtime fitness in the feedback loop

In this experiment, we adopt an implicit fitness evaluation on the feedback loop and use that feedback to adapt the agents during runtime. This kind of feedback minimizes the direct reference to the patterns of robot behavior. The reason for using such an evaluation is because the output of single agent does not directly determine the behavior of the robot. Indeed, only through the cooperation of multiple agents can each agent’s behavior affect the robot’s behavior. To evolve every agent, the system cannot reward any agent’s behavior without its context. On the other hand, the pattern of robot behavior also doesn’t depend on any single agent.

In this version, the fitness will consider four variables. The first variable is the movement distance between time steps (represented as d), representing the efficiency of the two actuators. The second variable is the exploration range of the robot (represented as e), representing the robot's general performance. The third variable is the replacement ratio of agents at every time step (represented as r), which represents how quickly the inner environment changes with respect to a single agent. The fourth variable is the current total number of agents, which gives information on the current complexity of the system (represented as n). To an agent, the better the efficiency of movement, the broader the search range, the smaller the replacement ratio and the simpler the system complexity, the higher the chance for that agent to accumulate a high fitness score. The following formula shows how we deal with the fitness in our current system (i represents the current time step, p the weight parameters and T the time):

$$Fitness_i = \frac{d_i}{d_{avg}} \times p_d + \left(\frac{e_i/T_i}{e_{i-1}/T_{i-1}} + \left(\frac{n_{i-1}}{n_i} - 1\right)\right) \times p_e - \frac{r_i}{n_i} \times p_r$$

As can be seen from the formula above, fitness in our simulation reflects a comparative situation rather than an absolute value. That means the fitness score will be compared with its previous record at first and then the ratio will be regarded as the final fitness. If there is no comparable record, the fitness level will be kept neutral. The final fitness is calculated by the following formulas:

$$Fitness_{avg} = \frac{\sum \mathrm{Fitness}_{0-i}}{i}$$

$$Fitness_{final} = \frac{\mathrm{Fitness}_i}{\mathrm{Fitness}_{avg}}$$

Agents in the GRN controller

In the GRN controller of the robot, we present the different types of agents that make up our agent-based model.

1) Genome agent: A genome agent reads the relevant genes from the artificial genome. This artificial (fixed length) genome consists of a sequence of randomly generated nucleotides (A,C, G, T). The main functionalities of the genome agent are:

· Reading the artificial genome input file, if such a file is provided.

· If an artificial genome is not provided, generate a random genome.

· Look up a target gene in the genome when a binding site is provided by the signaling agent.

· Altering a particular region of the artificial genome according to environmental conditions provided by the signaling agent (for example in order to enhance gene expression).

· Mutating the artificial genome through random point mutations.

· Provide a new (and possibly altered) copy of the artificial genome at the end of a robot's lifetime.

2) Signaling agent: An signaling agent reads the sensor inputs and establishes combinations of sensor values in the robot. In other words, a single signaling agent does not correspond to a single sensor input (or a transformation thereof), but to a combination (unweight sum) of the different sensor inputs. The resulting gathered sensory information of the signaling agent is then used as a binding site during the scan of the genome in search of a fitting gene, one for each combination of sensory inputs. When such a gene is found, the indicator region of that gene is scanned to check whether that gene is a regulatory gene or an expressed gene, for which a fitting agent (i.e. either a regulatory or an structural agent) is then created by the signaling agent. The main functionalities of the signaling agent are:

· Update (i.e. read) the sensory input values of the robot at every time step.

· Every time step, the signaling agent evaluates the status of the robot (i.e. calculates the different combinations of the sensory inputs) and calculates feedback on the robot's performance.

· If the sensory information of the robot has changed (sufficiently), the signaling agent will detect different regions of the genome than in the previous cycle and create new agents (either regulatory or expressed). The old agents (of the previous cycle) are then considered at the end of their lifetime and are removed from the system.

· Optimizing the combinations of the sensory inputs. The signaling agent will calculate the adaptation value for each combination at each time step and will then try to optimize those combinations. The general procedure of this approach is as follows. This agent evaluates each sensory input combination by its adaptation value (this value is based on the feedback, the number of agents and the importance of the output for the particular combination) and checks if the combination can be adapted to the current environment. The signaling agent will delete those combinations that have a lower adaptation value than a given threshold and produce a new sensory input combination by randomly reading a special-purpose gene.

· Monitoring the gene regulatory network and ensuring the interactions of agents will not exceed the limitation. The signaling agent will check the number of agents to avoid that too many of them are created. The signaling agent will also delete those combinations of agents for which a corresponding gene cannot be found or for which the output value is too low/high.

3) Regulatory agent: A regulatory agent regulates, for example enhances or represses, a particular gene and must hence retrieve the gene it expresses. The main functionalities of the regulatory agent are:

· Finding the regulatory gene from the genome based on its corresponding signals. If there is an agent which shares the same gene in the system, the concentration degree of this existing agent will be increased. Otherwise, a new agent will be created and the new one will read the target gene to initialize itself.

· Identifying the target gene and sending an instruction to the genome agent to change the gene status (repress or enhance). When agents have been created, the position of the target gene on the genome will be recorded by its agent. All agents will have a limited life cycle in the agent-based system. When the agent dies, every agent will modify the adaptation value of its target gene according to its own feedback. This way, an agent that performs well will enhance the adaptation value of its target gene. The higher the adaptation value of a gene, the more chance this gene has to be read from the genome at a future occasion.

· Evaluating its own importance and adaptation in the system's interactions. The agent needs to evaluate its adaptive status in the system and needs to know the influence of its outputs. If the agent has more outputs than others, it will be regarded as more important in interaction and it will also has more responsibility with respect to the feedback. The adaptive status for an agent indicates how good the performance of the robot is when the agent is active. The adaptive status of an agent will be used to affect the adaptation value of its corresponding gene.

· Reading the artificial genome in order to find the gene, either regulatory or expressed, it regulates.

· Upon retrieval of the expressed gene, creating the structural agent.

4) Structural agent: An structural agent translates the encoded information of its underlying gene to an actuator. The main functionalities of the structural agent are:

· Binding with the corresponding actuators and output the value encoded in its underlying gene to an actuator.

· Identifying the target expressed gene and sending the instruction to genome agent to change the gene status (repress or enhance). The genome agent will change the binding site of the expressed gene to increase or decrease the probability that it will be read at a future occasion.

· Evaluating its own importance and adaptation in the system's interactions. The importance (score) of an structural agent is the sum of its actuators multiplied by the concentration degree of this agent. The adaptation status of an agent is used to evaluate its performance. For an agent that performs well, the agent-based system will increase its concentration degree and life time. For an agent that does not perform well, there will be a tendency for deletion of this particular agent.

· When multiple structural agents correspond to a single robot actuator, the different values in the expressed genes need to be aggregated into one output signal for the actuator.

Agent and pathway replacement

During runtime all agents and simulated pathways interact with their inner environment and connect to form an emerged dynamic structure. The creation of new agents is triggered by the constant input stimulations from the environment, while the replacements are always conducted by the inner feedback loop. A single agent replacement will be built up due to a particular signal combination from the environment. Signal pathways are initialized from the genome and are sensitive to its inner environment. Environmental changes could lead to a potential pathway being activated or the activated pathway being repressed. Agents also have their own lifetime and concentration value, something the pathways don't have. Even if the calculated fitness from the feedback loop is good, an agent will still be replaced when it has run out of lifetime or when its concentration value is too low. When this happens, the replacement will be regarded as successful with respect to the individual agent and the built up gene of that agent will be enhanced in the genome. In other words, the gene will become more competitive to be read in a certain input range. The concentration value of an agent will hence increase or decrease corresponding to fitness.

Simulation platform

All experiments were performed in the Player/Stage simulation environment[160]. Classes necessary to read, store and manipulate robot genomes were written in C++ for cooperation with the programming code in Player/Stage. The simulation map consists of a rectangular area with several obstacles and corners where the robot can become cornered or stuck. The fitness of an individual robot is a function of the amount of the environment the robot is able to explore (the more, the better). Inherent to this requirement for a high fitness value is the ability to perform obstacle avoidance. Simply calculating a measure to perform obstacle avoidance has the property those robots that simply turn in circles can also be regarded as ‘avoiding obstacles’, while it is not performing a useful task. The combination with map exploration fixes this problem.

In our experiments, we have tested the developed agent based system using a randomly generated population consisting of 50 robots. In other words, a (different) random genome was generated automatically for each of the 50 individual robots. The 50 robots were tested individually by placing each of them in a separate copy of the environment. While the starting position within the simulation map was set to be identical for each robot, the starting orientation of the robot was randomly selected to avoid that robots perform well simply because they are oriented in a direction with no obstacles.

We have used simulated e-puck robots in our experiments [161] . These robots have eight infrared (IR) proximity sensors (similar to the Khepera robots[162]) placed around the body which can be used to measure the closeness of obstacles and two stepper motors, controlling the movement of the two wheels.

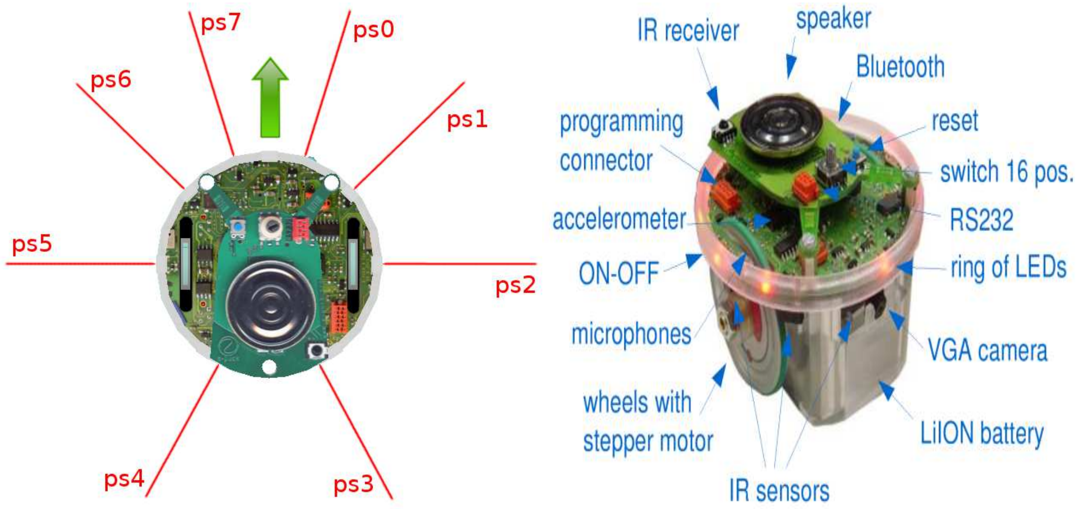


Figure 4.2 Illustration of the real e-puck robot.

The simulated robot used in the player/stage simulation is based on the robot shown above. They share similar sensors and wheel actuators.

4.1.4 Experiments and results

Resolving collisions and repetitive motion

The simulations performed using our developed agent-based system, based upon a large genome inspired by gene regulatory networks, show evidence of self-adaptive abilities (i.e. the ability to adapt itself depending on occurring problematic situations). These situations can be considered to be getting stuck in a corner, not being able to move away from a wall or even just undesired behavior in terms of achieving a decent fitness level. In order to do this, the agent-based system was provided with a feedback loop to signal potential problems with the current environment of the robot. This feedback loop ensures that the robot can adapt to new situations or environments up to a certain level. This also means that as long as the feedback loop does not signal any problems to the agent-based system, the robot's behavior remains unchanged.

An example of this type of behavior can be seen in Fig 4.3,

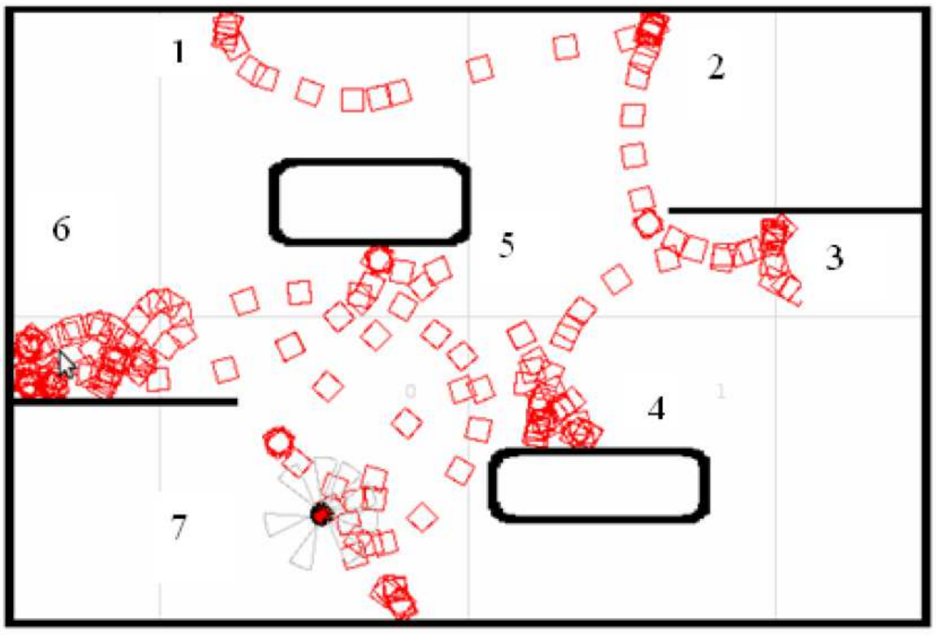


Figure 4.3 Due to the agent-based system, the robot is able to resolve difficult situations.

Starting from a fixed position near the center of the area (close to position 5), the robot encounters a series of problems, such as being stuck against a wall (positions 3 and 4) and being stuck in a corner (position 6).

Fig 4.3 illustrates how the robot smoothly adapts its behavior during the runtime of the simulation. When the robot hits an obstacle, it will adapt its behavior slightly in order to solve the problem. Should small changes in the robot's behavior not be sufficient to resolve the current problem, additional changes to its behavior will be made by the agent-based system until the robot is able to free itself. The more problematic the situation, the longer it will take for the robot to resolve the problem. This is an important aspect of the agent-based system used here, as in other approaches (see e.g. [163]for a neural network approach). The emergence of optimal robot behavior is obtained by removing robots that do not perform well from the population, across a large number of generations. This is a process which may end up taking a huge amount of time, even though bio-inspired approaches have been proposed to facilitate this process, i.e. to make the population of robot reach adequate fitness level at a faster pace. For example, Calabretta et al. have published a series of papers on the advantages of modeling gene duplications on the performance of a robot population[164-167]. A comparison between an approach with feedback enabled (such as our agent-based system) and an approach without such a feedback loop (for example, a simple artificial neural network) can be seen in Fig 4.4.

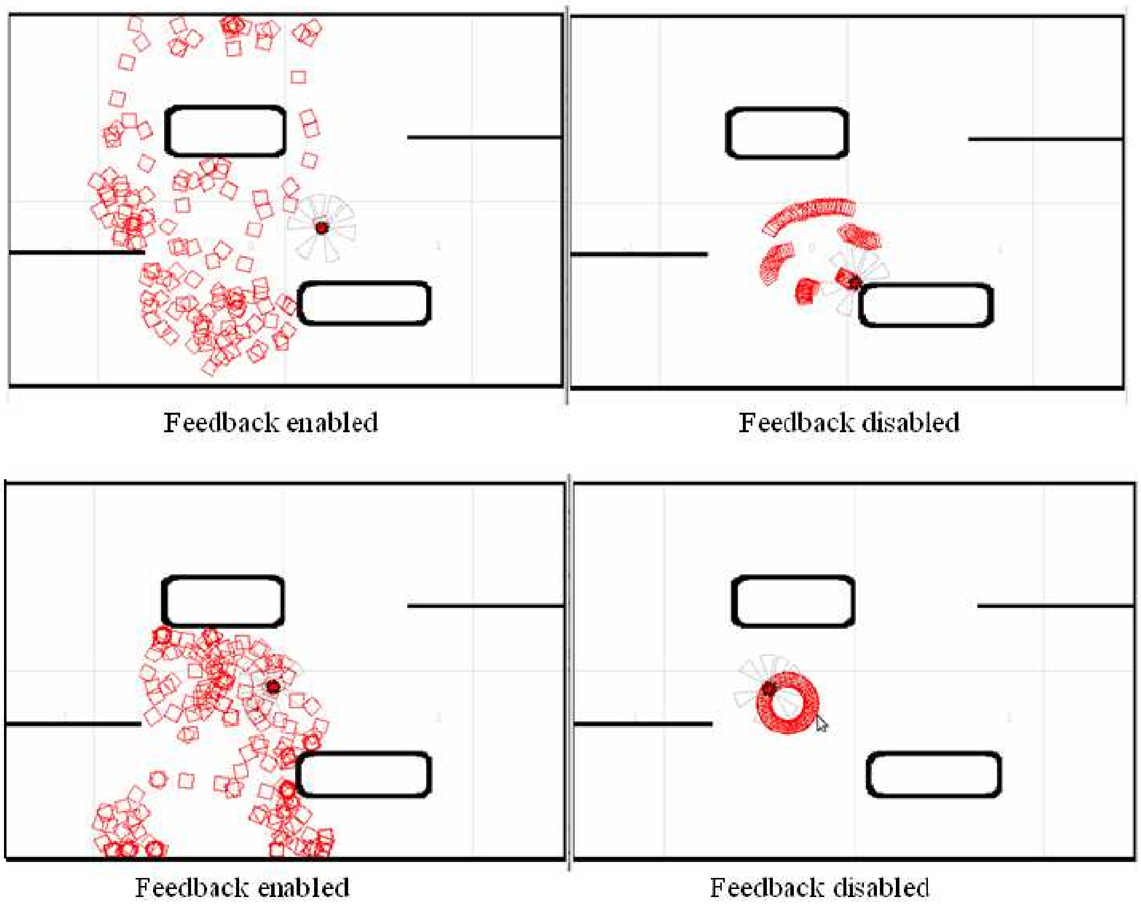


Figure 4.4 The feedback loop incorporated into the agent-based system also allows the robot to detect when it performs repetitive movements, such as just turning in circles, which may occur in the absence of such a feedback loop.

The way in which each robot's decision and sensing mechanisms works is encoded in its genome and is essentially the responsibility of the signaling agent in the agent-based system. Hence, when the robot gets stuck, the feedback loop informs the appropriate signaling agent of this, after which the current sensory information combination will be removed and a new sensory input combination will be proposed to the system by reading and decoding different genes from the genome.

Genome-dependent behavior

Since the agent-based system relies upon an artificial genome to create its various components, different artificial genomes will lead to different types of behavior (and hence a difference in performance) for different robots.

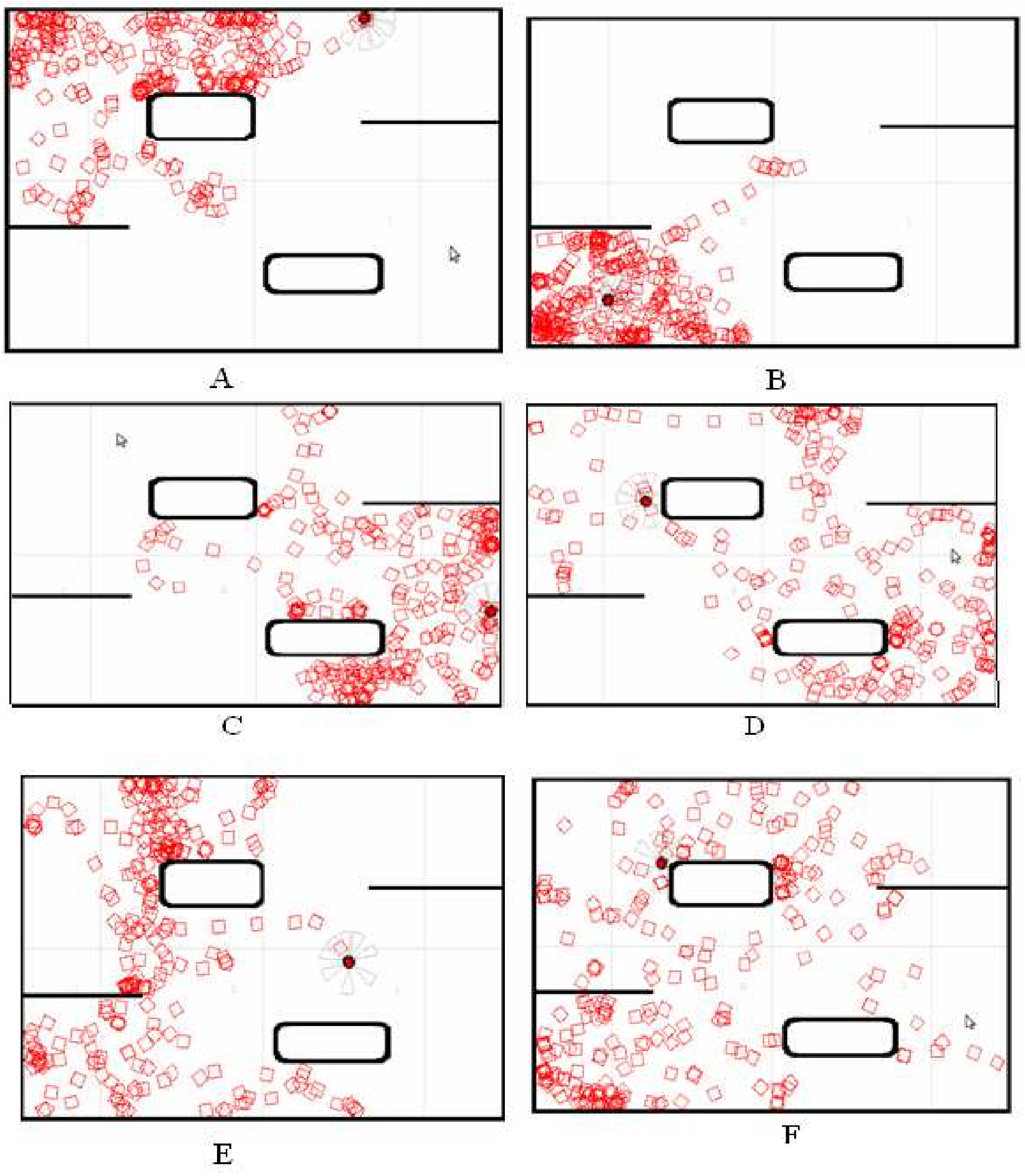


Figure 4.5 Depending on the genome of the robot, different behavior can be observed, with the feedback loop making sure that the robot does not get permanently stuck.

More complicated regions of the area (e.g. with only one possible escape direction) show a more dense trail as it takes longer for the robot to resolve the situation.

In Fig 4.5 it can clearly be seen that there are various locations in the area which are difficult for the robots to explore, such as the top-left corner (situation A), the bottom left corner (situation B; same goes for the top-right corner) and the bottom-right corner (situation C). In none of these situations does the robot remain stuck however, although it is apparent that the bottom-left corner is the most difficult part of the area to escape from, hence the large amount of time that the robot spends there. Situations D and E in Fig 4.5 show robot behavior for two artificial genomes that allow the robot to explore large portions of the area, with little time being spent stuck in a corner or against a wall. Finally, situation F in Fig 4.5 shows robot behavior for an artificial genome that allows the robot to explore the entire area, visiting all the difficult to reach (and escape) areas of the map. A given agent is able to autonomously select better-suited genes in order to achieve its goals. For example, if the behavior of a given robot in the current environment is rewarded, the corresponding agents responsible for the robots behavior will be automatically rewarded as well and the gene(s) that set those agents will also be slightly enhanced. Such enhancements can accumulate on the genome and eventually render those genes easier to read by agents, leading to next-generation agents that will tend to select better-suited genes to achieve the robots goal(s).

Agent dynamics and robot’s performance

The agents in our agent-based system can be replaced by new ones during runtime. Such replacements may result in a positive influence on the robot's behavior in terms of a robot's ability to adapt to its environment. In other words, a change in environment can mean the transition from a problem-free environment (when the robot does not encounter any obstacles or other difficulties) to a problematic environment (i.e. being stuck in a corner or against a wall). Agent replacements will hence occur most in difficult environments, driven by the data in the feedback loop, while there is no need for such replacements in problem-free environments. This way, each robot possesses a self-adaptive ability when confronting a new environment. For instance, we show in Fig 4.6 and 4.7 an example of a robot's behavior (i.e. its phenotype) and the corresponding agent dynamics within the agent-based system (i.e. its genotype). Fig4.6 shows the movement trail of the robot in the simulation map (with numbers indicating the different situations), whereas Fig 4.7 shows the corresponding agent dynamics during those time steps. As can be seen from Fig4.6 the robot encounters four difficult situations during its runtime (i.e. at situations 2, 3, 4 and 5), which results in a temporary halt in the robot's task to explore the simulation map.

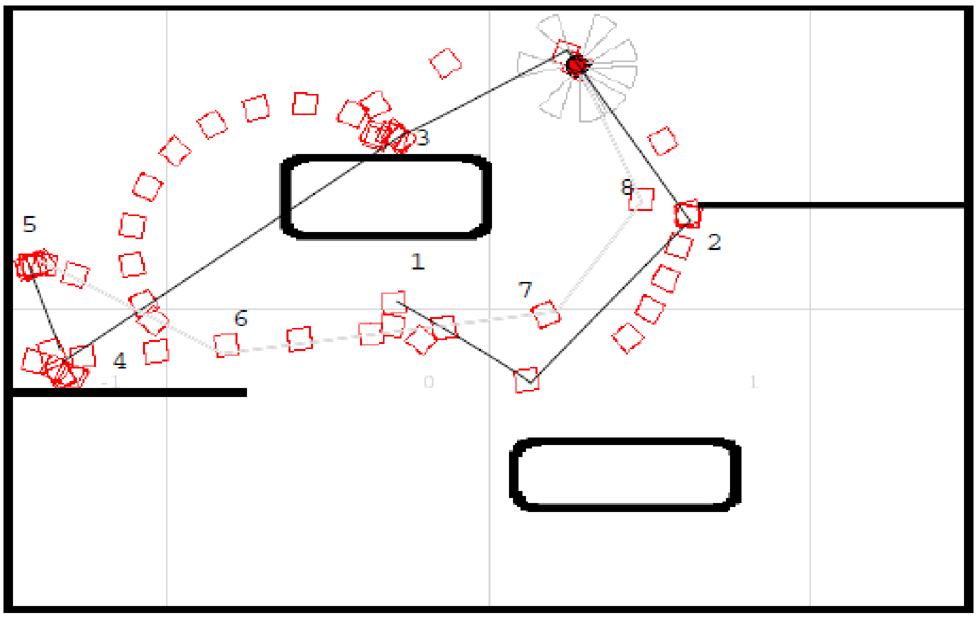


Figure 4.6 The movement trail of the robot during 200 time steps (i.e. input/output cycles).

Numbers 1 through 8 indicate (in order) the various situations the robot can be found in.

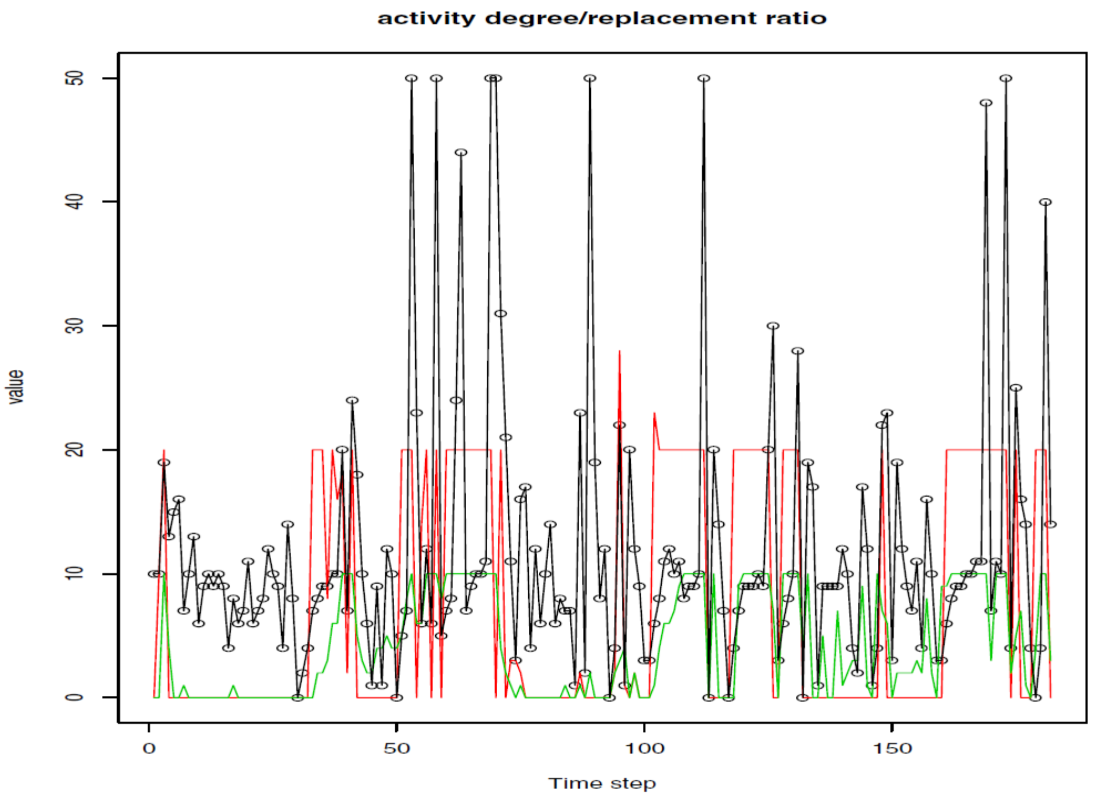


Figure 4.7 The replacement rates of agent and signal pathways are correlated with the robot's activity degree, which is the main measurement of fitness in this simulation.

The black curve represents the activity degree, while the green curve represents the signal pathway replacement rate and the red curve represents the agent replacement rate.

These problematic situations can be resolved through agent and signal pathway replacements in order to change the robot's current behavior, which will allow the robot to overcome the problem rather than to remain stuck. This will result in an increased performance of the robot, as otherwise the robot would remain stuck (see Fig 4.6). For example, the difficult situations the robot has to overcome occur after 50 time steps (situation 2 in Fig 4.6), 90 time steps (situation 3 in Fig 4.6), 120 time steps (situation 4 in Fig 4.6) and 145 time steps (situation 5 in Fig 4.6). Fig 4.7 shows a clear increase in agent and signal pathway replacement rate corresponding to these reported time steps. In other words, in order to resolve the fact that the robot is stuck in a difficult environment, agents and/or signal pathways are replaced in order to equip the robot with a suitable behavioral pattern, fit to the changed environment (i.e. corner or obstacle). Indeed, when a robot gets stuck, its ``activity degree'' (i.e. its ability to explore the area) drops to very low levels, indicating the need for a different behavior in order to be able to continue performing its task adequately. Following this drop, both the replacement rates of agents and signal pathways increase, after which the activity degree of the robot is seen to increase again.

4.1.5 Discussion and Conclusions

In this research we presented a simple version of an agent-based system aimed at controlling robot behavior. The main benefit of this system is the robot's ability to resolve problematic situations at runtime, in its goal to explore as much of a rectangular area, filled with obstacles, as possible. Currently, the area in which the robot performs its task is static, i.e. does not have any moving obstacles, and the robot's task is not overly complicated. The current set-up is however adequate to test our developed agent-based system as we aim to add more complexity to the robot's tasks and the simulation area as well. For example, it would be more realistic that a robot has a limited lifetime, depending on battery power, a scenario where the robot would not only have to explore a given area but also make sure that it doesn't run out of battery power. This is the subject of ongoing work. The structure of the artificial genome is currently a drastic simplification of the real-life workings of gene regulatory networks. Hence, the representation by the agent-based system of the gene regulatory networks in the artificial genome is oversimplified as well. The work presented in this paper however serves as a proof of principle and we aim to increase the complexity of our artificial genome to resemble biological reality more closely in future work.

We have shown the adequate performance of our developed agent-based system, which uses a bio-inspired artificial genome based upon current knowledge on the workings of gene regulatory networks. Robots equipped with our agent based system are able to find their way out of difficult situations, allowing them to continue performing their task. This is specifically due to the presence of a feedback loop in our agent-based system, which signals potential problems to the system, allowing for a solution to be found. Further, the desired behavior of the simple task in this paper (exploring a simulation map) is brought about without the need for an evolutionary strategy. However, we expect that more difficult simulation scenarios will require an evolutionary strategy in order to yield adequate results.

4.2 The self-organizing robotic organisms

This section 4.2 is partly reproduced from the following publication (in press):

Yao, Y., Marchal, K., Van de Peer, Y. (2015) Adaptive self-organizing organisms using a bio-inspired gene regulatory network controller- for the aggregation of evolutionary robots under a changing environment (Handbook of Research on Design, Control, and Modeling of Swarm Robotics)

4.2.1 Abstract

Here, we describe the biologically inspired concept of gene regulatory networks to develop a distributed swarm robot self-organization approach. In particular, we show that by using this approach, multiple swarm robots can aggregate together to form a robotic organism that can adapt its configuration as a response to a dynamically changing environment. In addition, we examined different evolutionary operators such as mutations and duplications and show that these also may have important roles in accelerating the adaptive process of evolving robotic organisms.

4.2.2 Introduction

Self-organization is a phenomenon that has been observed in disciplines as diverse as physics, molecular chemistry and biology. For example, in biological self-organizing, hundreds to even billons of homogeneous or heterogeneous cells can aggregate to form colonies or tissues. Social insects such as ants, termites, bees, or even schools of swimming fish can be regarded as self-organizing systems where coordination arises out of the local interactions between entities of an initially disordered system, which allows the system as a whole to perform more complex tasks than those performed by the individual entities [168]. In robotics, one of the most widely adopted approaches that mimic this biological behavior is swarm robots and robotic organisms [169-172]. Swarm robotics systems are self-adaptive systems in which individual swarm robots can aggregate and form a robotic organism with an emerging global behavior [173,174]. Ideally, such systems should have the potential to be adaptable by changing their configuration in different situations in an unbiased way[175]. In other words, swarm robots should decide themselves when and how to assemble, depending on environmental cues and the current configuration of the robotic organism. Configuration here refers to the topological 2D or 3D plan according to which individual swarm robots assemble into a more complex 'organism'. Ideally, such configuration of the robotic organism is the emerging result of the interaction between robots and the environment [176] and thus should not be predicated on predefined configurations that should be adopted under different predefined settings. This is difficult to achieve with robots driven by a global and explicit algorithm in which the configuration of the robots is predefined, the reasons being that an explicit algorithm cannot easily alter predefined topologies according to environment changes that occur during the aggregation process and environmental changes can be too many to be all considered in the program. Furthermore, since environments may be unknown to the developer it is very difficult to predefine a suitable adaptive configuration.

Previously, we have explored the adaptive potential of simulated robots that contain a genomic encoding of a gene regulatory network (GRN) [117]. In this study, we build upon this GRN based model to develop self-organization robotic organisms that can react to a changing environment. The developmental process of the organism to be assembled is driven by the current environmental situation. This way, the configuration of the robotic organism emerges from the interactions between individual robots and the environment. To test our approach using the concept of a gene regulatory network as implemented and described previously [117], we show through simulations that newly assembled robotic organisms can flexibly adapt to a changing environment by each time developing the most optimal configuration.

4.2.3 Materials and Methods

Aggregation model

In our approach to evolve and develop robotic organisms, we have implemented artificial gene regulatory networks, as described previously, that control the aggregation signals of the swarm robots. In the first step, we distinguish between swarm robots that have been selected for aggregation and those that have not (referred to as free living swarm robots, see further). Swarm robots have the ability to communicate with each other and can receive a message to aggregate from robotic organisms (this communication ability is supported by the LED-based communication of the robot). Our GRN-based controller is only activated in those robots that engage in the aggregation process. Depending on the environmental input, each robot will then activate a GRN encoded by its genome. The resulting activated genes (determined by their 'gene expression' values) will determine the status and the functionality of each single robot. As the genome evolves during adaptation, the activated GRNs of the different robots will change concomitantly during its adaptive process. Based on the signals of its activated GRN and its prespecified functionality, the robot will know how to aggregate and interact with other robots and to identify its position in the robotic organism (see further). When all robots in the organism eventually have identified their position and role, the robotic organism will be formed. Through local interactions and the collective behavior of swarm robots to shape the complex self-organizing patterns (i.e. the formation of the larger robotic aggregated organism) is the approach of our aggregation model and this method have been proved as a very efficient way in many recent studies[177-179]. In this study, we have adopted such an interaction driven aggregation method with an artificial gene regulatory network. The function of the artificial gene regulatory network is to connect the environmental conditions with the developmental aggregation process. As a result, artificial evolution of the genome and GRNs of robots could optimize the aggregation based on the different environmental backgrounds.

Swarm robotic platforms and the GRN controller

As we focus here only on aggregation, we have not explicitly tested or implemented controllers for other kinds of behavior control. To separate the aggregation control from other behavioral control of the robot, we perform our experiments in two ways. First, we use the 2D simplified simulation to simulate the aggregation of the swarm robots. This way, other behavioral controls for aggregation like communication, movement or alignment are ignored and the robot controller only decides which robots will aggregate with each other and on the position of each robot in the multiple robot organism. Furthermore, we test our controller on the well-developed Symbrion robot[172] (see Fig 4.8) and run the same simulation in the robot3d simulator(simulator developed based on Delta3d [180])(Fig 4.9). This way, we could use the developed controllers in the Symbrion robots to control other relevant behaviors and perform aggregation in a real environment. With the real robot hardware and robot3D simulator, we test the feasibility of our approach in practical applications. The Symbrion robot is the basic component of a self-evolving swarm robotic platform which is developed in the framework of the EU project Symbrion [67] and the project also developed a corresponding robot simulator called robot3D, reflecting the physical features of the Symbrion robot. By adopting the open-source game engine Delta3D, robot3D can provide a realistic 3D physical environment for the robotic simulation. However, it also has a limitation at the scale of the simulation due to the high computational resource requirement. Due to these limitations, in practice we could not run larger scale robot aggregation on the real hardware or robot3D simulator within a reasonable time so the real symbrion robot and robot3D simulator are only used to examine the feasibility and compatibility of our approach. The evolvability and adaptability of our controller is therefore tested on the simplified 2D simulation.

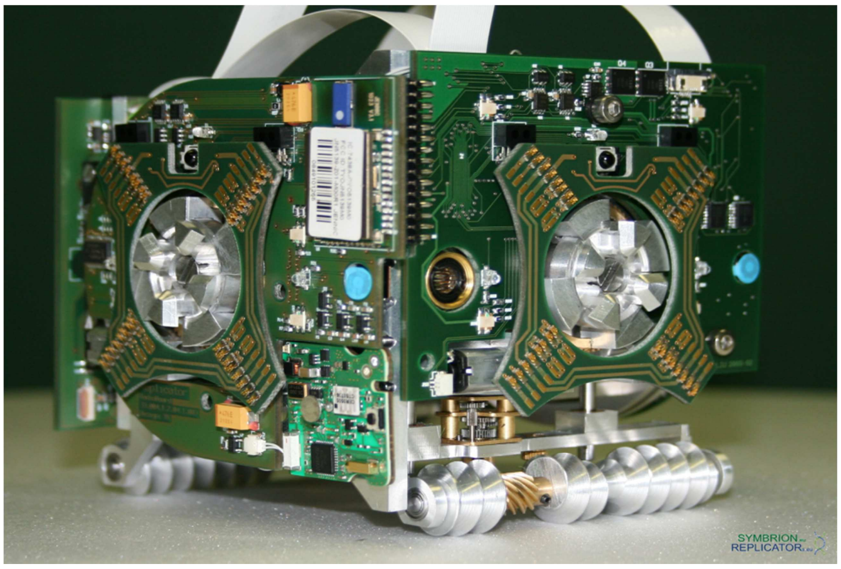


Figure 4.8 Prototype of the Symbrion robot.

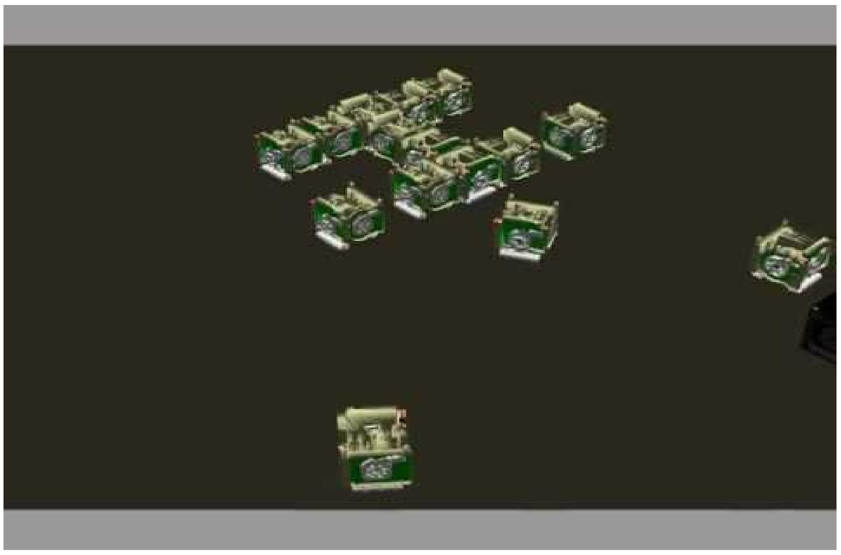


Figure 4.9 Screenshot for the robot3D simulation

Our proposed platform is the Symbrion robot, which has a 32-bits Blackfin microprocessor[181] and 64MB memory. In this limited amount of memory, it needs to store several other controllers for other behavioral control so the available memory space for our controller is not much. Therefore, to reduce the occupied memory space of our GRN controller in the real robotic systems, we have simplified the GRN controller instead of using the previous model described higher (using agent-based systems). Here, gene products are only effective for one time step, rather than allowing them to be dynamically altered during several time steps, as in our previous work[117]. In this GRN based model, gene regulation is optimized by the continuous feedback of the robot's performance. Positive feedback will enhance the gene expression level, while negative feedback will repress the gene expression level. Positive feedback and improved adaptability (basically measured as success in finding energy sources, see further) will thus influence particular 'gene expression' patterns in subsequent time steps. In other words, if a robotic organism performs well, feedback loops will make sure the same genes in the genomes of its constituting robots will be expressed at higher levels, while if a robotic organism performs poorly, negative feedback will decrease the expression level of certain genes in the robots. In addition, the genes on the genome can, at each time step, undergo substitutions and duplication [182]. The rate of mutations in a gene depends on the (history of the) feedback as well. Under more positive feedback, the rate for mutations will be lower (mimicking purifying selection, i.e. a gene that performs well should not be changed too much), while the duplication rate will increase (if a gene performs well, it might be good to increase dosage of the gene).The implementation of such bio-inspired evolutionary processes tends to protect the more 'adaptive' genes while accelerating variation in the rest of the genome [117,183].

Multiple robot organism development

In our simulation, multiple swarm robots can aggregate to develop a robotic organism. Every swarm robot has four sides (based on the Symbrion robots) that are available for docking with another robot. Fig 4.10 shows a simple example of the aggregation possibilities for a single robot. The formation of the robotic organism will start from a swarm robot population (Fig 4.11).

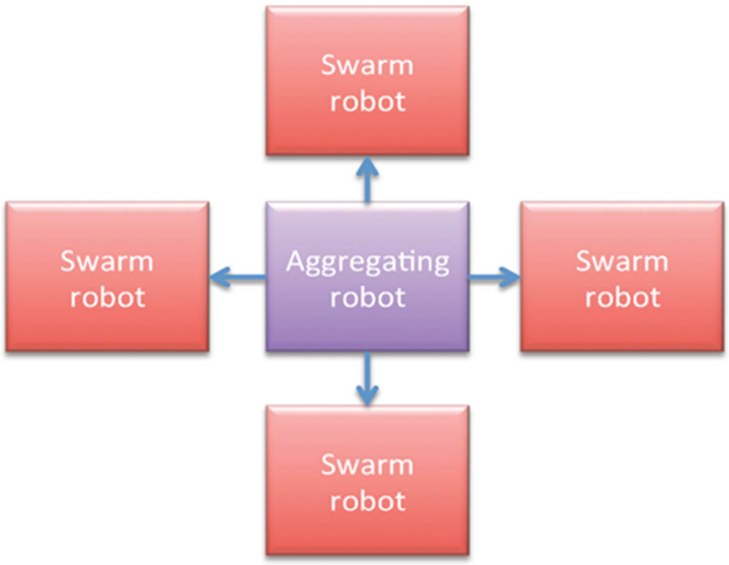


Figure 4.10 Aggregation based on one single robot.

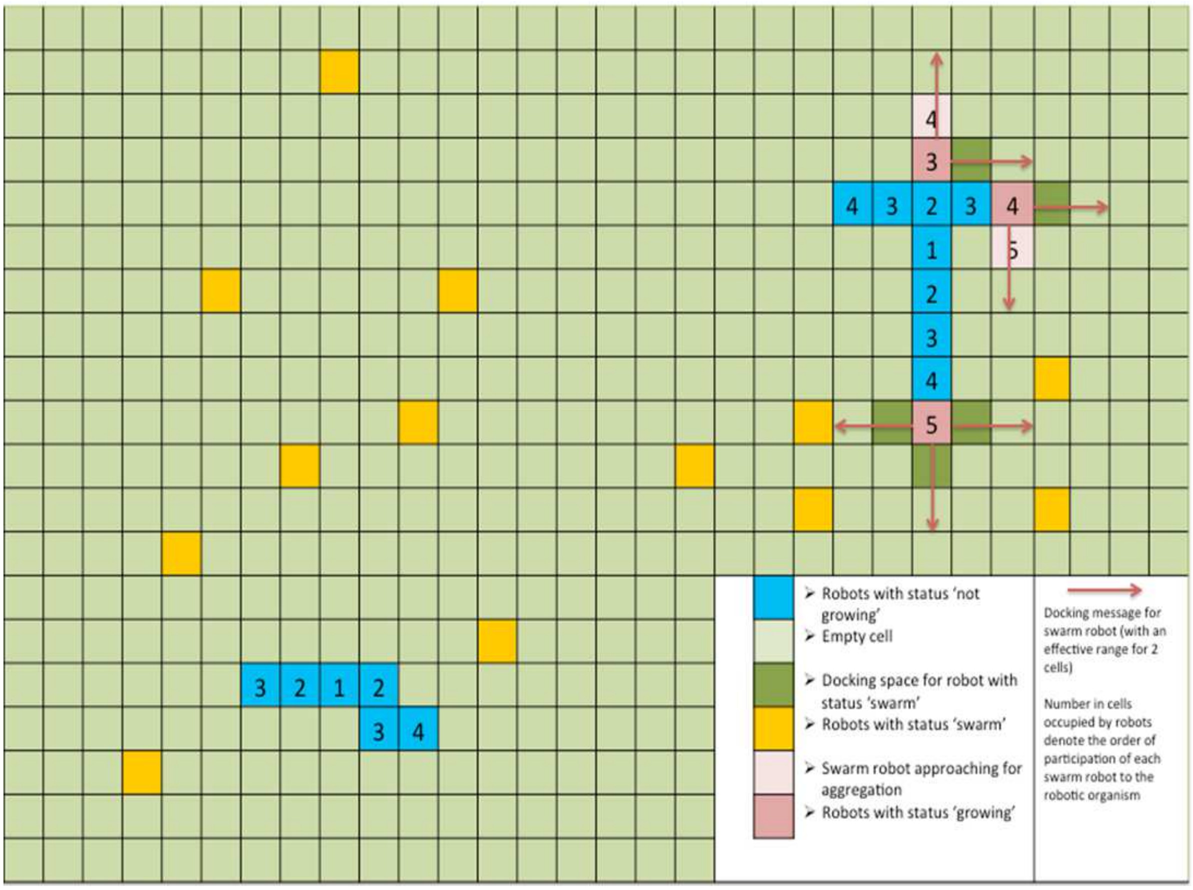

4
3
4 3 2 3 4
1 5
2
3
4
5
3 2 1 2
3 4
Robots with status 'not growing'
Empty cell
Docking space for robot with status 'swarm'
Robots with status 'swarm'
Swarm robot approaching for aggregation
Robots with status 'growing'
Docking message for swarm robot (with an effective range for 2 cells)
Number in cells occupied by robots denote the order of participation of each swarm robot to the robotic organism

Figure 4.11 Example of developing robot organisms during the simulation.

Each square on the map represents a cell in the grid that can contain one robot. Every non-green colored square represents one robot in. Different colors represent robots with a different status. Numbers in squares represent the order in which the robots have joined the robotic organism. Energy packets are not shown.

In the population, after a certain number of time steps, the system will randomly select some robots (i.e. Thus selected for aggregation) and give them a random genome. This corresponds to the activation of our GRN based controller that drives the aggregation process by determining their status. The decision for the current robot status (i.e. growing/aggregating or not) is based on the environmental inputs of that robot. In our simulation, the environmental inputs for one individual robot consist of six different values. These are:

(1) The number of surrounding robots (more surrounding robots will promote the decision to generate robotic organisms consisting of more individuals)

(2) The energy level of the robot

(3) The rank order in which the robot was added to the robotic organism: the joining of the robotic organism by individual swarm robots can be represented by a tree structure. The rank gives information on when the robot was added to the robotic organism.

(4) The size of the robotic organism, which is represented by the number of the individual robots in the robotic organism. Robotic organisms of different sizes can have different growing strategies.

Always extending the size of the body might not prove the most adaptive. This information input should allow the robot to evolve an ideal size in a particular environment

(5) The time the robot has waited before docking with other robots

(6) The increasing or decreasing rate of the energy level of the robot

Depending on the robot's status (growing/aggregating or not) an instruction will be generated on how to interact with other robots. According to its status, each robot will verify its corresponding interaction functionalities during the developmental process. If the status of the robot is 'growing', the embedded GRN will decide the corresponding docking strategy of that robot. For instance, it will need to determine which side(s) of the robot needs to dock with another robot, after which the docking side(s) will send docking messages to attract other robots to join the robotic organism. In other words, for every robot for which the status is 'growing', the GRN controller will determine a local configuration (fig 4.10) and will develop the corresponding functionality of the robot. The swarm robots that have not obtained a genome and are controlled by a simple movement program will search randomly for the growing robots until they receive a message to dock or aggregate with other robots.

A new swarm robot that joins the robotic organisms will inherit the genome from the robot that directly recruits it and will then develop a GRN itself. For these newly recruited robots, the status 'growing' is the default, but not necessarily the final status. For all robots that have the status 'growing', at every time step, the local GRN will re-decide the status of the robot and check whether the new status is still 'growing' or has changed its status to 'stop growing'. The status of the robot will always change to 'stop growing' when the robot has reached its local configuration plan by aggregating with other robots at all proposed docking sides. The 'stop growing' status will stop all aggregating functions of the individual robots. In addition, when and only when the status has changed to 'stop growing', the local GRN controller will be shut down and the robot's status will no longer be able to change until the end of this robotic organism's lifecycle. Through this morphogenesis process, each robot in the organism will tend to achieve its local topology by aggregating with other swarm robots. When all included robots have achieved their own proposed local topology, this process will eventually lead to a self-assembled organism. Another characteristic of this process is that the robotic organism can flexibly adapt to the environment during development. Unexpected environmental changes (like robots breaking down or not sufficient robots having been discovered for docking) during development will not terminate development of the robotic organism but will be overcome by developing a novel configuration plan.

4.2.4 Results

We developed a particular simulation scenario to assess to what extent our GRN-based controller would allow self-assembling organisms to flexibly develop and rethink their configuration plan under changing environments. More specifically, we simulated a 2D grid in which energy sources are provided according to a certain 2D pattern (topology). Robots that aggregate into a configuration that follows the food distribution topology will be able to consume more energy and obtain higher fitness levels (see more details in appendix D). Robots thus should learn to assemble according to a specific topology. To mimic fluctuating conditions, the food distribution topology was altered with a predefined frequency.

Concretely, in our simulation, during every time step, the individual robots in the robotic organism will consume a certain amount of energy. As a result, the robotic organism has to find enough energy otherwise it will disassemble. During every time step, the simulation will distribute energy packets over the simulation matrix (a 90 by 90 cell grid) according to a certain pattern. The robotic organism can detect such energy packets and harvest their energy content. Absorbed energy will be divided over all robots in the robotic organism. When the energy of the robotic organism cannot satisfy the energy consumption of all robots in the organism, the organism will disassemble and all robots will be re-initialized as independent swarm robots (Fig 4.10). Contrarily, if the energy of the robotic organism reaches a predefined maximum level, the genome of the organism might get replicated(with a 1% probability) and the replicated genome will be transferred to a randomly selected swarm robot, which can then start developing a new robotic organism. In our set up, a robotic organism can only grow but cannot move during the simulation. As a result, the configuration of the robotic organism is the sole factor determining the efficiency of energy collection. Furthermore, every particular topology has a different possibility to overlap with each certain energy distribution pattern on the simulated map and the more overlap between the robotic organism and the energy distribution pattern, the more energy will be collected.

For investigating the adaptability of the robotic organisms under a changing environment, we frequently change the pattern of the energy packets distribution, which simulates environmental change (we use four different distribution patterns that are altered every 200 time steps). The average energy level of the robotic organism and the number of robotic organisms that survive during the simulation are used as an indication of the average adaptability of the robotic organisms in a changing environment. Hereby we assume that more 'adaptive' organisms will, on average, have

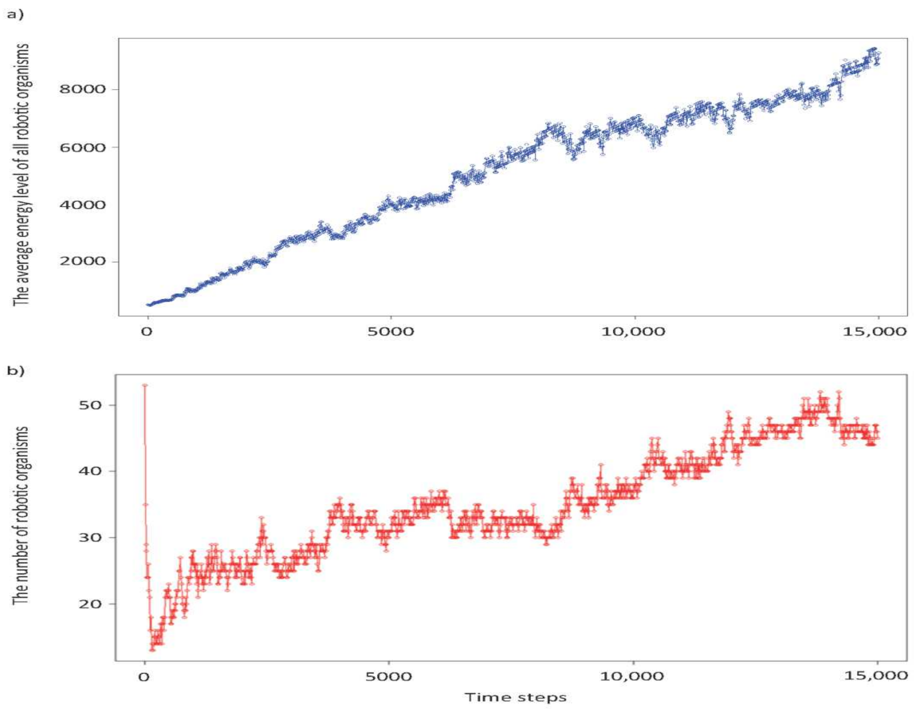


Figure 4.12 a) Evolution of the average energy level of all robotic organisms during 15,000 time steps. b) Evolution of the number of robotic organisms over 15,000 time steps.

higher energy levels, while more ‘adaptive’ genomes will lead to more organisms that survive.

Therefore, we evaluated the average energy level of all robotic organisms as well as the number of surviving organisms in our simulation after every ten time steps. All simulations were performed with 300 robots. At the start of the simulation, 200 robots act as swarm robots that have a random controller that allows the robot to move around and turn to avoid obstructions. The remaining 100 robots are defined as ‘organism robots’ that have a genome and are ready to develop a robotic organism by aggregating with the swarm robots. ‘Organism robots’ cannot move. Through time, the number of both kinds of robots will dynamically change but the total number of the robots always amounts to 300. Indeed, some swarm robots will join the organism robots and become part of the robotic organisms, while some robotic organisms will disassemble and become swarm robots again.

The simulation starts with 100 individual robots selected for aggregation. However, the actual number lies around 50 because about 50% of the initial robots will also actually develop into a robotic organism. In turn, most of these initial robotic organisms cannot successfully survive in the simulated environment, which explains the drop in the number of robots early in the simulation. Only when the robotic organisms become more adapted, their number steadily increases.

In Fig 4.12a, we show the average energy level of all robotic organisms during runtime in one simulation experiment, while Fig 4.12b shows the evolution of the number of surviving robotic organisms during runtime. In this simulation, we have set the default mutation rate on the artificial genome as 5*10-6 (changes per site in the genome per time step; see further). The fluctuations that can be observed in Fig 4.12a and b are caused by the changes of the environment (determined by the topology of the energy sources but also by the number of swarming robots that can possibly recruited to dock). However, in general, we can see that there is a continuous increase in the average adaptability of the organisms during runtime and the robotic organisms develop more adaptive configurations in a frequently changing environment, amongst other things because the optimal behavior under a certain condition is getting encoded in the genome of the robot.

To test the added value of having a mutation rate that is varied according to the fitness function towards adaptability, we have compared results from simulations obtained with robots that either have a fixed non variable default mutation rate versus robots that have a variable but different mutation rate (Fig 4.13).

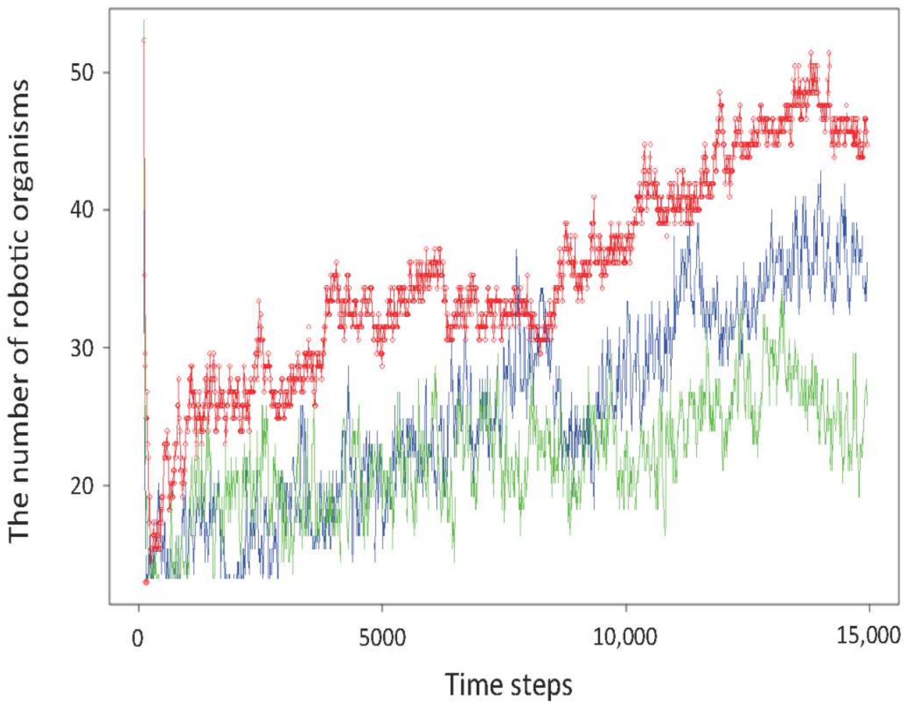


*Figure.4.13 Evolution of the number of robotic organisms as a function of different mutation rates. The red curve represents the increase in the number of robotic organisms when the genome has a default mutation rate of 5*10-6 but different regions in the genome evolve at different rates (from 3*10-6 to 9*10-6), while the green curve represents a mutation rate of 3*10-6 and the blue curve represents a mutation rate of 9*10-6.*

As can be observed in Fig 4.13, a genome with variable mutation rates will lead to better adaptation than a single mutation rate applied to the whole genome. This result also shows that too many mutations will damage the adaptability by destroying the adaptive genetic features on the genome and GRN, but on the other hand, too few mutations will not bring about sufficient variation to adapt fast enough to a changing environment. Similarly, We also compared simulations obtained with robots with or without the gene duplication as a mutational event. Fig 4.14 shows that organisms have a better chance of survival when gene duplication is allowed, which implies that also evolutionary processes such as gene duplication can indeed improve the average adaptability of the robotic organisms.

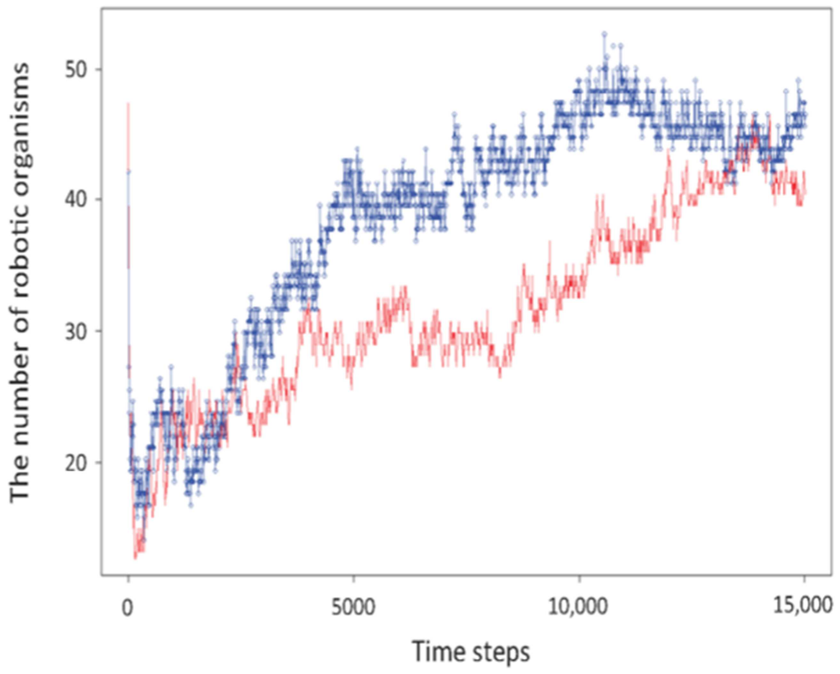


Figure 4.14 Evolution of the number of robotic organisms during 15,000 time steps. The red curve represents the evolution of the number of robotic organisms when gene duplication is not invoked; blue represents the result when gene duplication has been turned on.

4.2.5 Conclusions

In this research, we developed a distributed swarm robot self-assembling approach, using the biologically inspired concept of gene regulatory networks. The 2D Simulation experiments show that, by using this approach, multiple swarm robots can aggregate together to form a robotic organism that can adapt its configuration as a response to a dynamically changing environment. In addition, we examined different evolutionary operators such as mutations and duplications and show that these may have important roles in accelerating the adaptive process. On the other hand, we design and tested our approach based on the real Symbrion robot. Due to the limitation of resource, we only tested our controller with the single Symbrion robot for compatibility and performed the swarm robots simulation with a limited scale on robot3D simulator. (see Fig 4.15) These experiments proved the feasibility of our approach in the practical environment.

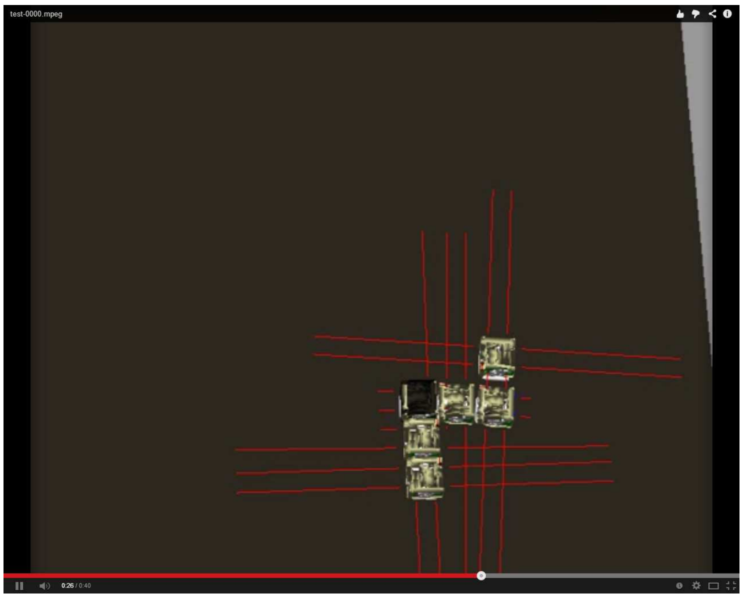


Figure 4.15 Screenshot for the aggregation in robot3D.

4.3 Surviving scenario in an artificial ecosystem

This section 4.3 is partly reproduced from the following publication:

Yao, Y., Marchal, K., Van de Peer, Y. (2014) Improving the adaptability of simulated evolutionary swarm robots in dynamically changing environments. PLOS One 9,e90695.

4.3.1 Abstract

One of the important challenges in the field of evolutionary robotics is the development of systems that can adapt to a changing environment. However, the ability to adapt to unknown and fluctuating environments is not straightforward. Here, we explore the adaptive potential of simulated swarm robots that contain a genomic encoding of a bio-inspired gene regulatory network (GRN). An artificial genome is combined with a flexible agent-based system, representing the activated part of the regulatory network that transduces environmental cues into phenotypic behavior. Using an artificial life simulation framework that mimics a dynamically changing environment, we show that separating the static from the conditionally active part of the network contributes to a better adaptive behavior. Furthermore, in contrast with most hitherto developed ANN-based systems that need to re-optimize their complete controller network from scratch each time they are subjected to novel conditions, our system uses its genome to store GRNs whose performance was optimized under a particular environmental condition for a sufficiently long time. When subjected to a new environment, the previous condition-specific GRN might become inactivated, but remains present. This ability to store 'good behavior' and to disconnect it from the novel rewiring that is essential under a new condition allows faster re-adaptation if any of the previously observed environmental conditions is

reencountered. As we show here, applying these evolutionary-based principles leads to accelerated and improved adaptive evolution in a non-stable environment.

4.3.2 Introduction

An important goal in evolutionary robotics is the development of systems that show self-adaptation in dynamically changing environments[184,185]. Searching for the 'fittest phenotype' is only one aspect of the self-adaptive behavior of such so-called complex adaptive systems (CASs), because under a dynamically changing environment, a solution that is optimal at a certain time might be different from an optimal solution at a later time. A truly self-adaptive system thus should not only reach higher performance in one particular environment, but should also evolve a better self-innovating ability that allows it to survive under different and changing conditions. This requires the ability to learn from past experiences, because although environmental changes are unpredictable, they are likely to reoccur.

Being naturally occurring examples of complex adaptive systems, biological systems provide an important source of inspiration [17,75,186,187]. The molecular mechanisms underlying the adaptability of biological systems are Gene Regulatory Networks (GRNs), which are composed of interacting genetic entities such as genes and proteins[138,188,189]. These networks transduce signals rising from environmental cues into a proper phenotypic behavior that allows the organism to flexibly respond to environmental changes. The signaling networks active in a cell are the result of an underlying genetic encoding, provided by the genome. Evolutionary processes acting on this genome gradually can lead to novel emerging circuits (evolutionary network rewiring).

Several bio-inspired systems have been developed that use an artificial genome (AG) and a corresponding controller, usually a network structure represented by an Artificial Neural Network (ANN)[72,190]. Here, a distinction can be made between systems that rely on a direct versus an indirect encoding. Systems that make use of direct coding use an ANN network design with an a-priory defined structure that directly determines the robots' phenotype. Such systems are generally well suited to efficiently evolve an optimal behavior towards a particular predefined task because they have very good learning abilities [72]. Systems that use indirect coding do not impose a predefined network structure, but only predefine rules. For instance, a 'gene' will define a node in the ANN, but this node will find and connect with other nodes in the ANN based on the given conditions. The ultimate structure of the network will therefore develop itself, according to the predefined rules. Compared to a system that makes use of a direct coding scheme, one that uses indirect coding in general allows for a more compact and flexible encoding of the genome and its corresponding GRN, mainly because not all details of the network structure need to be predefined and the GRN structure can evolve during the developmental process[191,192]. Such indirect coding approach is therefore more suitable to develop self-adaptive systems. Recent indirect coding approaches encode their 'rules' with a more biologically realistic version of an AG that mimics features of real biological genomes, for instance by means of mimicking an encoding of transcriptional interactions between TFs and their targets[1,71,103,193,194].

However, most earlier implementations have in common that, irrespective of their structure and implementation specificities, the evaluation of fitness or performance acts directly on the network controller by either affecting its structure or the weights of its interactions whereas the AG serves as nothing more than a convenient encoding of the GRN on which the evolutionary algorithms are applied. In contrast with real biological systems, most of these previously developed approaches do not allow for an uncoupling between the genomic encoding and the part of the genome that is activated in a condition-dependent network structure. In real biological systems, this uncoupling is achieved through different regulatory mechanisms. Condition-dependent activation of genes is, for instance, mediated through transcriptional regulation. Upon certain environmental cues, only part of the genome is translated into an active network. Short-term environmental feedback can then be achieved by post-transcriptional or post-translational modification of this activated part of the network, whereas long-term adaptation is largely the result of selection acting at the level of the genome.

In this study, we developed a self-adaptive system, which combines a 'bio-inspired' artificial genome with agent-based modeling (further generally referred to as our GRN-based controller) to mimic the condition-dependent way in which only part of the genome is activated following the interaction between the robot and its environment. Using a simulated dynamically changing environment, we demonstrate that the condition-dependent activation of the GRN and its uncoupling from the genomic encoding increases the potential to evolve and adapt in a non-stable environment.

4.3.3 Material and Methods

Design of the bio-inspired GRN based controller

We assume that the genomic encoding of the cellular regulatory network and the way this encoding is translated into an activated GRN is a feature of natural systems that is key to flexible and robust adaptation. Fig 2.4 provides a general overview of our framework. To implement the uncoupling between the genomic encoding and the part of the network that is activated in a condition-dependent way, our GRN-based controller consists of two distinct encodings of the GRN (see the following Materials and Methods sections for an extensive description). The 'core GRN' is encoded by the AG that defines the genes and all their possible interactions. In this AG, genes are not pre-specified, but identified in a randomly created string of digits. Potential interactions between genes are encoded in this genome by mimicking the encoding of a transcriptional network (see more details on section 2.2 in chapter 2). Although this AG encoded GRN defines all possible interactions between genes, the set of interactions that will be activated is condition-dependent.

The condition-dependent instantiation of the core GRN is mimicked by an agent-based system. Agents can be considered as the translation products of the corresponding genes in the AG. For each gene type, a matching agent type has been defined (see more details on section 2.3 in chapter 2). Which part of the AG will be translated into the agent-based instantiation of the GRN depends on the encoding of the interactions in the AG: upon a certain environmental cue, a sensory agent will activate a regulatory or structural agent, according to the interaction rules that are currently present in the AG. Once this agent is activated, in turn, this agent can activate another agent following the interaction rules in the AG and so on. The action of the sensory and regulatory agents thus mimics the way biological systems integrate environmental stimuli and pass them to the regulatory network. Structural agents transduce the signals perceived from the network into a pre-specified phenotypic behavior, such as moving, docking, etc.

Rather than relying on a pre-programmed static GRN, defined by the AG [195], the GRN and its genomic encoding will evolve through the effect of evolutionary forces such as mutations and duplications acting on the genome. Because the long time scale over which newly evolved strains originate through mere Darwinian evolution in biological systems is very impractical in evolutionary robotics, we increased the adaptive potential of our robots by also allowing for a direct feedback from the environment on the evolvability of the GRN. Agents are central in this feedback mechanism (through their adaptability value): upon increasing fitness values, agents will be able to extend their own life time (mimicking higher protein levels), allowing to directly influence the active part of the GRN. In parallel, agents will also act at the level of the genomic encoding of the GRN, e.g. by lowering the mutation rate of their respective genes, using a gene specific evolution model. This localized feedback, both at the level of individual genes and agents, allows introducing the flexibility and robustness, characteristic of complex adaptive systems.

Implementation of the GRN based controller

The GRN-based controller actually consists of two separate layers: a bio-inspired AG and an agent-based layer. The AG is based on the model of Reil [121]. For a detailed description of the genome structure, we refer to chapter 2. Key to our model is the explicit distinction between signaling, regulatory and structural genes, which all have the same basic structure but differ in their 'content region', which specifies their functionalities. For signaling genes, the content region encodes a potential ANN structure that receives and integrates signals sensed by the robot, while for regulatory genes the content region defines the connectivity of the regulatory network, i.e. for each regulator it defines which targets the regulator can potentially interact with and the mode and extent to which the regulator can activate its targets. For structural genes, the content region defines the robot's actuators on which the structural gene can potentially act. All functions and interactions of the genes encoded in the AG are referred to as 'potential' because they will only become activated upon the translation of the gene into a corresponding agent. The bio-inspired AG thus encodes the core GRN (the full regulatory network or entire collection of genes and all its possible interactions). The core GRN is an emergent system that changes over time by the evolutionary forces acting at the level of the genome. The total genome size consists of 10 chromosomes of 10,000 characters.

The second layer consists of an agent-based system that represents the condition-dependent instantiation of the core GRN (see Fig 2.4). Three types of agents have been defined, each corresponding to a specific gene type. Agents can be seen as the translation product of the genes. The agents that correspond to the gene type execute the action defined by the gene type: signaling agents include an embedded ANN, which reads the sensor input values and establishes combinations of sensor values in the (simulated) robot and channel the integrated sensor signals to the GRN by converting them into a 'sensed value'. This 'sensed' value is used to activate genes in the network.

Regulatory agents correspond to regulatory genes, which mediate signal transduction in the network by activating or repressing other regulatory or structural agents according to rules that are defined in the AG. A structural agent will translate the encoded information of a structural gene to an output parameter, which drives the actual actuator (e.g. wheel) of the robot. Each actuator usually receives many parameter values from different structural agents and will average these into one final value that will then be used as the control parameter (output value) for this particular actuator (see chapter 3 for the section 'The common functionalities all robotic organisms').

If a gene is translated into an agent, the 'concentration' of this agent depends on the expression level of the gene (which is determined by the rules encoded in the AG). In general, the higher the concentration of the agent, the higher the influence of the agent on the final output. Once translated, the concentration of the agent decays with time, mimicking protein degradation. If the concentration of the agent drops below a pre-set minimal level, the agent will be deleted. The change in concentration of an agent is determined by a default decay rate and the so-called adaptability value (AV) of the agent (see chapter 2 for 'The adaptation on the individual gene'). Adaptability values, which express the 'added value of the agents' presence' for the phenotype, depend on the current fitness value. During its lifetime, the agents' concentration and survival time will increase with an enhanced adaptability value. Adaptability values of agents are thus key towards incorporating feedback from the environment.

Mutational events acting at the level of the artificial genome

As evolutionary forces acting on the AG, we implemented both substitutions and duplications (see chapter 2 section 2.2.1). The mutation model in our system follows the adaptive mutation model, described earlier [196]. In general, the intergenic part of the genome has a higher mutation rate than the 'coding' part. Also signaling genes have lower mutation rates than other genes, to guarantee that the environmental signals perceived by the robot remain relatively stable during a minimal time span. The mutation and duplication rates are gene specific and are dynamically determined by the fitness of the system. Genes with high expression levels are assumed to be under selection pressure. Therefore, the mutation rate of those genes will be lowered, mimicking the long-term effect of natural evolution in which genes that are under selection tend to be more conserved, or will be preferentially duplicated.

Simulation framework and scenarios

We have used artificial life simulation[57,143,197,198] to see how our GRN-based simulated swarm robots perform in a changing environment. In our simulations, every robot has seven different functionalities, each of which comes with a different energy cost and energy consumption style (see detail in chapter 3: table 2 and 3). The total energy consumption for one robot during one time step depends on three factors, namely 1) a basic energy consumption required for each time step, 2) the energy consumption for performing certain functionalities, and 3) extra energy consumption for aggregation, if this takes place. The robots live in a two-dimensional 90 by 90 matrix or grid in which a number of energy sources (e.g. food) are distributed. During every time step, the robots will sense the number of surrounding robots and food sources, after which the robot will determine its next action based on its input and GRN controller values. As previously stated, selection and fitness of the robots are all based on energy (in the form of food sources). Several types of food sources exist that differ from each other in the minimal amount of energy required to access the food source (see Table 3). Robots can have different energy consumption styles, each of which comes at its own cost. For instance, food sources of Type 3 require a minimal energy level that is higher than the maximal energy level a single robot can possess. These food sources are therefore only available to robots that have aggregated with other robots. At the same time, maintaining the aggregation with other robots will cost extra energy, but comes at the benefit of being able to acquire more costly food. Such complex functions allow robots to explore more complex behavior[199,200]. If a robot does not have enough energy to cover its basic living energy consumption, it will be regarded as dead and removed from the simulation. Depending on the experimental set up, different simulations were performed. The details of the simulation parameters can be found in chapter 3. Note that in the simulations, the distribution of food (energy) not only depends on a random distribution function, but also on the interaction of the robots with their environment.

The different experiments were as follows:

Four different experimental designs were used to test different aspects of the robots. For each set up, 50 simulations were obtained.

Experiment 1: Comparing the adaptive behaviour of ANN and GRN-based robots. Here we run the simulations using the parameter setting described above for both the GRN and ANN based robots. Food sources were randomly initialized. Simulations were run for 4000 time steps. When the number of robots in the population drops below 100, food resources are initialised again. Robotic AGs were randomly initialized.

Experiment 2: Competition experiment. The simulation set up is identical as the one mentioned above except that robots with the two different controllers (ANN and GRN based) were competing in the same simulation and could mutually influence each other.

Experiment 3: Test to assess memory behaviour: The simulation set up is identical as the one mentioned for experiment 1, except that the simulation was run longer (and shown for 7000 time steps only). The experimental set up was run for several consecutive cycles in a row allowing the robots to continuously adapt their GRN. All simulations were repeated 50 times. In the main text, only few representative results are shown, as due to stochasticity, the behavior is not always exactly the same. The 50 other simulations resulted in a similar behavior, which we assessed as follows:

1. the difference between the maximal energy level between the last and the first cycle as a measure of the global energy gain and averaging those figures over the 50 simulations.
2. by calculating for how many simulations the energy is monotonically increasing over the different cycles (no fall-backs)
3. by calculating the average energy increase between two consecutive cycles (also assessed at the point where in each cycle the maximal energy level is obtained)

Experiment 4: Set up was identical as the one in experiment 3 except that here we compared the performance of:

- the GRN-based controller with full condition feedback (that is feedback of the environment on the life time of the agents, the gene specific mutation rate, and the condition dependent activation of the AG in activated agents
- the GRN-based controller with reduced feedback i.e. the feedback from the environment on the life time of the agents and the gene specific mutation rate (through the AV values) is disabled, but these robots can still uncouple the core network encoded by the AG from the condition-dependent activated network (data not shown)
- All input from the environment was disabled

All simulations were repeated 50 times. In the main text, only few representative results are shown, because, due to stochasticity the behavior is not always exactly the same. The 50 other simulations resulted in a similar behavior, which we assessed as in experiment 3.

Related controller types

To assess the extent to which a GRN-based controller results in an improved adaptability in a dynamically changing environment, we have compared the performance of our evolutionary GRN-based controller with that of two other types of controllers. The first one is a simple controller, implemented as a static, non-evolvable ANN that transduces environmental signals over a randomly initialized network structure (referred to as a Random ANN). The second controller is an evolutionary ANN controller that uses similar genome and evolutionary operations as the one used in Bredeche et al.[201]. All control parameters, including the nodes and the weights of all edges of the ANN have been randomly initialized and the controller will respond to the environmental inputs based on these control parameters. To make the comparison between ANN and GRN controllers as fair as possible, we have limited the maximum number of agents of our GRN controller to 200, thereby reducing the inner complexity and the size of the dynamic network in our simulation. On the other hand, we also used similar feedback loop and local optimization methods for the ANN as the ones used for the GRN controller. More specifically, the ANN controller we implemented makes use of a distributed learning function that allows every edge between two nodes in the ANN to change its vector and weight value based on the feedback of the robot performance. The weights of all edges in the network structure will be optimized separately at each time step. So just as in the agent-based system, the connections and the weights of the connections between the nodes (taking the role of the agents in the GRN-based controller) in the ANN are changing dynamically in response to the environment. Changing the network structure thus corresponds to the genetic alteration in our bio-inspired artificial genome, whereas changing the weights of the edges corresponds to the changes we impose on the agents. This implementation therefore uses principles that are similar to the ones used by Subagdja et al.[202] and Yu et al.[203]. In the ANN controller, each edge corresponds to an agent that responds to the global fitness of the robot's Fi in the following way (as determined by its adaptability value)

The adaptability value of the agent present in the ANN at time step i is:

$$\mathrm{AV}_i = \frac{F_{i-1} + F_i}{2}$$

Two parameters (AV_{max} and AV_{change}) will be assigned to each agent. These two parameters can differ amongst different agents. If AV_i is smaller than AV_{max}, the agent will calculate the value $AV_{distance}$ at time step i:

$AV_{distance(i)}= (AV_{max}-AVi)+ AV_{distance(i-1)}$

If $AV_{distance(i)}$ is greater than AV_{change}, the agent will change the weight parameter at that time step otherwise the agent will keep the same weight parameter. If AV is greater than AV_{max} , $AV_{distance}$ will become 0.

If the agent decides to change the weight parameter (W), it will add or subtract a certain value, based on $AV_{distance}$. The value increase C at time step i is determined as:

$C = AV_{distance} *R_{change}$

Where R_{change} is randomly assigned in the range of 0% to 50%, decided by the gene.

The mutation rate of the genome is based on the energy level of the robot. The lower the energy level, the higher the mutation rate. The mutation rates ranges from $2*10^{-4}$ to 0 (for the 350 genes in genome).

Assessment of the adaptability of robot controllers in simulation experiments

The average energy level of the robot population, the average energy gain of the robot population between subsequent time steps, the number of 'untouched' food sources, the number of robots that survive, and the population size are all parameters used to assess the general adaptability of the robot population. The energy level reflects, for each robot, its energy at a certain time point. The average energy level then corresponds to the average of the energy levels of all robots present in the population at a certain time step (i.e. total energy of all robots divided by the population size). The energy gain between consecutive time steps reflects, for each robot separately, the net increase in energy level between the considered time points, irrespective of the historical context of the robot. The average energy gain is defined as the average of the energy increase of all robots in the population between consecutive time points (i.e. total energy gain of all robots divided by the population size).

In our set up, robots with increased adaptability will have higher energy levels, which will lead to fewer deaths and more offspring, both of which result in larger population sizes. Based on the indicators mentioned above, the average adaptability of the robots is assessed. Besides measuring the overall energy level of the robots as a measure of their adaptability, we also traced their overall phenotypic behavior. More in particular, we assessed the evolution of the population size, and occurrences of attacks and aggregations (docking) during every time step over the whole population.

4.3.4 Results

Performance of GRN-based versus ANN-based controllers

Our GRN-based controller is different from previous controllers in several aspects. One of the most prominent features of our GRN-based controller is the uncoupling between the core and activated genome, which is achieved through the interaction between the 'agent based layer' and the 'bio-inspired AG' that defines the rules according to which the core network is translated into an activated network. To test the specific contribution of this combination to the performance of the controller, we compared with an ANN that is very similar in set up to our GRN-based controller, except for the design of its artificial genome, which does not allow for such uncoupling. As such, we hypothesize that most of the observed differences in adaptability of robots controlled by either controller can be attributed to the differences in the design of their respective artificial genomes. We compared the performances of both controllers under a dynamically changing environment (simulation parameters are described in the chapter 3). As a baseline we also assessed the performance of a simple non-evolutionary ANN based controller (referred to as a random controller). As expected, under all simulations, robots with an evolutionary controller greatly outperformed those with a random controller (not shown). The differences in adaptability, using average energy levels as indicators, between robots with evolutionary-based ANN and GRN controllers are shown in Fig 4.16. From these plots it is clear that, despite their similar performances at the beginning of the simulations, after a certain time, robots with a GRN based controller are more efficient in finding food sources (not shown) and therefore reach higher average energy levels than the ANN based robots. For all types of controllers, the energy levels drop after having reached an optimum for some time, which is due to food exhaustion (not shown). Interestingly, robots driven by a GRN-based controller show more variation in obtained energy levels between individuals than the ANN controller-based robots, reflecting the difference between robots with ANN and GRN-based controllers in exploring the search space and dealing with constraints imposed by the changing environment.

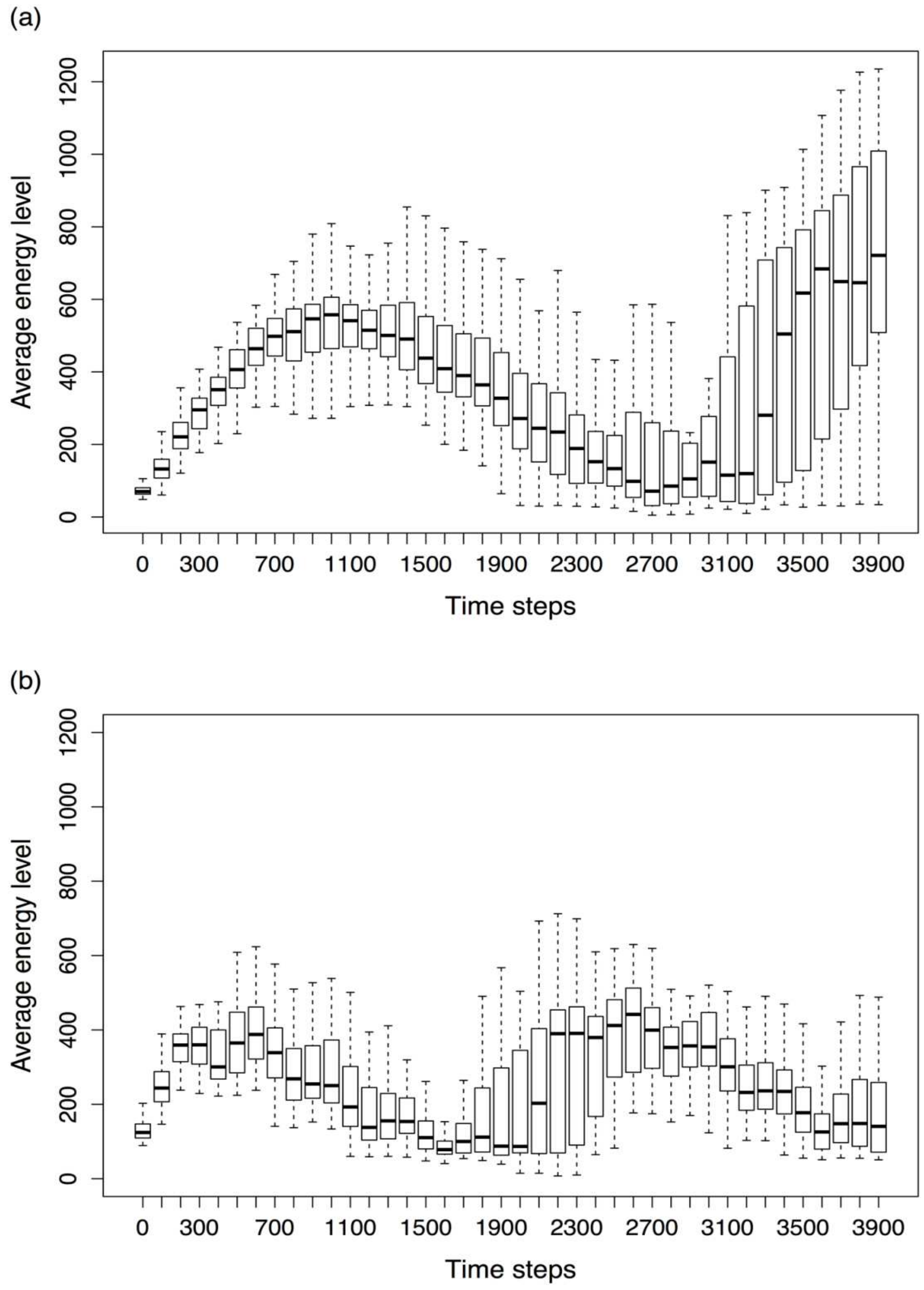
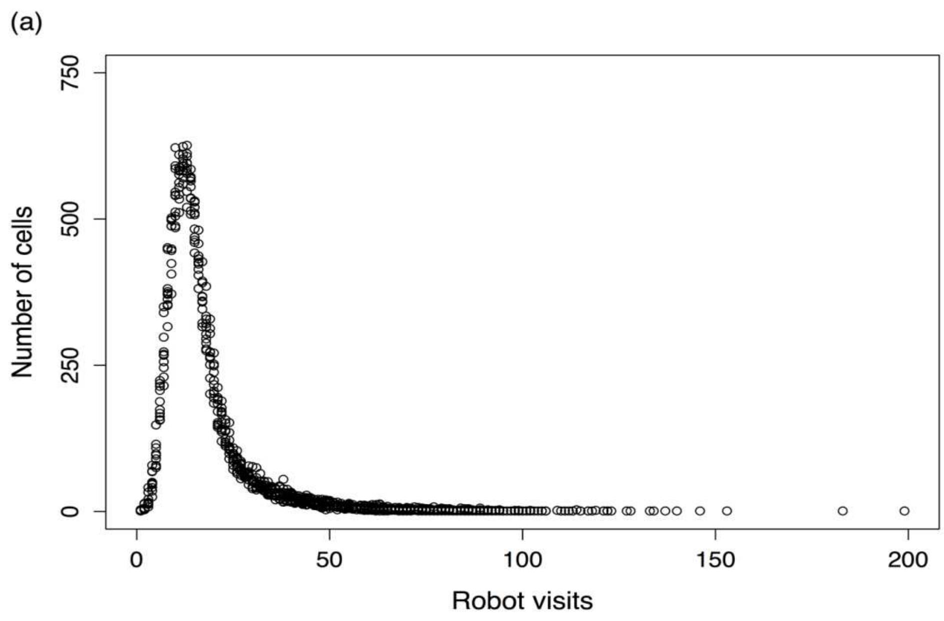


Figure 4.16 Comparison of the dynamics of the average energy level between robots with GRN (a) and ANN-based controllers (b). The x-axis represents running time measured in time steps, while the y-axis represents the populations' average energy level.

The populations' average energy levels are summarized for 50 independent simulations by means of box blots in which the solid line in the box represents the median value of the average energy of all simulations, the box borders correspond to respectively the first and third quartile and the extreme values correspond to respectively the lowest and highest average energy values observed in any of the 50 simulation experiments. When the number of robots in the population drops below 100, food resources are initialized again.

Besides measuring the overall energy level of the robots as a measure of their adaptability, we also traced their general phenotypic behavior. For instance, Fig 4.17 shows the area explored by ANN and GRN robots, respectively. As can be observed, GRN robots explore the environment more evenly than ANN robots. The difference in area exploration between the two different types of robot controllers is a reflection of their more variable movement behavior. The fact that, for ANN robots, a considerable number of cells are 'visited' many times (Fig 4.17a), implies either that, during the simulation, some robots wander around the same place for a long time or, alternatively, that more robots gather together at the same place. Considering the search for food sources and resource limitation in the environment, both situations are not ideal for the performance (adaptability) of the robots. GRN-based robots on the contrary tend to less frequently get 'trapped' in a certain situation (Fig 4.17b). They show generally more variation in the areas that get explored and therefore are less repetitive in their behavior. This implies that GRN robots more easily change movement strategies depending on the environmental situation in which they reside.



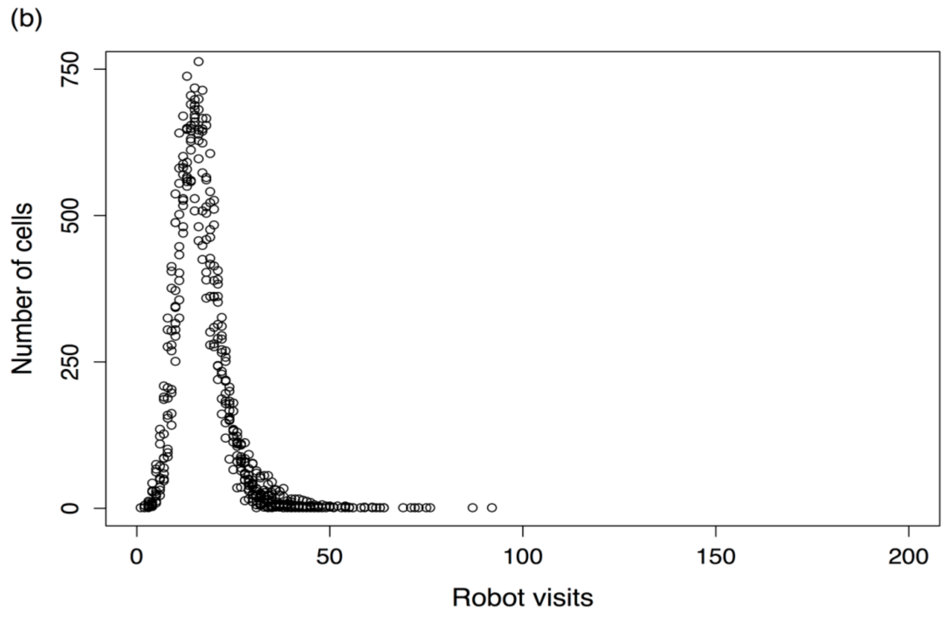


Figure 4.17 Movement behavior for (a) ANN and (b) GRN-based robots. The X-axis represents the number of robot visits, over 50 simulations, while the Y-axis represents the number of cells that have experienced that specific number of visits (non-cumulative).

Cells that have seen many visits (which is mainly true for the ANN robots) represent robots that spend much time visiting the same cell (i.e. robots have been trapped in these cells for a comparatively longer time), which implies that they do not explore the area as efficiently.

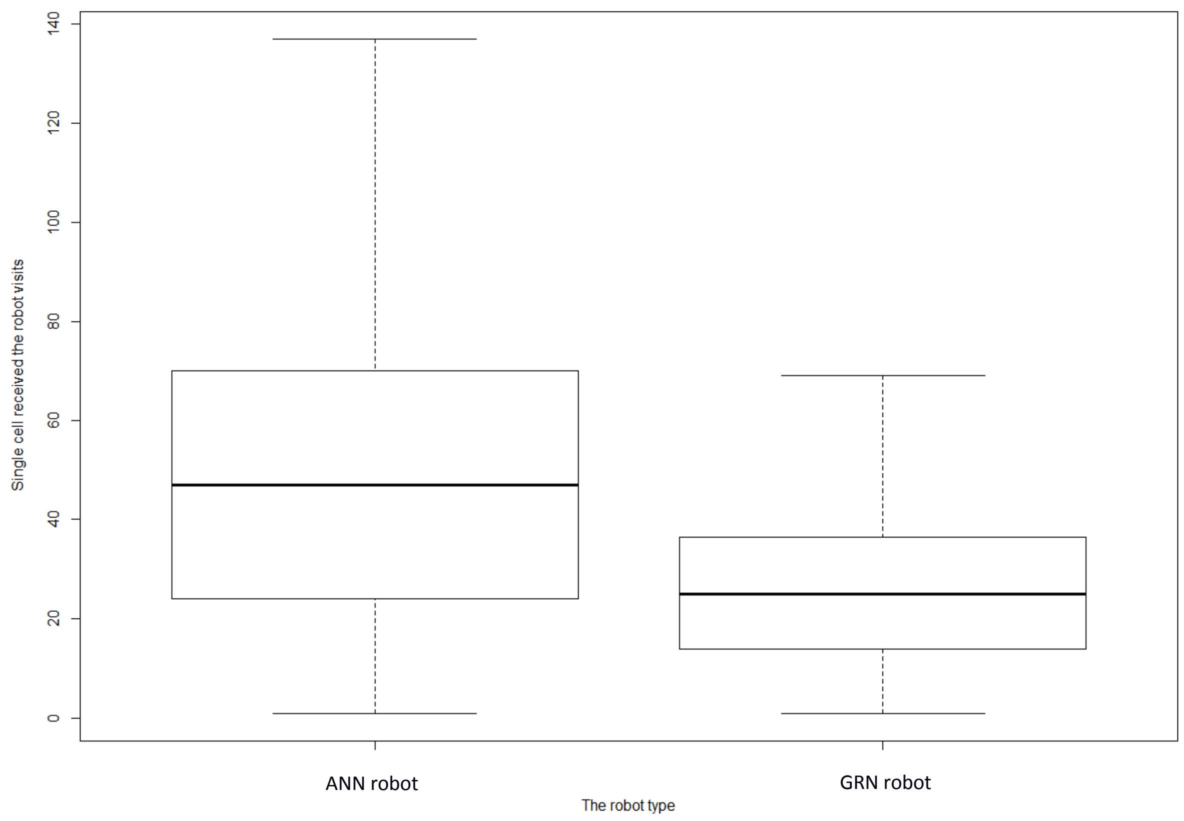


Figure 4.18 The statistic analyze on the robot visits record.

The figure above directly shows the range of the robot visits in the ANN and GRN robot simulations. The result clearly indicates that the robot visits in GRN robot simulation show a more even distribution (we only retrieve the data when the cells number and robot number in both kind simulations become constant and same, so more even robot visits distribution means better exploration under this situation).

T test report:

p-value < 2.2e-16

alternative hypothesis: true difference in means is greater than 0

sample estimates:

mean of ANN = 49.01080; mean of GRN= 26.47368

To directly compare the adaptability of our GRN controller with that of an ANN-based controller, we also performed competition experiments in which both controller types were run together in the same simulation environment (see details on section 'Simulation framework and scenarios'). In this experiment, the size of the initial swarm robot population was the similar for both controller types. As can be seen in Fig 4.18, the population of ANN-controlled robots in general adapts faster to the initial environment than the GRN-based robot population, as is shown by the more rapid initial increase of its population size, assessed as a higher value of the first derivative of the population increase over the first 1000 time steps, a behavior that was observed in 80% of the simulations. However, after this initial fast increase in robot population size, when food sources become more limiting and finding food more challenging, GRN-based robots tend to outcompete ANN-based robots, indicating that they can better cope with the changes in environmental conditions. Disappearance of the competitors decreases the competition imposed on the GRN-based robots, leading to a faster increase of the GRN-based population, a behavior that was observed for all (100%) simulations, for an average running time of 4000 time steps. At the end, the rapidly increasing population causes the food resources to become exhausted, resulting again in a decrease of the GRN population.

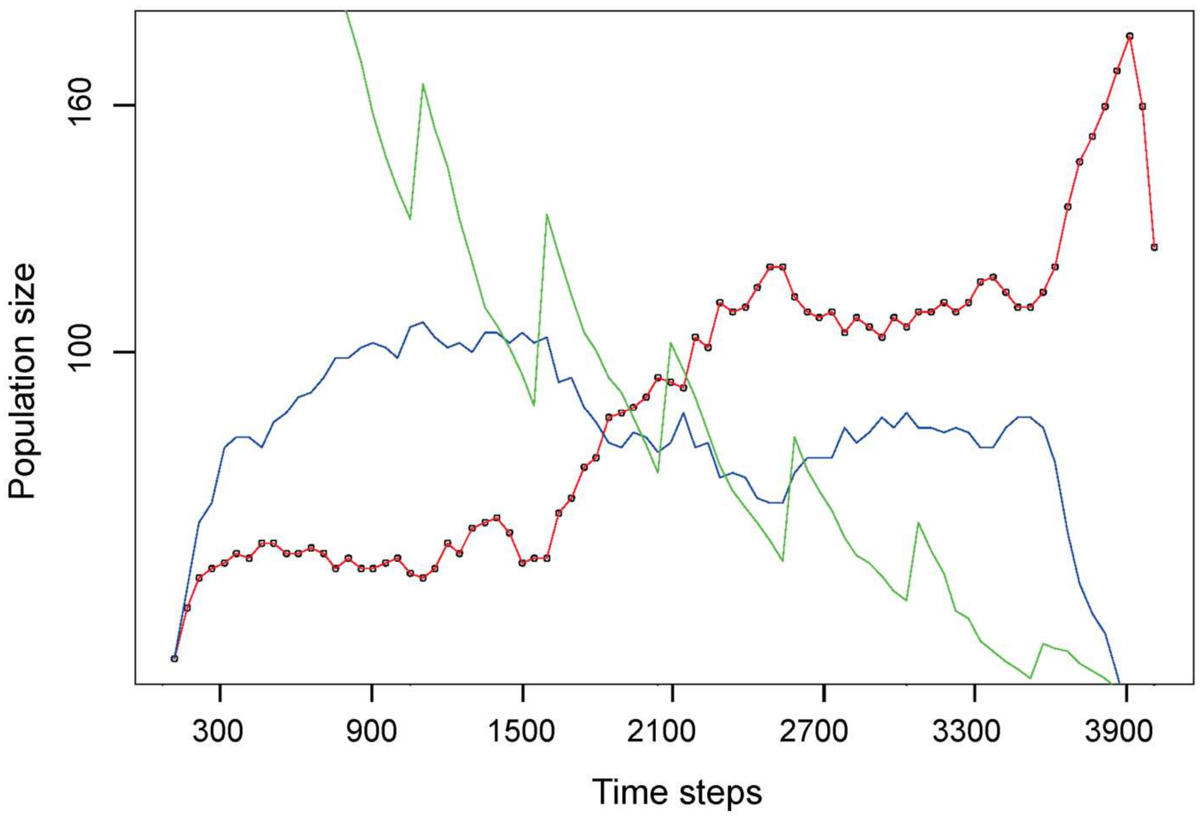


Figure 4.19 Evolution in population size of ANN and GRN-based robots in a (representative) competition experiment.

The X-axis represents the different time steps during the simulation. The red curve shows the population size (Y-axis) of GRN-based robots while the blue curve shows the population size (Y-axis) of the ANN-based robots. The green curve shows the number of available food sources. Increases in the number of food sources are due to the fact that the system will add new food sources with a certain rate after a pre-set number of time steps.

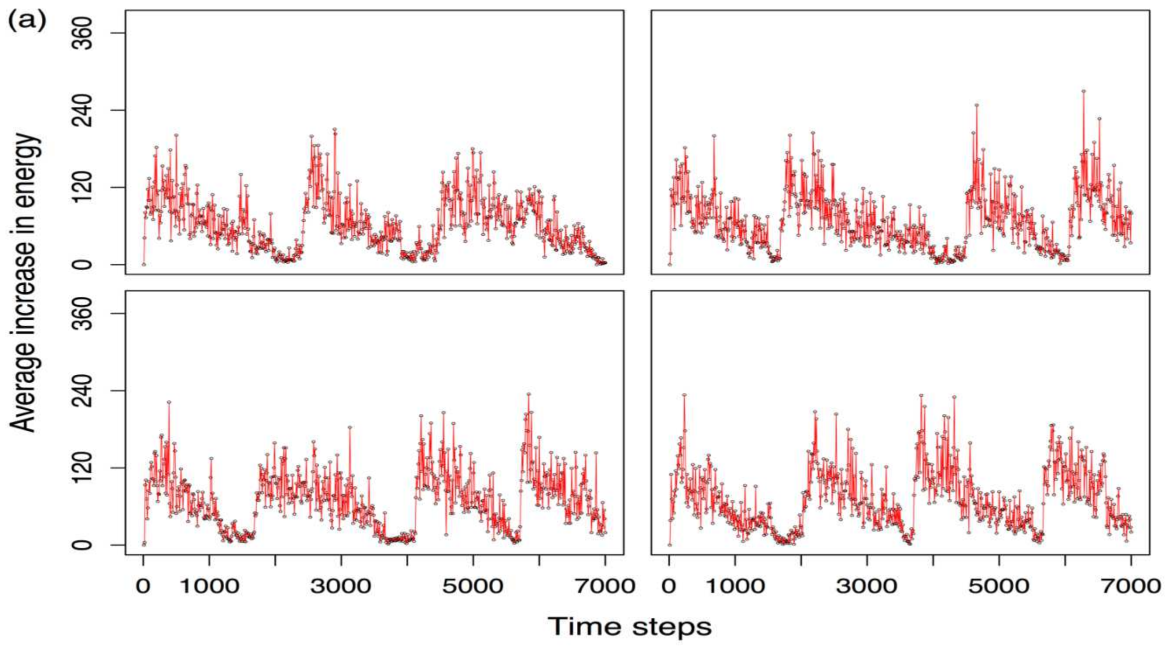
The results of these (and other, data not shown) simulations suggest that in general the GRN-based robots gain a higher fitness and show richer phenotypic behavior (better explore the search space, show more variable phenotypes, and are more resistant to limitations in the food resource) than ANN based robots. We hypothesize that this difference in behavior can be mainly attributed to the uncoupling between the core and activated network which is a main feature of our GRN based controller: by mimicking the presence of condition-dependent transcriptional activation through the encoding of ‘transcriptional interactions’, an environmental condition activates only part of the ‘bio-inspired’ genome. Only this activated part of the genome will contribute to or adversely affect the robots fitness, whereas its ‘non-active’ part will randomly change (due to evolutionary operators) without directly interfering with the fitness, allowing the system to more easily escape from local optima and to explore the search space more efficiently. For the ANN-based controller on the other hand, any alteration in the network structure will cause a global influence. So once the system has reached some optimum, a small change will often have a deleterious effect, making it hard to escape from the local optimum[204].

The ‘bio-inspired genome structure’ contributes to improved memory behavior

The specific way in which the GRN-based controller reaches its optimal energy levels reflects another important characteristic of GRN-based robots. In contrast to an ANN-based robot that re-optimizes its network each time it is subjected to a novel condition, our GRN-based system uses its bio-inspired AG to ‘store’ behavior that was optimal under a particular environment for a sufficiently long time. When subjected to a novel environmental condition, the previous condition-specific structure might become inactivated, but remains present. This ability to store ‘good behavior’ and to potentially disconnect it from the novel rewiring that is essential in a novel condition, allows fast re-adaptation if any of the previously observed environments is reencountered. In other words, GRN-based robots, as implemented in this study, theoretically leave a historical imprint in the system, here referred to as memory behavior.

To further demonstrate this behavior, we devised the following experiment in which we repeatedly imposed the same initial environmental condition and tested to what extent the GRN-based robots tend to rely or fall back on a previously evolved network to more efficiently adapt to a major switch in the environment (Experiment 3). As with all simulations, food sources were restored to their initial levels as soon as the robot population drops below 100 individuals. Also here, we compared the results to those obtained with an ANN-based controller that does not make use of the ‘bio-inspired genome’ and thus should lack the memory behavior.

Results are presented in Fig 4.19 and clearly show that the GRN-based controllers are more efficient than ANN-based controllers in finding food (or alternatively prey other robots), while they also survive longer, when an initial condition re-occurs, which can be inferred from the fact that the average fitness of the population (here assessed by the average increase of energy over ten time steps) is increasing despite the condition-resets. For the ANN-based controllers this behavior is less pronounced, and sometimes even reversed. For instance, we have calculated the rate of the average energy increase from the start of the environment reset to the next environment reset. For ANN robots, the rate of the average energy increase is 12.9 energy units/10 time steps and 14,85 energy units/10 time steps for the first and second condition reset, respectively. For GRN robots, these values are 15.91 and 21.74, respectively (computed and averaged over 10 different simulations). The fact that the GRN-based robots adapt faster suggests their controller can, upon a condition reset, invoke a stored part of the GRN (or the set of agents representing the GRN) that was already previously ‘optimized’ for survival on the encountered conditions. The fact that fitness increases, suggests that the robots continue to improve an already partially optimized network structure and do not have to start evolving the network from scratch again after each condition reset.



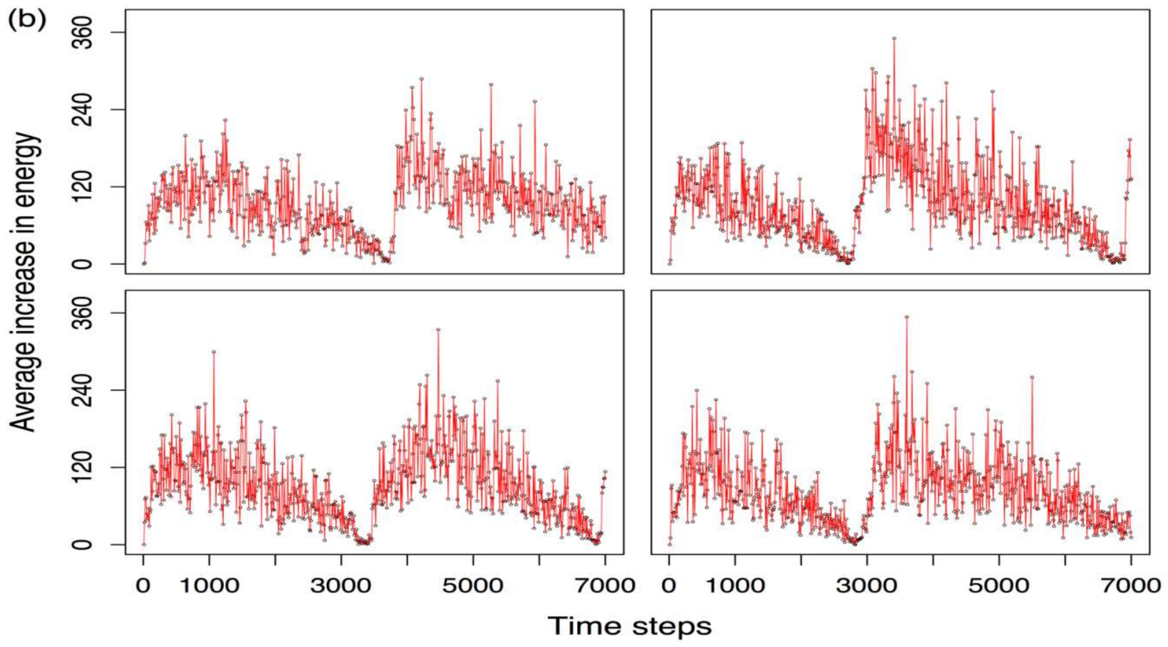


Figure 4.20 Average increase in energy for robots with ANN versus GRN controllers.

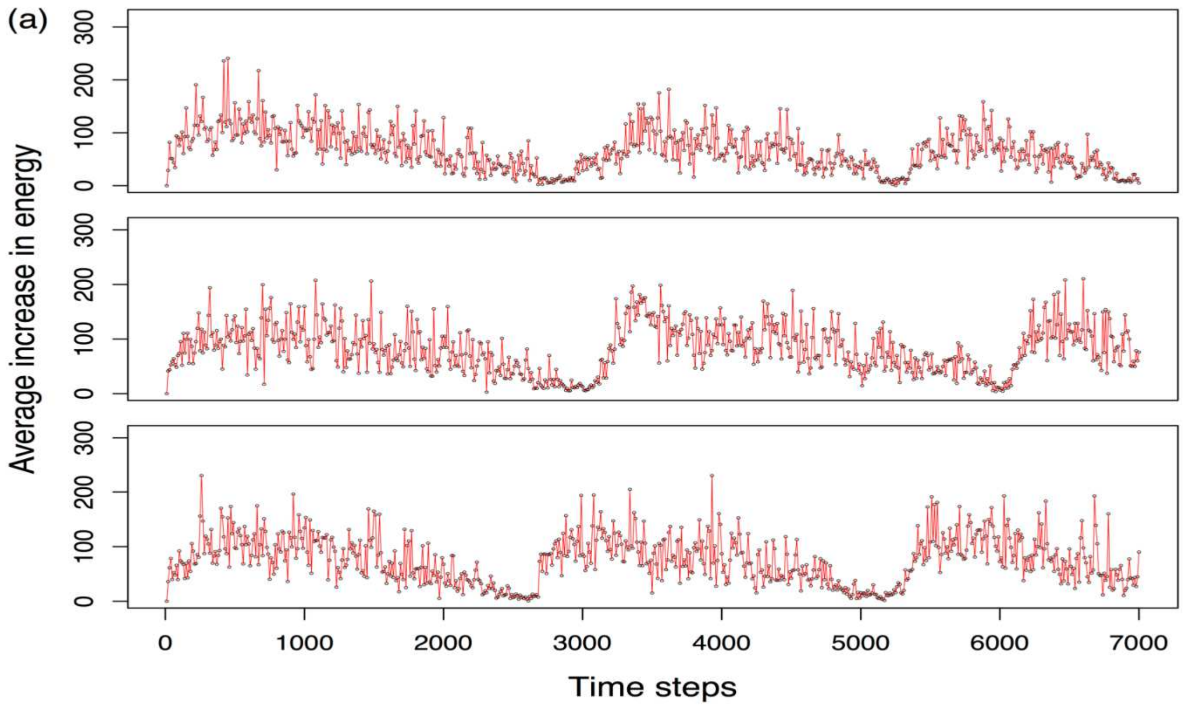
The Y-axis represents the average (of the entire robot population) increase in energy measured over ten time steps, while the X-axis represent running time measured in time steps. a) Four consecutive simulations are shown for robots with ANN controllers. b) Four consecutive simulations are shown for robots with GRN controllers. Drops are caused by food resource exhaustion. When the number of robots in the population drops below 100, food resources are initialized again, causing the population to recover.

Disentangle the effect of the environmental feedback from the bio-inspired genome

Besides the condition dependent activation of the agent-driven activated network (encoded by the AG core network), feedback from the environment is also used locally and can affect individual network components (more in particular the agents' life time and the gene specific mutation rates). Although we implemented an ANN-based system that can also cope with feedback acting locally on single genes and edges and that only differs from our GRN based system in not having the condition dependent activation of the AG, we cannot completely rule out that the improved performance of our GRN-based robots over the ANN-based robots can also be attributed to the differences in the way this local feedback is implemented in both systems.

Therefore, to unequivocally assess the relative impact of the way feedback is dealt with versus the conditional uncoupling of the core from the activated network, we disentangled the impact of both factors in the GRN-based system: we compared the fully functional GRN-based controller with, respectively, a GRN-based controller in which all feedback has been disabled (i.e. the feedback from the environment on the mutation rate and the agents' life time as well as the feedback responsible for the condition-dependent activation of the core genome into an agent driven activated GRN) and a GRN controller in which only the feedback from the environment on the mutation rate and the agents' life time was disabled.

Fig 4.20 shows the overall differences in adaptability of a controller where all feedback has been disabled and a controller in which all feedback has been enabled. As expected, in general, fully functional GRN-based controllers reach higher fitness, again measured as the average increase of energy over time. Although the initial performance of the robots without feedback is similar to the ones where feedback has not been disabled, the fully functional GRN robots show much better performance, particularly after the environment has been ‘reset’, suggesting that the feedback mechanisms are instrumental for the improved performance, hence adaptability, of the robots. Importantly, simulations where only the feedback from the environment on the mutation rate and the agents' life time was disabled, show a performance that is quite similar (only slightly improved) to that of fully enabled systems (data not shown), suggesting that it is indeed mainly the feedback responsible for the condition-dependent activation of the GRN that is crucial for improved adaptation.



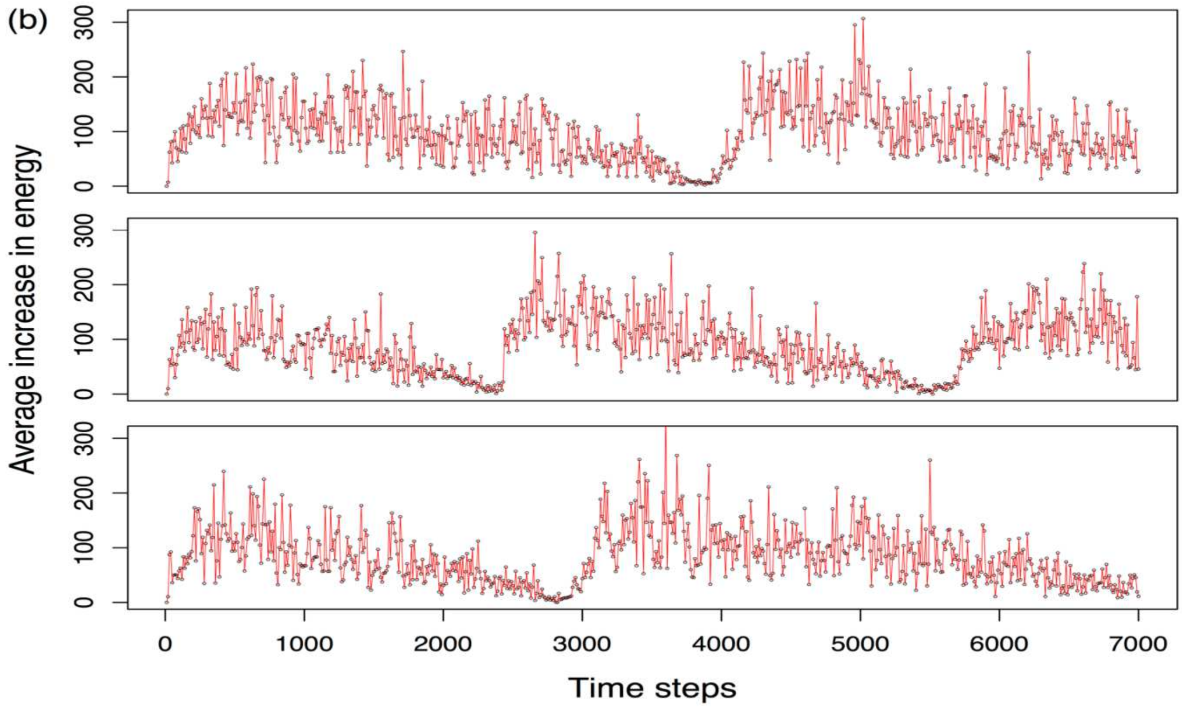


Figure 4.21 Comparison of the GRN controller robots with and without feedback.

The Y-axis represents the average (of the entire robot population) increase in energy measured over ten time steps, while the X-axis represent running time measured in time steps. a) Three consecutive simulations are shown for robots with GRN controllers with all feedback disabled. b) Three consecutive simulations are shown for robots with GRN controllers with feedback enabled. Drops in average energy increase are caused by food resource exhaustion. When the number of robots in the population drops below 100, food resources are initialized again, causing the population to recover. The difference of performance is mainly represented by two values. The first value is the maximum energy level that robots have reached in the simulation. For the feedback enabled group, before resetting the environment, the maximum energy level is in the range of 200-250 but the value has increased after the environment has been reset (the new range is about 290-320). The second value is the surviving time for the whole population. When the population size is above 100, we regard the population as surviving. How long the population can survive during the simulation represents the adaptability of the population. This value (indicated by the interval time between environment resets) also has apparently increased after the environmental reset in the feedback enable group (the actual extension of surviving time is variable based on each experiment but it always have a clear extension about 15%-30% compared with the one before environment reset). For the feedback disable group, we did not observe any apparent increase on both of these values after the reset.

4.3.5 Conclusions

The self-innovating nature or evolvability of biological systems depends on their ability to store information acquired during the past that can be reused on later occasions. For instance, bacterial systems that have been subjected to reoccurring conditions have been shown to develop memory behavior after several rounds of training[205]. Another key factor contributing to the evolvability of biological systems is the presence of epistasis or the ability to explore a vast combination of mutations, some of which can be neutral or even deleterious to the fitness but of which the combination can largely enhance fitness values[206,207]. Being able to explore the search space trough fitness valleys therefore is a key factor of evolving novel emergent behavior[208]. In this work, we hypothesize that key to this memory behavior and ability to release epistatic interactions is the decoupling of the genomic information encoding the full regulatory network (here referred to as the core GRN) from the activated part of the network. This is, amongst others, proven by the fact that cryptic variation in genomes, i.e. variations that can occur without directly interfering with the fitness, have been shown to contribute largely to the evolvability of natural systems[184,209]. In addition, billions of years of evolution have shaped the genetic contingency of natural systems to be highly modular and degenerate. This modularity (e.g. presence of well-defined pathways) and degeneracy is the result of selecting systems that can efficiently anticipate on novel conditions without the requirement of a network rewiring that would prove detrimental in other conditions[210,211].

Here, we tested whether imposing such bio-inspired design in which the genome and the activated part of the network are uncoupled could also improve the evolvability of an artificial self-adaptive system. To this end, we developed a robot controller that combines an artificial genome with an agent-based system that represents the activated part of the regulatory network. As in biological cells, the full regulatory network is encoded in the genome, here represented by an artificial genome consisting of both regulatory and structural genes. Depending on the environmental signals or cues, part of the encoded network is activated following the rules of transcriptional regulation. The activated part, modeled by an agent-based system, is responsible for sensing the environmental signals (signaling agents), transducing these signals through the network (regulatory agent layer, reflecting the gene products of the corresponding genes) and translating them into the proper behavior (mediated through the structural agents). Whereas the artificial genome represents the encoding of the transcriptional network, the agents can be seen as the functional gene products (i.e.

proteins) of the encoded genes. This way, the agent-based system mimics the active regulatory network and signal transduction system that is also present in naturally occurring biological systems.

Our simulations indeed show that separating the static from the conditionally active part of the network by using a bio-inspired design contributes to a better adaptive behavior. We believe that the specific ‘memory’ behavior and improved ability to deal with changing conditions can be mainly attributed to the ‘bio-inspired genome’ that allows uncoupling between the static and the condition-dependent part of the network. It should be noted that this work represents only a first implementation of our approach and more work is necessary to see how we can further improve on the realistic mimicking of gene regulation in artificial life forms.

Chapter 5

A novel simulation framework for simulating multiple level interaction-based evolution with a nested system architecture

Author contribution

I performed all the development, experiment implementation, result collection and computational analysis described in this chapter. All these works were done under supervision and significant contributions of Yves Van de Peer and Kathleen Marchal.

This chapter is partly reproduced from a paper in preparation:

Yao, Y., Marchal, K., Van de Peer, Y. A novel simulation framework for simulating multiple level interaction-based evolution with a nested system architecture

5.1 Introduction

To achieve our aims introduced earlier, namely to set up a computational framework to study adaptation in complex systems as a response to a changing environment, we have implemented multiple level interactions in our simulation framework, which thus includes the robot level, the GRN level and the gene level. In chapters 2 and 3, we have separately discussed artificial gene regulation and artificial life simulation. Artificial gene regulation focuses on processes that happen inside an organism (GRN level and gene level), while artificial life simulation emphasizes the interaction between different organisms in the same population and simulates various adaptations of the different organisms dynamically (robot level). In our simulation experiments, we have integrated these two parts in one simulation providing a holistic view on evolution. We embedded artificial gene regulation (through GRNs) in every simulated robot. The output of the artificial gene regulatory process determines the phenotype of the organism, while the environmental cues such as the number of food sources or the number of surrounding organisms can also affect gene regulation through feedback mechanisms.

In chapter 4, we have demonstrated that our framework can improve the adaptability of the robots in a changing environment. In this chapter, we will further discuss and demonstrate the relation between complex adaptation and the multiple level interactions using the same artificial evolution simulation experiments as used in chapter 4.

5.1.1 Emergent complex adaptation

For a trait to be adaptive, this usually requires the cooperation of several genes while their contribution to fitness is influenced by many factors. For instance, even when the external environment is stable, the importance of a particular trait is still dynamically changing because of changing behavioral strategies, dynamic niches, aging processes and so on. In addition, adaptation of an organism is not based on any single trait but usually involves several traits. This makes adaption complex and therefore we talk about complex adaptation [206,212,213]. To understand how complex such integration between different traits can be, we will take the metaphor example of company mergers or recruitments. For a small company, integration may be not really being a problem because the cooperation is simple and every position has a clear function. However, for huge international organizations, making a successful merger or recruitment almost becomes an impossible mission even for experienced HR experts. The only way to integrate such complex organizations is allowing enough time for interactions between individuals to evolve gradually. Through interactions, integration will grow organically and will become self-organized in the system. Suddenly inserting a novel individual or entity in a complex system can lead to an unpredictable result even with careful design and preparation. The more complex the system is, the more risk it will take for introducing changes. Therefore, the complexity of adaptation is actually often difficult to understand. One consequence of complex adaptation, as discussed by Michael Lynch [207], is that "character alterations require more than one novel mutation to yield a functional advantage". Even if one single 'mutation' could lead to a novel trait (for instance in the case of genetically modified organisms), the novel trait still needs to 'exist' and 'persist' in a biological context (non-genetic factors such as niche, culture and so on). It is clear that any novel adaptive trait requires interaction with other traits and any one of these seems not likely to adapt with the novel trait without a clear biological context[214-216]. Therefore, it is hard to see how complex adaptation of organisms can emerge from random and discrete mutations. If a single mutation does not solely lead to a certain adaptive trait, it will tend to be removed from the genome under selection pressure. As a result, to explain why novel adaptations continuously occur during evolution, some argue that the pleiotropic effects of genes and phenomena such as gene duplication can counteract the effect of individual mutations (by for instance increasing the redundancy)[217]. However, Fisher, Wagner, and Orr also brought up another problem about pleiotropy and gene duplication in their evolutionary studies [218-220], which is referred to as the "cost of complexity". Pleiotropy and gene duplication will inevitably increase the complexity of gene regulation[220]. Based on Fisher's geometric model, the rate of adaptation decreases quickly with the rise in organismal complexity (also see again the example of company mergers), because complexity makes mutations to have more pleiotropic effects on the phenotype. To balance all these effects and then reach the subtle equilibrium in adaptation eventually will require more precise changes. In addition, it is obvious that the adaptive status with higher complexity is easier to be destroyed by random changes. For complex systems, the more random changes that occur, the more damage they can cause. All in all, higher complexity tends to require more precise mutations to form an adaptive alteration while it decreases the tolerance of the system to random trial and error. Based on the traditional systems' framework (without considering interaction at multiple levels), the contradiction discussed above makes it difficult for the system to overcome the "cost of complexity" (see more details in [221]) and to achieve novel complex adaptation in evolution. However, we assume that considering interaction at multiple levels of evolution can help the system to avoid this problem. Interaction at multiple levels can separate the whole complexity into individual agents. Through selectively picking the particular agents at different time, the system can always rearrange the evolutionary modules in a dynamic way, which increases the robustness of the whole system during evolution and the possibility of evolving complex adaptation. Ulieru and Doursat [222] proposed a similar idea of using a 'bottom–up' design to improve the efficiency of the system under a complex context and they referred to the way of such system design as "emergent engineering". In our research, we implemented 'emergent engineering' by an agent-based system and simulated complex adaptation in our artificial evolution. The results (see also chapter 4) suggests that our new model can accelerate the adaptive emergence of complex behavior under a complex scenario.

5.2 Methodology

5.2.1 General description of the simulations

In this particular experiment, we have run 100 different simulations (based on the same simulations discussed in section 4.3 but we use the different part of the result). The general environmental setting was the same for all simulations (see also chapter 4). Half of the simulations used GRN robots and half used ANN robots. Detailed information on the simulation environment can be found in chapter 3. What is different in the result part discussed in this chapter is that we focus on analyzing the collective behaviors of robots here and the result does not include the part after food sources have been restored to the initial situation. The population will be regarded as experienced an extinction when the population size is smaller than 100 (robots). After the environment restored, the result will be influenced by the previous evolutionary context (variable replication rates and the population partition on each simulation make the result between simulations are less comparable) and this makes the demonstration too complicate so we only analyze the result before the environment restore in this chapter. The more complicated scenario (i.e adding the extra external changes like the environment restore) will be investigated in future work. There is no explicit fitness function. The energy level of each individual robot is the only critical factor for its survival or reproduction but its energy level is not directly responsible for any particular trait or behavioral pattern (robots can have various strategies to obtain or save energy during the simulation, so the energy level of robots is always the combination of multiple strategies and various environmental conditions). The energy level of the robot is completely dependent on the interaction of the robot and other robots.

Through the simulations, we compare the adaptations of the two kinds of evolutionary robotic models in a dynamically changing environment. This simulated changing environment mimics the features of a natural environment (see the details in the chapter 3). The environment dynamically 'interacts' with the robot's behavior during the simulation and the corresponding adaptation of every individual robot also can be different based on the specific context.

To compare the result of our novel agent-based framework (discussed in detail in Chapter 2) with other network-based frameworks, we also implemented an artificial neural network (ANN) into the robotic organisms. For the ANN robots, we use a direct encoding method to produce an ANN to represent regulation. We also set reinforcement learning [223,224] on each edge of the ANN to optimize the structure of the ANN. The details of the ANN robot can be found in Chapter 2 and the appendix.

5.2.2 Comparing GRN and ANN-based robots

The experiment that will be discussed in this chapter is a same implementation as the one discussed in section 4.3.

As stated before, in the GRN robots, we use a bio-inspired genome (while the ANN genome is a randomly generated string of numeric values) that is based on a 4-digit sequence while the agents are corresponding to a particular gene product (see chapter 2). The function of the agents is not the tuning of the parameters between one gene and one particular function but rather defining/deciding on the binding with a particular genetic region (i.e. binding region, different genes can have the same binding region) on the genome or on the interaction with other agents. Gene regulation in the GRN robots is actually defined by the interaction between all agents (which forms the translated GRN). Compared to the ANN robots, there are three main differences:

The first important difference is that the pattern of selection in the ANN is unique and only based on the performance of the robot. In other words, there is no multiple level selection in the ANN robot.

For the GRN robots, the pattern of selection has a multiple level architecture, as described before. At the level of the single organism, selection is based on the behavior and the environmental context of the organism, however at the agent level, each agent in the robot can have its own particular pattern of selection based on the 'inner environment' (the local situation in the same robot: i.e the concentration level of other agents in the same robot) and the 'functional features' of the agent (i.e the similarity of the binding motifs between agents). At the genetic level, the pattern of selection can vary from gene to gene based on the performance of the corresponding agent (see chapter 2 for details).

The second difference is that the behavior of the GRN robots emerges from the interaction between agents (i.e. gene products, robots) at multiple levels (e.g. the genes will be selected based on the long term context while the agents (gene products) will be selected based on the short term context). The behavior of the ANN robots is based on the explicit network structure and the current environmental inputs, so they assume less interaction (between nodes in the network). For instance, when one connection in the ANN has changed, the vector and weight parameters of other connections will not adapt to that change (output values will change but not the parameters). Therefore, the effect of one change in the ANN is certain. In the GRN on the contrary, changes in one gene may lead to the activation or repression of other genes and as such not only change the output but can also influence the selection pressures and local environment (context) of these affected genes or agents. So changes in the GRN may propagate into directions that cannot really be predicted or anticipated (emergence), while this is much less the case for the ANN robots.

The third difference concerns the nature of mutations. Our ANN robots use a genome that uses direct encoding, in other words, the genome directly encodes the initial parameter values of the ANN. In contrast to bio-inspired genomes, the effect of mutations on the direct encoded genome is limited by the predefined genome structure. In the bio-inspired genome, mutations at different positions will have different effects on gene regulation and gene expression while such diversity of mutational effects is greatly reduced in the direct encoding genomes of ANNs, because every mutation in the direct encoding genome will explicitly change only one corresponding parameter of the ANN network. In addition, the mutation rate of the ANN genome is fixed while the mutation rate can vary according to the 'adaptation' of each gene in the GRN robot (feedback mechanism, see Chapter 4).

The way of selection, the use of a simpler 'context' (fewer interactions) and the global mutation process simplifies the evolutionary process on the ANN genome and the role of genes as independent evolutionary objects. This is evident from our simulations comparing ANN and GRN based robot controllers (see further). However, it should be stressed again that the aim of study is not so much to show the superiority of the GRN model over the ANN model (because there are limitations in comparing both models), but rather to show and evaluate the potential of the GNR approach in adaptation of complex system behavior, which we do feel is more likely to occur in the GRN based approach(or the similar emergent engineering approaches).

5.3 Results

In this section, I will present the results of our most recent experiments.

The result includes the simulation running with GRN robots and ANN robots. At the population level of our simulation experiments, the collective behavioral pattern is always evolving during the artificial life simulation (same for GRN robots and ANN robots); at the genetic level, for GRN robots, both the genes (through mutations) and the GRN (through the interaction of the individual agents) are evolving in the robot. For the ANN robots, the whole artificial neural network directly interacts with the environment and the robot's behavior through learning programs too (see Appendix), but the components in the network (i.e. the weight parameters of the different edges and the number of nodes) are not dynamically influenced by the interaction between learning programs after the ANNs have been formed. Therefore, GRN robots have a more comprehensive system of interaction between genes, and expression and phenotypic features.

Such difference between GRN and ANN robots makes GRN controllers in the robots focus on reaching an equilibrium rather than on the certain system structure. In other words, the GRNs in the robots have a group of dynamic statuses and the transitions among these statuses are based on the interaction of agents, genes and environmental conditions. Contrary, in the ANN robots, there is no corresponding interaction to influence the individual components inside of ANN, so only the whole network interacts with the environment but not the particular sub-components.

By comparing the evolutionary trajectories in the GRN robot and ANN robot simulations separately, we hope to get more insight in the complex interactions that occur during the artificial evolutionary process and complex adaptation. In addition, the result also demonstrates that simulating the fine-grained interaction at multiple levels accelerates the emergence of novel adaptive patterns during the artificial evolution.

During the simulation, we track different features (i.e. energy level, the number of prey actions, the number of new agents and so on) of the robots every 10 time steps.

5.3.1 Evolution of collective behavior

The collective behavior of organisms reflects the interaction at the robot level, while the collective behavior of a population in turn determines the local environment of the individuals in the ecosystem. In other words, the collective behavioral pattern of the population can also influence the organisms' local niche and a changing niche or environment, in turn, can effect the further adaptation of the organisms. Here, during our simulations, we specifically analyzed the evolutionary process regarding the collective behavior of organisms. The collective behavioral pattern in a population is based on the interactive behavior of each individual organism while the behavior itself is dynamically evolving during the whole evolutionary process. In our scenario, here, we particularly focus on prey and aggregation behavior, since these two kinds of behavior represent some basic relationships of interaction during evolution[225,226]. The prey behavior represents the competitive relationship between robots, while the aggregation behavior represents the cooperative relationship between robots. It should be noted that before the predator actually preys, it needs to do an attack. An attack action does not always result in preying: if predators are comparatively weaker, an attack action will only cost more energy.

In the simulated environment, there is a basic equilibrium that robots need to reach for surviving. The equilibrium concerns the food resources and the food consumption (food consumption is related to food searching efficiency, robot population size and so on). As we introduced in chapter 3, new food replication (similar to plant growth for instance) is based on the number of the current food sources and is inversely proportional to food consumption and the population size. If robots consume the food too fast, food resources will get exhausted rapidly, which will lead to extinction of all robots. On the opposite, if the robot cannot find and consume the food efficiently, the particular robot will become weak or will even die of starvation. All robots that survive and the whole population need to adapt their behavior to reach a state of equilibrium during the simulations.

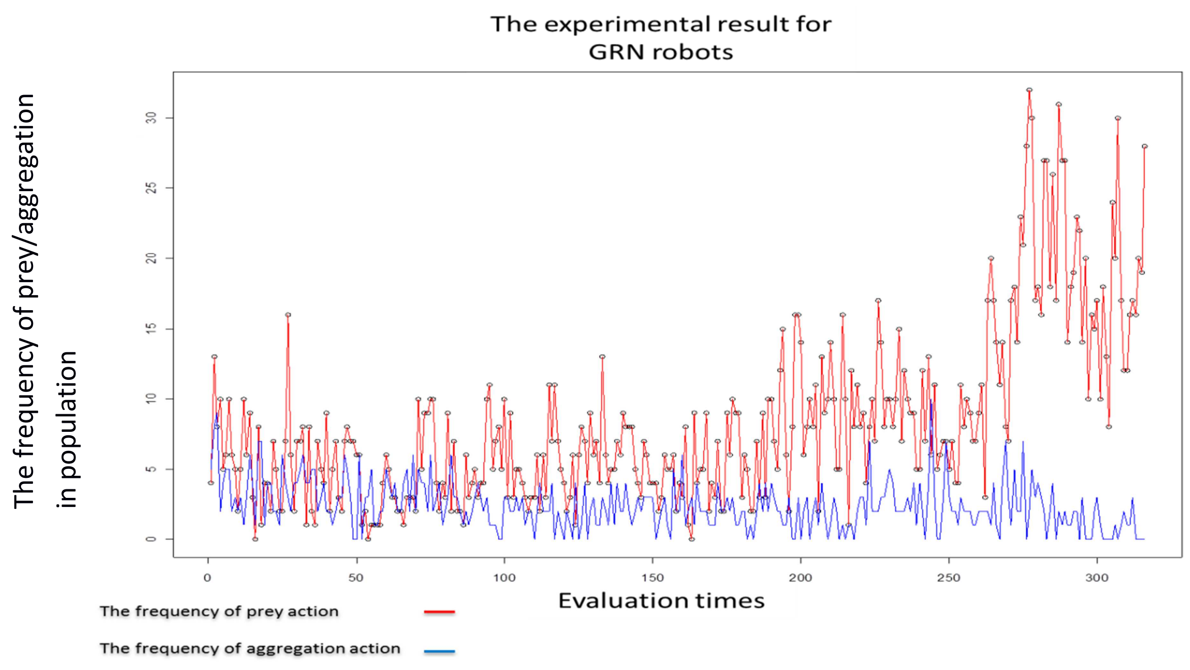
In our scenario, efficient searching behavior can help robots to find more food but it also may lead to overly-fast food consumption rates. Anyway, the searching behavior will be optimized during evolution since it could directly increase the adaptability of individual robots in a short time. To slow down the total food consumption of the whole population, some robots can prey others to gain energy instead of searching for food, if that would turn out to be the better strategy. However, prey behavior requires an extra energy cost and it needs the predator robot and target robot to be in the same cell space. Compared with searching for static food sources, finding moving robots and attacking them to gain additional energy will be more risky (i.e. pursuing and attacking other robots does cost extra energy) when food sources are abundant. On the other hand, robots could choose to aggregate to share the energy and be able to better defend against possible attacks. For joining the aggregated multiple robot organisms, every robot will also ‘pay’ an extra risk for the integration (the different GRN controllers on each aggregated robot may have disagreement on the common behavioral control of the robotic organisms and each aggregated robot has to share its energy with the others). At every time step, every robot in the simulation can decide its next behavior based on the inside GRN or ANN controller.

Comparing the GRN robots simulation with the ANN robots simulation, we found that both robots show different collective behaviors. At first, as can be seen in figure 5.1, both prey and aggregation behavior happens more frequently in the ANN robots. During the simulation, the GRN robots explore the grid more efficiently than the ANN robots (see further, and chapter 4) and this simply reduces the possibility of attacks or aggregation, simply because the robots ‘avoid’ each other. However, more or less prey and aggregation behavior does not necessarily indicate better adaptability of robots because adaptation needs cooperation between several different kinds of behavior and behavior can be completely different for GRN robots than for ANN robots. Therefore, in our experiments, more than comparing ANN and GRN robots (comparisons have their limitations, as discussed before), we are interested to investigate the change in behavior for the same sort of robots but at different times or situations (see how they evolve and adapt during the experiment/simulations).

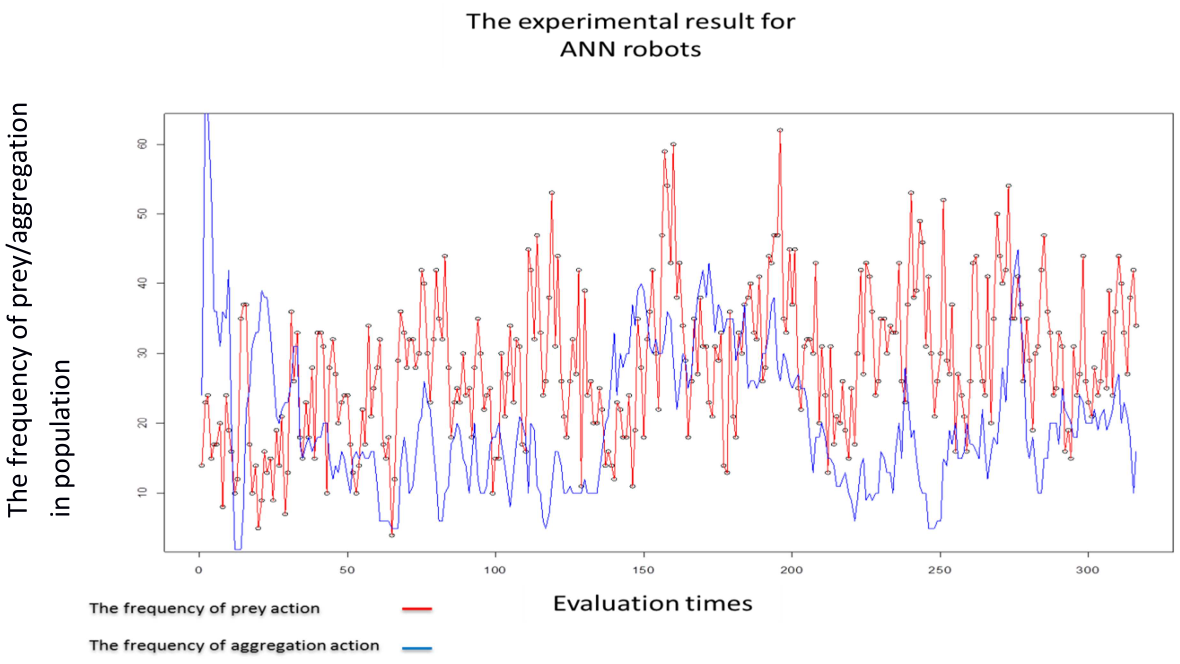
Maybe a bit unexpectedly, in both simulations (GRN and ANN), aggregation behavior does not look very pronounced. We assume that this can be explained by the fact that aggregation requires extra cooperation between multiple robots and evolving such cooperation needs a longer time than evolving other kinds of behaviors. This will be investigated in future simulations with longer run times.

Furthermore, in our current scenario, the aggregated robotic organism has all the single robots' controllers running and the organism is driven by the average of all controllers' output. Moreover, the evolving context based on the single robot's adaptation could be deleterious for the cooperation (i.e. different members have their previous strategies and such strategies have been evolved individually. After joining the group, all members have to use the same strategy and neutralizing their individual strategies could be deleterious to most members). To really to be able to evolve interesting aggregation patterns in our simulations, we probably need to simplify integration and cooperation in the multiple robotic organism (i.e. give a common controller instead of neutralizing all controller's outputs), otherwise the adaptation will repress aggregation as we think we observe now.

Regarding prey behavior, in the GRN robot simulations, unlike in the ANN robot simulations, we often observe that the occurrence of prey behavior greatly increases in the population during later stages of the simulation (see the example in Fig 5.1a). For the ANN robot simulations, (Fig 5.1b), high prey frequency distribute more evenly during the experiments and do not seem to be a specific adaptation when food becomes more scarce (more details will be discussed later).



(a)



(b)

Figure 5.1 Comparison of the frequency of successful prey and aggregation actions in ANN and GRN simulations. (a) shows the result of GRN robots in the simulation, as observed in several cases (about 10% of the simulations). (b) shows the typical result for ANN robots in the simulation (1 Evaluation time=10 time steps).

Figure 5.1 only shows one comparison between two simulations. In fact, many GRN robot simulations show very similar tendencies (more examples shown in the figure 5.2). However, based on different initial genomes and food distributions, increasing prey behavior in different GRN robot simulations usually emerge at slightly different times. As mentioned above, the aggregation patterns did not change much during the simulations due to the integration cost, and therefore we have here only focused on prey behavior.

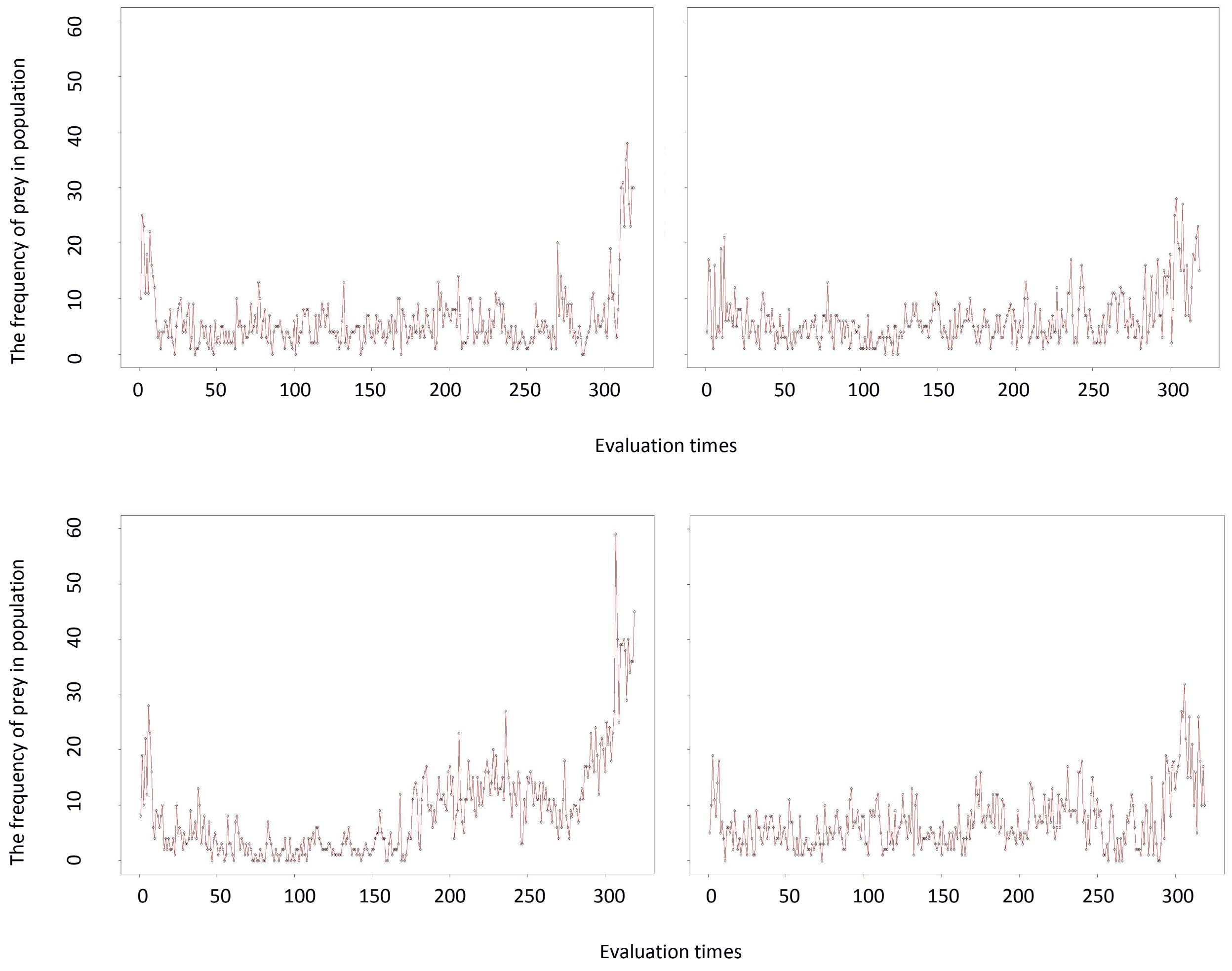


Figure 5.2 Additional examples of prey behavior adaptation in the GRN robot simulations.

When food is becoming scarcer, in some simulations, prey frequency often rapidly increases for GRN robots (see examples in figures 5.1a and 5.2). This likely represents an adaptation of the population because prey behavior can become more important when the food becomes scarce. Prey behavior reduces the population size but increases the energy of the surviving robots. On the other hand, prey behavior also comes with a risk and costs energy to every individual robot in the population (defence and attacks both cost extra energy). In the simulation, both kinds of robots will gradually optimize their food searching ability. When food becomes scarce, prey behavior cannot only be more feasible for the individual robot to gain energy but also to keep the balance between the population size and food availability. Therefore, increasing prey frequency is a form of adaptation when food becomes scarcer, while it also keeps the necessary balance between food availability and population size.

In the GRN robot simulations, increased prey frequencies seem to emerge only when the food sources become really scarce and reach the lowest levels. In these cases, evolving prey behavior could be seen as some sort of last resort in order for the population to survive (longer). In general, already without adaptation for prey behavior, GRN robots can survive much longer with fewer food sources than the ANN robots. While on average, the ANN robot population dies when no more than 120 food sources are available, the GRN robots survive until the number of food sources has been reduced to, on average, 80 or less (see Fig 5.3).

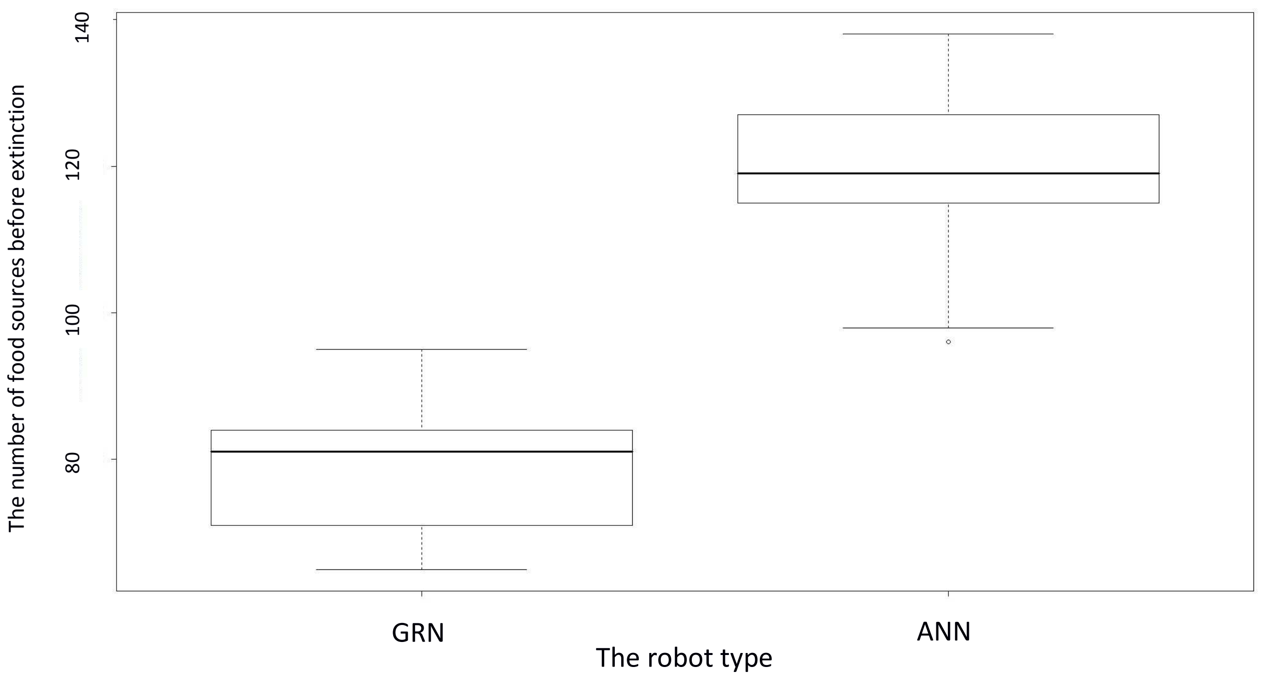


Figure 5.3 Comparison on the number of food sources left before extinction of the population (sample estimates: mean of GRN=78.38889, mean of ANN=119.52174: Significantly different based on two sample t-test: p-value < 2.2e-16).

The fact that the GRN robots can, on average, survive much longer, is probably due to the fact that they are better in finding the food sources (and in exploring the grid, see also further in section 4.3), also when these become scarcer. This has also been demonstrated previously (see Chapter 4, figure 4.17). Finally, as observed before, GRN robots have higher overall fitness (evaluated based on the energy level) than the ANN robots at similar times in the simulation (see Fig. 4.16). We thus conclude that the (average) adaptation of the GRN robots is much better than that of the ANN robots.

In conclusion, we observe that the GRN robots survive (much) longer with fewer food sources. GRN robots explore the grid more efficiently (as already shown in chapter 4), while they also seem to evolve alternative strategies, such as prey behavior (as shown here), as adaptations to food scarcity. We think that such dynamic adaptation resembles natural evolution more closely and shows that considering the relevant context and equilibrium is important in evaluating adaptability. In some cases, fitness measured at one particular level (e.g. the fitness of a single organism) is not sufficient to fully describe adaptability, since the entire context is important and the overall adaptation of the population might not necessarily reflect the fitness of the individual.

Based on the design, we assume that the interaction of agents encoded by the GRN is the main reason for the difference in showing prey behavior between ANN robots and GRN robots.   While prey behavior occurs in the ANN robot population, unlike of what happens in the GRN population, it does not seem to be an active choice (adaptation) due to food sources becoming too scarce. The ANN robots in the simulation are expected to achieve a good network model for adapting within the environment. Through evolving the weight parameters of the network that is encoded by the genes, ANN robots may ultimately efficiently reach a good network model for a certain task. However, achieving equilibrium (between the food source availability and the population size) and balancing between multiple unidentified tasks (such as preying, defending, replicating) is still hard for an ANN, especially when such equilibrium has to be reached in a changing context and environment. Although equilibrium can possible be reached by the ANN, it will cost extra time to allow the ANN to evolve such corresponding network structure. In addition, when the environment has changed, the equilibrium may change as well and the previous structure could prevent to reach a new equilibrium point efficiently.

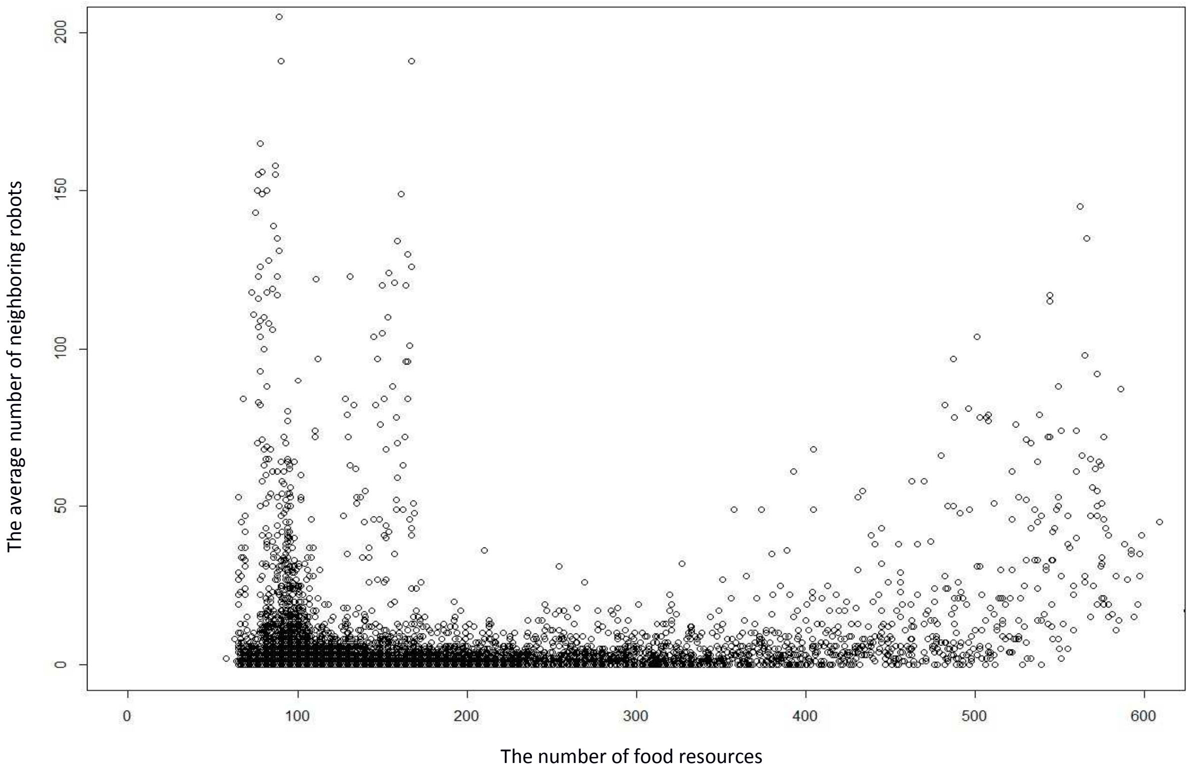
The GRN robots directly connect the environmental information to the particular agents and through the interaction of these agents at the GRN level, the agents and GRN will remain active as long as performance is good. This way, a self-organized GRN will emerge in the robot. The formation of the GRN itself is a process to reaching the equilibrium between agents and each individual agent could be seen as part of the task solution. The key feature or goal of the GRN is not to find the solution procedure for a particular environment or task but to reach a dynamic equilibrium in a changing environment. Through selecting the right agents (without having a direct influence on the system as a whole), GRN robots are more tolerant to changes. Moreover, when the equilibrium has changed because of a different environmental context, the GRN robots only need to activate and re-organize the corresponding agents instead of evolving all connections again. In our GRN controller, if the previous agents become irrelevant in the current situation, the environment will quickly repress them and there is no more interference from them. Based on the interaction of highly modular agents (attached with various environmental conditions), the GRN is more flexible to respond to environmental changes and more sensitive to the equilibrium in the environment.

5.3.2 Complex adaptation in artificial evolution

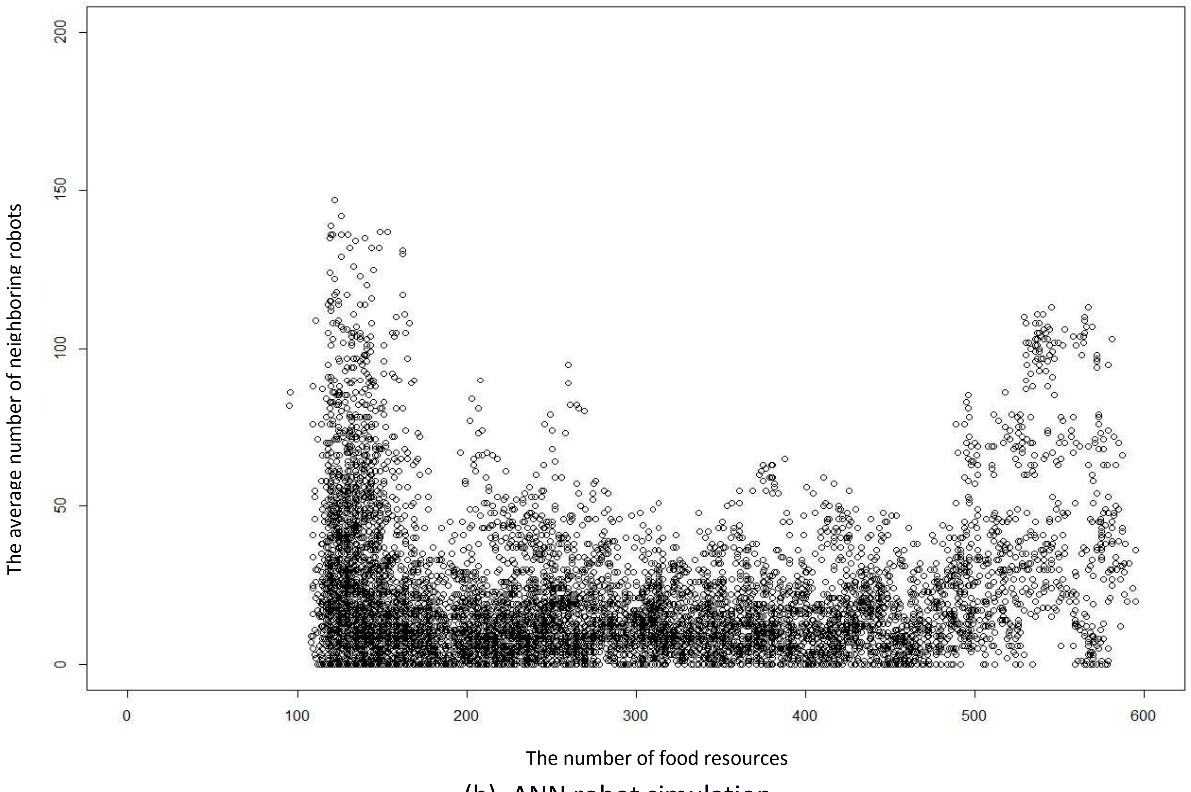
Adaptation in the natural environment usually requires a cooperation and balance of several particular traits rather than just one, which greatly increases the difficulty of a single level evolutionary model to reach a certain adaptation. To allow the evolution of complex adaptation, we adopted a multiple level evolutionary model in the GRN robot. We believe that, with the multiple level evolutionary framework, the system allows different sub-modules (i.e. the agents in GRN, or the robots in population) to independently evolve at the lower level while the cooperation pattern among multiple modules is evolving at the higher level through the interaction of all activated modules. In other words, the system can separately evolve different traits at different evolutionary levels and subsequently evolve the relationships of interaction of these different traits.

In our scenario, successful preying not only depends on the preference of robots and the energy (determining the success of an attack or defense) but is also highly influenced by the position of other robots (as suggested previously). The robot cannot prey others if they are not in the same cell of the grid. Therefore, prey behavior also needs a corresponding moving behavior, otherwise preying will not happen. On the other hand, the food searching behavior also requires corresponding movement behavior and the interesting point is that food searching behavior usually goes against prey behavior. Robots flocking together is ideal for preying but it will be disadvantageous for searching food. The most adaptive strategies seek a balance between prey and food searching. When food is plenty, food searching should be encouraged, otherwise, preying could be more advantageous. Such adaptation is based on the dynamic balance present and is more complex than adaptation based on a certain fixed task scenario.

For tracking the movement behavior of the robots, we have also evaluated all environmental conditions over different stages based on the number of food sources. Since it is difficult to discuss the moving trajectories for all robots during multiple experiments, we have used the average neighbour robot number (the sum of all neighbour robots/the number of robots*100) as a way to represent the movement of robots. This number directly influences the prey and food searching during the simulation.



(a) GRN robot simulation



(b) ANN robot simulation

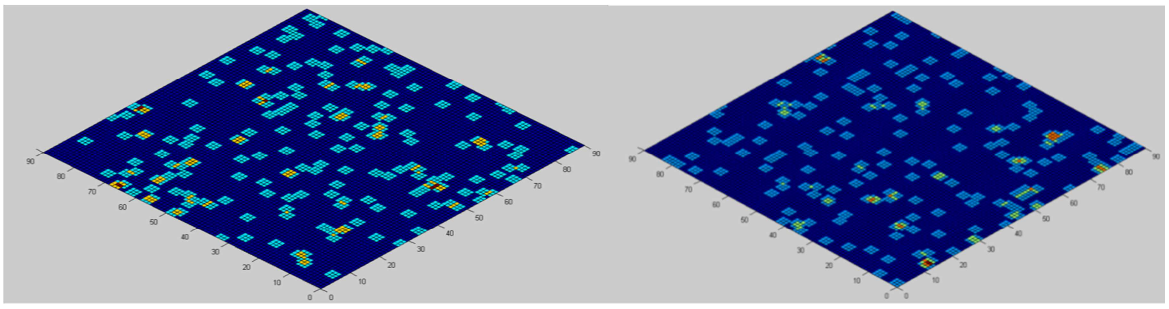
Figure 5.4 Comparison of the average number of neighboring robots in different robot simulations

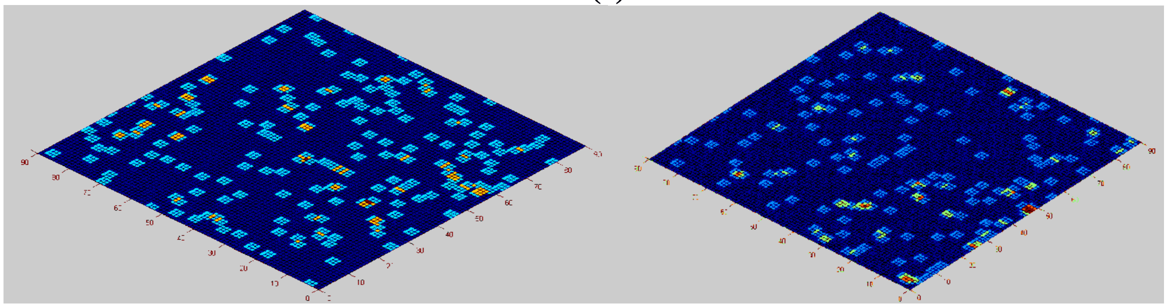
In the comparison (50 plus 50 runs) above, we see that both kinds of robots tend to have more neighbors when food is abundant or scarce. When the food is abundant, the robot's energy level is usually higher and this enhances the reproduction rates of all robots. Offspring will be created in the same cells as their parents, which will increase the number of neighbors. When the food becomes scarcer, we assume that some robots will stop moving for saving energy and some of other robots will tend to flock together for having better prey conditions or for competing for the limited food sources. Although both kinds of robots are following basically the same strategies, they will differ in the detailed behavior. First, the GRN robots spread more efficiently over the grid (see higher), and as a result, GRN robots have fewer neighbors most of the time. More efficient dispersion corresponds to wider exploration area and more efficient searching and thus leads to better food searching abilities. Second, when food becomes scarcer, in some cases GRN robots can start to see having much more neighbors. The interesting fact is that such fluctuations in the numbers of neighbors is also reflected by the prey behavior of GRN robots. When the food is abundant, such correlation between the numbers of neighbors and prey frequency is not as apparent as when there is food scarcity. This implies that such correlation is based on some kind of cooperation rather than random coincidence. On the one side, spreading more evenly over the grid could help the search for food, but on the other side, flocking together could increase competition and prey behavior when this is considered necessary. Moreover, when food becomes scarce, searching food and preying may become both important (compared to the situation when food is abundant, searching food will become more important) and it requires a new balance between the two strategies. Based on the comparison between GRN and ANN robots, GRN robots show a greater fluctuation (larger range data distribution) on both prey behavior and average neighbour number and the fluctuations are more synchronous when food is scarce. These features of the GRN robots imply that GRN robots generally better adapt to a changing environment. Under the same complex scenario, ANN robots shown less fluctuation (variation) when similar changes (food reduced) occur in the environment and usually show weaker adaptation, proven by their considerably less long survival time.

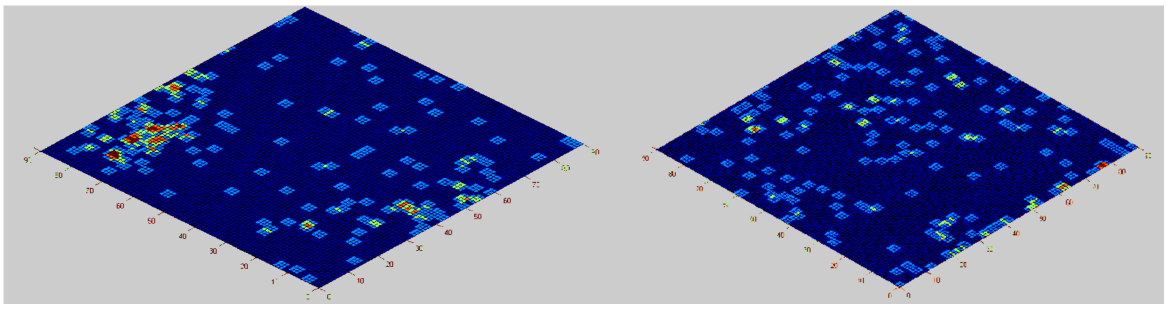
The reason for the more fluctuated behavioral pattern of GRN robots is due to the structure of the GRN. Through the interaction of agents encoded by the GRN, the agents are dynamically activated or replaced until the GRN reaches a stable status (adaptive status for the robot). During this process, the environmental conditions and the relevant modules can gradually connect with the particular agents in the GRN (through feedback mechanisms) and the interaction between agents make all existing agents in the GRN tend to reach a common stable status constantly. When new environmental conditions occur, this will infer the activation of new corresponding agents in the GRN and these new agents may change the current equilibrium. At that time, we will observe a corresponding change in robot behavior. Later, the agents (both new agents and the already existing agents) will tend to cooperate (exist together) and they will reach equilibrium again. Therefore, behavior will fluctuate to adapt to changing conditions. Such fluctuations can happen many times until the GRNs in the robots reach a new equilibrium or, alternatively, the robot dies (when not adapted). Therefore, re-balancing processes are inherent to the GRN robot framework. Contrary, in ANN robots, every new environmental change will directly influence the selection on the whole network and the selection will only have two possible outcomes: either the rewired network fits the new environment or the rewired network does not fit. There is no efficient re-balancing process after rewiring of the network in the individual ANN and the behavioral pattern for a certain ANN (without any change on the genes) is global rather than local or based on the modular organization.

Using interactions at different levels let the system more efficiently reach a balance between multiple traits and make the whole system become more robust to environmental changes. Moreover, it can accelerate complex adaptation. When some modules might not be beneficial right away, the system still has a chance to regulate the cooperation of modules (i.e. switch off inappropriate agents in the GRNs, but still remain the corresponding genes in the genome), instead of entirely fail in the selection process.

In the Fig. 5.5, I present a comparison of the distribution of GRN robots and ANN robots in one example.


(a)


(b)


(c)

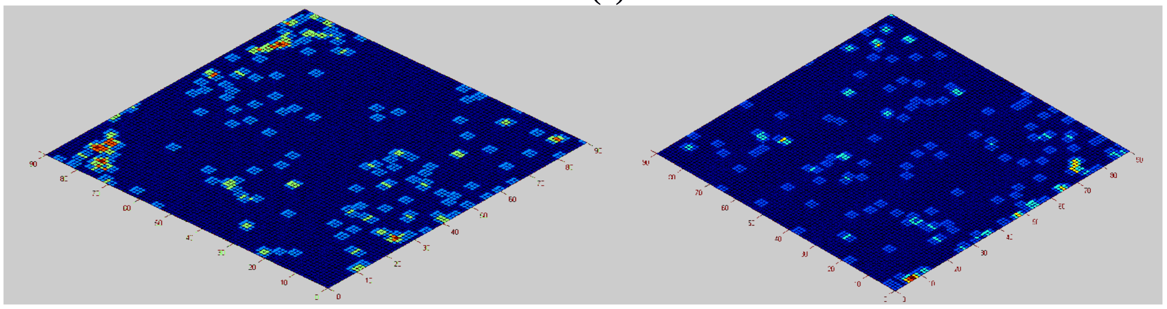

(d)

Figure 5.5 Comparison of the distribution patterns during simulations for ANN (right) and GRN (left) robots.

The diagram presents the distribution of the GRN and ANN robots at different time steps during the simulation. The matrix (i.e. grid) corresponds to the simulated environment. Each cell of the map represents one basic space that can be occupied by a robot. All cells are marked in colors dependent on the number of surrounding robots. Cells in dark blue represent cells where no robots are present in surrounding cells while cells in red represent cells with larger numbers of robots surrounding them. (a) time step 500 (current number of food sources: GRN 324, ANN 455); (b) time step 1500 (current number of food sources: GRN 190, ANN 286); (c) time step 2800 (current number of food sources: GRN 107, ANN 122). For the GRN result (left diagram), we can start observing colonies to be formed; (d) result at time step 3300 (current number of food sources: GRN 97, ANN 105)

As can be seen, at later stages of the simulation, the GRN robots still seem to show some structure in the distribution of robots occasionally, while for the ANN robots, robots seem to be randomly distributed over the grid all the time. Different with being trapped in one place for a long time, the colonies in the simulation dynamically move or change. Therefore, such colonies do not bother the food searching too much.

From Fig 5.5 (c,d), at different time steps, we can see some novel patterns emerging in the GRN population, while for the ANN population, robots always seem to be positioned on the grid in a less structured manner. Furthermore, ANN robots often prefer to stay at the same point for a long(er) time (see also Chapter 4).

However, it is important to note that, for the GRN robots, the specific robot distribution patterns as shown in Fig. 5.5 only emerges when the food sources become scarce. Based on this fact, we could partly explain the large range fluctuation on figure 5.5 (left panel). When such pattern emerges, the neighbor number will quickly increase. However, these patterns are dynamically changing all the time and can quickly varnish so we also observed many low neighbor number records before or after the high records as well.

5.4 Conclusions

In this chapter, we have compared a GRN robot population with an ANN robot population. Based on the results of the simulation, we believe that the agents that are relevant for prey behavior, food searching behavior and moving behavior, closely interact and are not independent from one another. The equilibrium and cooperation between these behaviors in our simulations can be regarded as an example of complex adaptation in a changing environment. In our simulations, such complex situation usually appears at later stages when the robots have consumed most of the food sources and they need to change their strategies (adapt) to survive. As we discussed above, only the combination of a limited number of food sources and large population sizes make prey action a trait for selection and therefore an advantageous adaptation. If food sources maintain to be abundant to the population, simply searching for food is obviously a more easy and efficient option than preying. Therefore, we do not observe the combination of prey and flocking together behavior evolving at the early stages of our simulations when food resources are still abundant.

Chapter 6

Discussion and future prospects

Author contribution

The content of this chapter was written by myself. It resulted from many fruitful discussions with both my promoters and all partners I had the chance to work with during my PhD studies.

Evolution is a complex process that includes numerous interactive causalities[227](we refer to these as interactions) at multiple levels (see chapter 5). Evolutionary studies of artificial evolution are often limited by the use of oversimplified models to describe these interactions. This work aims to use a novel computational framework to mimic the (interaction between) different evolutionary processes at work and to help researchers investigate evolution in a more realistic context. To this end, we modeled evolution and the different evolutionary processes as a complex adaptive system (CAS) with a nested architecture. In our simulations, each CAS model is represented by a bio-inspired (swarm) robot or agent. Through the collective behaviors of robots or agents, we can simulate the different evolutionary processes at play at multiple levels, as in real biological evolution. Furthermore, the novel framework can also deal with a dynamic evolutionary context (a changing environment), which is usually not possible in other simulation platforms. Using our approach, researchers could therefore simulate the dynamic evolution of self-adaptable robotic controllers. On the other hand, such simulations could also be used to identify the evolutionary “relationships of interaction” at different levels. For instance, experiments based on our simulation framework could identify how a few mutations can affect the composition and evolution of GRNs and the cooperation between organisms at later stages. Having implemented more comprehensive contexts and more realistic evolutionary trajectories will allow researchers to have a more holistic view on evolution, at multiple levels and at different time frames. In this chapter, I want to summarize the unique features of this research and briefly discuss what we have learned so far.

6.1 Self-adaptive robot controllers in a changing environment

In this work, I have developed several kinds of robot controllers based on the bio-inspired principles of evolution. These controllers are able to evolve the robot’s behaviors to adapt to environmental changes. As described, with adopting these controllers, environmental input activates the corresponding gene regulatory network on the artificial genome of the robot while the corresponding gene products are then represented as agents in the robot. The collective behavior of these agents then determines the phenotype of the robot (see chapter 2). Also, the feedback of the robot’s behavior will influence the adaptation of these embedded agents. Through the signals from the environment and the feedback, each agent will evolve and form part of a more ‘adaptive’ GRN. Actually, the evolving GRNs in the robot form the main body of the controller in each robot. We have shown that, in unpredicted and unknown environments, this approach can efficiently help the robot to improve its adaptability[117].

6.1.1 Controller for area exploration and collision avoidance

This controller that we have used for area exploration and collision avoidance is based on the E-puck robot (Fig 4.2) while the experiments were implemented in the player/stage robotic simulator (see chapter 4, section 4.1). For a giving unknown maze environment, the controller could develop a real time strategy to avoid collision and at the same time explore the area maximally. The results showed that the robots equipped with our (simplified) GRN controllers had a certain self-learning ability.

Based on our simulations, it was shown that the controller could self-adapt to new situations in the environment. The controller mimics biological gene regulation and gene evolution through environmental cues that activate an artificial GRN in the robot after which the GRN controls the behavior of the robot.

6.1.2 Controller for the multiple robot aggregation

This controller that has been developed for the aggregation of swarm robots was based on the Symbrion robot while the simulation experiments were performed by both the Robot3D simulator and the 2D cellular automaton simulator. Except for the simulation experiments, we also tested the controller on single real Symbrion robots for potential future practical applications. This version of the controller could determine the interactive behavioral pattern of each swarm robot based on the local environment after which the collective behavior between these swarm robots will self-organize into the most suitable robotic organism to achieve a certain task. The main function of the controller is, based on the particular environmental condition, to determine the shape of the robotic organism and the possible aggregation process of each independently involved swarm robot. Different tasks or environments require different kinds of robotic organisms, but these mapping relations are usually unknown to people and robots a priori. Through feedback of the previous aggregation, our controller can optimize the aggregating behavior for each robot. This way, every robot can learn how to help making a more adaptive multiple robot organism under a particular environment. The basic mechanism of this controller is inspired by cell specification, gene regulation and gene evolution.

6.1.3 Controller for complex adaptation

This controller follows similar principles as controllers described previously, but the simulation scenario has been separated from the particular tasks and robot platforms. This controller is designed for all sorts of complex tasks under unknown and changing environments. We have tested this controller in an artificial life simulation with multiple virtual swarm robots and compared the performance of this controller with other ANN based controllers. Based on our simulations, we could show that the controller with the artificial GRN leads, on average, to a better adaptability than the ANN based controller.

6.2 Artificial evolution and its nested architecture

Starting from the artificial genome and GRNs developed in this study we introduced a particular embedded genetic regulation and evolutionary process in each robot. During the whole simulation, we simulate evolution at different levels (the genetic level, the organism level and the ecological level) and all those levels are simultaneously evolving. After given the simulated evolution this nested architecture, we can not only observe the effects of the genetic changes to a particular robot but can also identify the correlation between the genetic changes and the changes of the collective behavioral patterns. Furthermore, all models in our simulation are self-organized through the interaction of the different interactive components. In general, in our simulations, artificial evolution has the following features[117]:

- **Agent-based genetic regulation and gene evolution in the single robot:** In each simulated robot, there is one evolutionary process based on the artificial genome and gene regulation. This also represents the bottom level of our artificial evolution while the other evolutionary processes (such as the interaction between robots and between robots and the environment) are all based on it. Through the interaction and expression of genes, the corresponding gene regulatory networks are also dynamically formed as a self-organizing process. The phenotype of the robot is based on the dynamics of the GRN while the activated genes on the genome will be selected by the adaptation of their corresponding agents.

- **Niche construction and symbiosis:** Based on the different phenotypes of the robots, individual robots can establish various relationships with others in their local environment and such relations could have a corresponding effect on the adaptation of the robots at later stages. Such relationships could become stable after evolution and the corresponding stable local environment is called a niche. Such niches are evolving as well as the phenotype of the robot.

- **Evolving context in the virtual ecosystem:** At a larger scale, the different group of robots also interact with each other at the population level. Like different species in a biological ecosystem, the role and functions of these different groups of robot are various and these features are dynamically evolving. At the top level of our simulations, we identify the emergent behavioral patterns in the population and keep track of all events of the interactive behaviors for each robot.

- **The holistic view on evolution:** Simulating the various evolutionary processes at different levels synchronically allows us to identify the effect of the different evolutionary processes on the whole of evolution and the relationships of interaction between these evolutionary processes. This knowledge may give us a chance to reveal the potential mechanisms of evolution or inspire us to invent more efficient approaches for improving the adaptability and evolvability of the computational system.

- **The evolutionary context based on the interaction**: In our artificial evolution experiments, the evolutionary models are self-organized and form a developmental system through their interactions. Therefore, all events may experience a different context at different times (like genes may be under different selection or have different mutation rates at every time step). Compared with predefining the context (predefine explicit selection rules or mutation rate) in the simulation, it avoids the problems of a missing or (too) artificial context. The results of the simulation are therefore more realistic.

6.3 Interaction-based Evolution at multiple levels

In our experiments, we have simulated the nested architecture that is commonly observed in biological evolution (see Fig. 1). From our point of view, biological systems are also complex adaptive systems and based on complex theory. The reason for such architecture existing in nature is related to the origin of complex systems. In the natural environment, when enough interactive components are present, these components always tend to construct a complex adaptive system through complex interactions [228]. Therefore, all complex adaptive systems in nature are inherent to a nested architecture.

Another inherent feature of complex adaptive systems that should be mentioned is that every complex adaptive system always dynamically adapts to its environment [229]. When we consider these two facts, we actually could infer that such nested architecture should be the common feature of all natural evolutionary systems since all such systems could be regarded as complex adaptive systems [230].

An important implementation in our work on artificial evolution is the extension of the concept of evolution. The commonly agreed consensus defines evolution as a developmental process based on selection. Initially this definition applied to all biological systems but later it gradually extended to non-biological objects like computer programs, ecological niches, memes and so on. This extension was not only because these follow similar principles as biological evolution but also due to the fact that at least some of them are part of an evolutionary context. Actually, the evolutionary context determines the selection pressure and is closely connected to all aspects of evolution. Therefore, an improved model or concept of artificial evolution should include the environment and context of evolution as evolving components since they are evolving as well. Another new concept in our approach of artificial evolution has to do with the selection of evolution. As discussed above, evolution is based on selection, but the concept does not explain the reason of selection because it has been assumed to depend on the environment and the environment is usually predefined in artificial evolution experiments. Based on our new model, we suggest that selection actually comes from the interaction between different parts (different levels) of evolution while all components of evolution form the environment. This explains and emphasizes that the different parts of evolution can have different selection pressures while such selection pressures can be evolving as well, just as any other evolutionary component.

In contrast to the traditional artificial evolutionary model, evolution in our new model has to be regarded as a kind of co-evolutionary process acting at multiple levels, and should be motivated by the interaction among all interactive components rather than a single evolutionary process based on selection in a given environment. To emphasize this difference is important because it leads to many new perspectives on the view of artificial evolution and it could help us to design a better evolutionary framework in both the simulation and engineering fields. In the next paragraphs, I would like to give some examples to show how the new model could help us in future research.

First, the traditional evolutionary model usually imposes evolutionary context by using random events or predefined rules, such as for example gene mutation rate or the distribution of food in the environment, the fitness evaluation and so on. However often, the simulated cases are far removed from reality, while in real life also the evolutionary context (environment) keeps changing at all time. With a holistic approach, the solution to the problem above is to replay the context based on rules of interaction among components. Compared to identify the concrete context, the rules of interaction and possible components in evolution are easier to be identified. Since all contexts are based on dynamic interactions in evolution, replaying these dynamic contexts from interaction rules is a more feasible approach to predict the real case than using random or fixed context. Considering the complexity of adaptation and subsequent changes can have a domino effect in evolution, the effect on the ultimate result could be huge. Furthermore, new model also give us a more open view on some phenomena such as the orphan genes.

Second, the new evolutionary model could be applied to improve artificial evolution in programs. In the artificial system, there is often a tradeoff between evolvability and complexity. However, this contradiction is rarely observed in biological systems. Biological systems have much better evolvability and adaptability than any artificial system. Based on our view, we assume that these differences between biological and artificial systems are partly explained by recently discovered new features of evolution. There is an essential principle that has been observed not only in genetic evolutionary process, but in all kinds of biological evolutionary processes. All these evolutionary processes are based on interaction. This principle has been investigated by Margulis [231], Duffin [232], Watson [233] and others. Numaoka [234] and Webster[235] for instance concluded that all evolutionary units, from replicating molecules to organisms, have "relationships of interaction" with others and there is no system that can survive independently as an autonomous form in evolution. These relationships of interaction have been evolving all the time through the interaction between evolutionary units and they all contributed to forming new evolvable units at higher levels during evolutionary transitions. Based on each particular evolutionary unit, the evolutionary process has been directly driven by two kinds of forces. On the one hand, we distinguish an extrinsic force as the interaction relationships of the units that have evolved in an external environment. On the other hand, we consider the inside of the unit that we refer to as the intrinsic force. The intrinsic force is represented by the evolved interactive pattern of internal components and it directly determines the characteristics of the unit. The transitions in natural evolution discussed above actually are linked to both kinds of evolutionary forces. For example, from the single cell organism to the multicellular organism, the multicellular organismic form presents a remarkable progress regarding adaptability and complexity. However, the single cell form is still contained in the new multicellular organism and the interaction of all internal single cells regulates the growth of the multicellular organism. Such nested architecture is ubiquitous in all evolutionary units of nature and the listed transitions could be regarded as the emergence of new adaptive forms at higher levels. The new form (i.e. the multicellular organism) has been directly evolved from the interaction of previous forms at lower levels or smaller scales (i.e. the single cell), so we also could say that it evolves as the extrinsic environment of small-scale forms and the higher level or larger-scale form is an adaptive pattern of a group of relationships of interaction. Furthermore, the small-scale forms may also include the same nested architecture and they are dynamically evolving by an intrinsic force. Apparently, these two kinds of evolutionary forces are closely connected with each other. As Watson and Pollack stated [236], this composition of pre-adapted extant entities into a new system is a fundamentally different source of variation from the gradual accumulation of small random variations and it has some interesting consequences for issues of evolvability. Based on the paper of Clune et al. [123], a key driver of evolvability in biological organisms is the widespread modularity of networks — their organization as functional, sparsely connected subunits. Clune et al. [123] also show that an imposed selection to maximize network performance and minimize connection costs could improve the modularity and adaptability of systems. As an extension to Clune's hypothesis, we suggest that such modularity is not only based on the imposed selection from the external environment but is a kind of inherent attribute based on the iteration of the nested evolutionary processes discussed below. Self-organized evolutionary units organize into new systemic forms at larger scales through interactions. After the formation of the larger-scale new systems, those small-scale unit forms become "functional, sparsely connected subunits" and well-adapted modules. This way, we can explain major transitions in natural evolution, the continuous increase in complexity and the correlations among modularity, evolvability and adaptability based on the same evolutionary mechanism. To adapt to new environments, the natural system always tends to minimize the impact of changes through the interaction of modules that are part of the nested architecture. In other words, compared with the random rewiring of a part of the whole system, evolving the new interaction relationships among a few modules could greatly reduce the necessary changes to reach a novel adaptive status. As a result, few introduced changes will influence the relationships between the different modules but will leave the inside of the modules unaffected. A similar idea has recently been mentioned by Marc W. Kirschner as well. In his papers [237,238], he defined such multiple level modularity as facilitated variation and discussed the advantages of facilitated variation to evolution.

Moreover, in natural evolution all modules are pre-adapted entities, which means that those self-organized modules possess a certain self-adaptability. This self-adaptability potential of modules makes the whole system more evolvable than systems only including predefined modules because the re-adaptation process here is not only based on random mutations but also depends on the context ("memory") of the modules. The new adaptive status emerges as part of interaction through a developmental process while the process is regulated through feedback from external and internal environments. The current artificial evolutionary models suffer from the "cost of complexity" because these models usually do not consider the above-mentioned interactive context and nested architectures. Without those, artificial models will lose the corresponding self-organized modularity and developmental adaptation that biological evolutionary systems have. In our study, we presented a novel bio-inspired evolutionary framework that aims to mimic the interactive context and nested evolutionary processes observed in natural evolution, and apply this to artificial evolution. We believe such a framework could significantly improve the adaptability and evolvability of artificial evolutionary systems [117].

Third, combining with the assumed multiple levels evolutionary model, interaction and nested evolutionary architecture could better explain many questions in evolution that we have met with the traditional evolutionary model. For example, the origin of modularity in biological evolution has been debated for decades in evolutionary biology. Based on our new model, this can be easily explained by the two-side selection (interaction) and evolution at different levels. Because there are different evolutionary processes self-organized at multiple levels , the various interactions will evolve the different patterns on each level separately. At a higher level, cooperation of evolutionary components at lower levels can be recognized as the particular modules. When stable cooperation between lower level evolutionary processes have enough time to be evolved as a module, new evolutionary components at higher level evolutionary processes are possible to self-organize through interactions as discussed above. As a result, we can observe biological systems to include an inherent modularity since they all evolved according to this bottom up style. Another question could be explained by interaction between multiple levels is why we observe sudden extinctions [239] company with sudden eruptions [240] on the majority of species in history. Elimination on higher level will encourage the mutual components surviving at lower level and the consequent changing environment usually shuffle all none-mutual components as well. More examples like red queen hypothesis [241] and so on also could be well explained by our new model while they are difficult to reconcile based on single level evolutionary processes.

6.4 Limitations and future works

Complex biological systems are difficult to understand or describe by simple models. Biological organisms obey the laws of physics and chemistry, but these basic laws do not explain for example their behavior. Each component of a complex system participates in many different interactions and these interactions generate unforeseeable, emergent properties[242]. Our research aims to provide an approach that could help people to simulate such complexity from a bottom up way and investigate the relationships of interaction between the various evolutionary components at different levels. Based on our proposed approach we observed that adaptive patterns self-emerged at multiple levels of the evolutionary process and identified the impact of the nested architecture in evolution. However, to build a sophisticated simulator for evolutionary biology, our artificial evolutionary framework is still limited and the simulations need to be further improved for investigating real biological evolutionary phenomena.

The main limitations of our current simulations are the following. First, the scale of the simulations is still limited by the computational resources available. In the experiments described in this thesis, both the population sizes and the genome length are limited by the amount of computer memory available and the speed of the computational platform. To allow fast enough simulations, we also have to simplify the interactive rules and the functions of the agents. These limitations may have an effect on the formation of the collective behavioral pattern and the functionality of the evolutionary operations. To remedy these at least partly, I have recently adapted our simulator to make it compatible with paralleling computing platforms (cluster computer) and can thus run on multiple processors[243]. With the new version of the simulator, we can hopefully greatly extend the population size and flexibly evolve the genome length during the simulation.

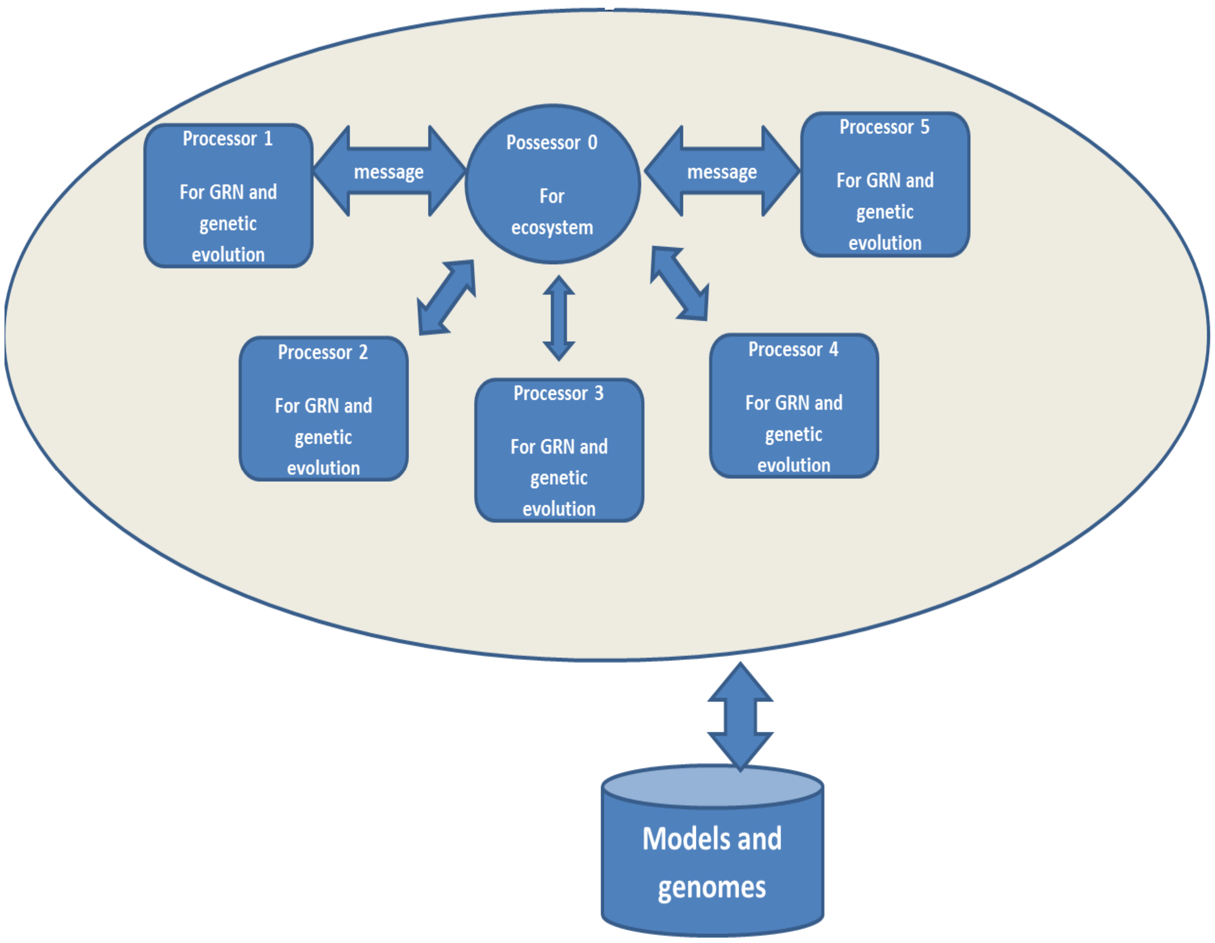

Processor 1
For GRN and genetic evolution
message
Possessor 0
For ecosystem
message
Processor 5
For GRN and genetic evolution
Processor 2
For GRN and genetic evolution
Processor 3
For GRN and genetic evolution
Processor 4
For GRN and genetic evolution
Models and genomes

Figure 6.1 The parallel computing framework for the GRN robot simulations

As shown in Fig 6.1, in the parallel computing framework, the artificial life simulation and the GRN simulations in the robot will be divided over different processors. All processors share the same basic models and knowledge of the interactive rules and each processor can communicate with the others by sending messages. This framework ensures the consistency of the whole simulation while it allows the simulation to extend the number of simulated robots by adding extra processors.

The second limitation is that the artificial genome and the basic interactive rules used in our previous simulations are still too simplistic and may need to be made more complex to simulate real biological gene evolution in organisms. For example, in the previous simulations, the agents always start reading the genome sequence from the same start point. In later versions of our simulator, we have changed this and allow the agents to randomly start reading the gene sequence from any location of the genome. This new change makes the location of gene having less effect on gene expression but binding motifs having more effect. In addition, we also have changed the structure of the genes to allow a more flexible and realistic gene regulation (for instance, multiple binding sites for one gene, various gene products based on a different regulation context and so on). The details of the gene regulation mechanism may not be very necessary to evolve the adaptive virtual organism in computing science, because these details may correspond to the particular evolutionary context in biological evolution, but they are critically important to researches that aim to investigate particular biological evolutionary phenomena. Moreover, the basic interactive rules between the agents also need to be further improved in the future. In the current simulating framework, the agents represent the various gene products or chemical compounds encoded for by the GRN. The possible chemical reactions between those ‘gene’ products are also complex. To better describe the possible interaction between various products (such as proteins or metabolites), we have started developing a knowledge base that can be used to define certain rules used by the different agents.

With the improvements discussed above, future work of this research is twofold. First, we will extend the artificial evolution at a larger scale, thus with larger population sizes and longer time frames. In this case, we hope to be able to observe more complex self-organizing patterns emerging at multiple levels and thus we hope to evolve more sophisticated artificial evolutionary systems. Second, we will further try to make our simulations more realistically. The simulation platform and the agents still need to mimic biological evolution more closely, we think. For instance, we hope to also include information on the metabolism in our simulations. Chemical reactions and cellular signaling in cell metabolism will be represented by particular agents and the corresponding reaction mechanisms will be stored in the corresponding knowledge base. We hope that by adding this particular knowledge, we could simulate biological evolution even more realistically.

Appendix A
Academic CV

Personal information

Name: Yao Yao

Address: Eikenmolenwijk 28, 9820, Merelbeke, Belgium

Email: yaocong111@gmail.com

Mobile: +32 470043302

Date of birth: 13/08/1980

Place of birth: He Nan, China

Education

2009-2015 Doctor of Science, Bioinformatics, Ghent University

2007-2009 Msc in Bioinformatics, Dublin City University, Dissertation: "An Agent Based Simulation of CyanoBacterial Blooms"

2003-2007 BSc (Hons) in Computer Science, Dublin Institute of Technology, Dissertation: "Semantic E-learning System Based on Multiple Mobile Agents"

1999-2001 Joint Diploma in International Financial, HE Nan Education College (China) & Marshall University (USA)

Publications

Yao, Y., Marchal, K., Van de Peer, Y. (2014) Improving the adaptability of simulated evolutionary swarm robots in dynamically changing environments. PLOS One

Yao, Y., Marchal, K., Van de Peer, Y. (2013) Using novel bio-inspired principles to improve adaptability of evolutionary robots in dynamically changing environments. ECAL 2013, Italy.

Yao, Y., Baele, G., Van de Peer, Y. (2011) A bio-inspired agent-based system for controlling robot behaviour. Proceedings of the IA - 2011 IEEE Symposium on Intelligent Agents organized in IEEE Symposium Series in Computational Intelligence 2011 Paris, France.

Yao, Y., Baele, G., Van de Peer, Y. (2010) Book chapter 4; Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution

Kernbach, S., Hamann, K., Stradner, J., Thenius, R., Schmickl, T., van Rossum, A. C., Sebag, M., Bredeche, N., Yao, Y., Baele, G., Van de Peer, Y., Timmis, K.N., Mokhtar, M., Tyrrell, A., Eiben, A. E., McKibbin, S. P., Liu, W., Winfield, A. F. T. (2009) On adaptive self-organization in artificial robot organisms. Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns 33-43.

Selected Meeting

12th European Conference on Artificial Life 2013 (Oral talk)

IEEE Symposium Series on Computational Intelligence - SSCI 2011 (Oral talk)

IEEE Congress on Evolutionary Computation (CEC) 2010

Benelux Bioinformatics Conference (2009) (Poster)

IEEE Congress on Evolutionary Computation (CEC) 2009

Selected training

<u>Statistical thinking and smart experimental design</u>

VIB research training course

<u>How to write a winning Grant proposal</u>

VIB research training course

<u>Advanced academic English course (writing skill)</u>

Ghent University doctoral schools course

<u>Advanced Academic English (conference skills)</u>

Ghent University doctoral schools course

Appendix B
ANN in this research:

Regarding the ANN robots in our simulation, we keep the same settings for the ecological environment and actuators but, in every robot, have replaced the gene regulation implementation with an artificial neural network. Instead of an emerging GRN, the genotype-phenotype mapping is an ANN that is constructed by the artificial genome of the robot. A similar robot framework has been introduced before by Floreano and Keller [103]. However, our implementation differs from the one of Floreano and Keller [103], with respect to 1) The genome of the ANN based framework is based on direct encoding and each gene on the genome provides an explicitly defined effect to the corresponding ANN; 2) The ANN is a fully connected network and we only evolve the vector and weight of each connection. The reason of fixing the number of nodes of the ANN is to make a better contrast with the GRN robot (more discussion in section 2.4.3 about dynamic regulation). The constructed ANN in each robot will receive the environmental inputs and then determines the behavior of the robot through output signals. For the rest of this section (see also Chapter 4.3, Chapter 5), we will refer to this kind of robots as ANN robots (compared to the GRN robots).

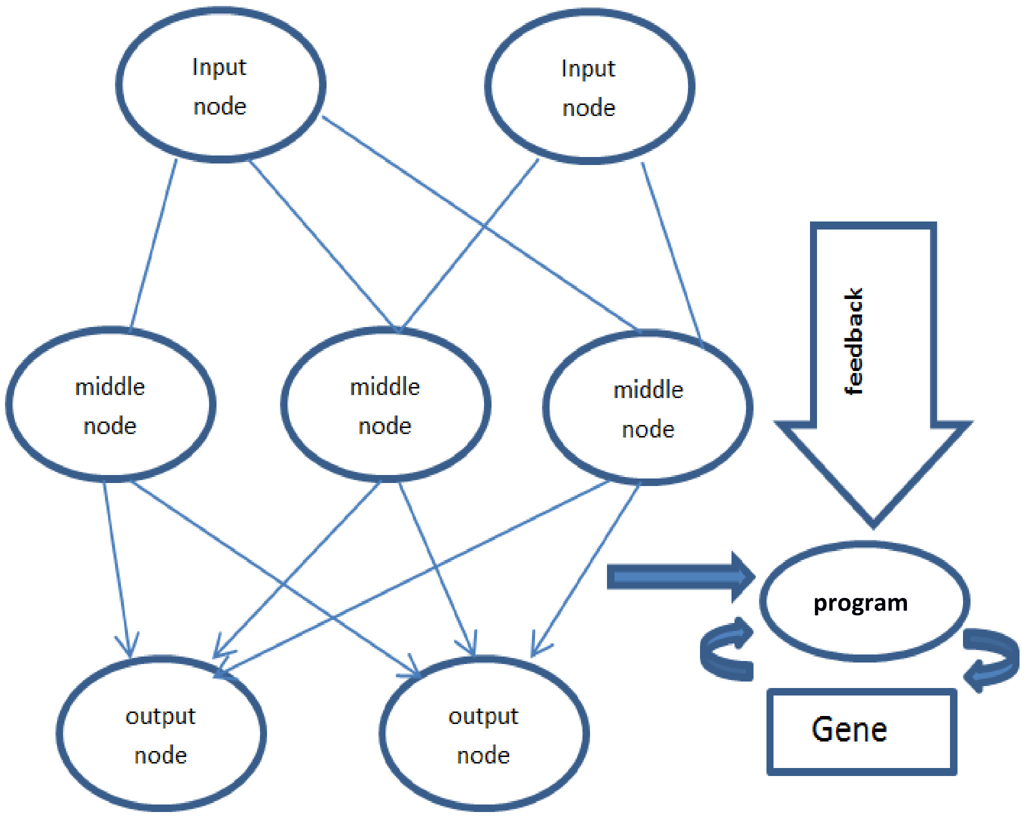


Figure S.1 ANN framework in the simulation

The connection between input node and middle node is given by genes, which is the same as the mapping between the signaling agent and inputs

14 output signals * 25 middle node = 350 edges

On each edge, one learning program is able to optimize the weight parameter on the particular edge based on the feedback of robot. When the feedback value has reached a particular threshold, the learning program will change the vector and value of the weight parameter. The changing rules for redefining the weight parameters are encoded in the genome of ANN robot and these rules are evolved during the simulation (mutation and gene recombination).

Appendix C

More details about the GRN agent's loop:

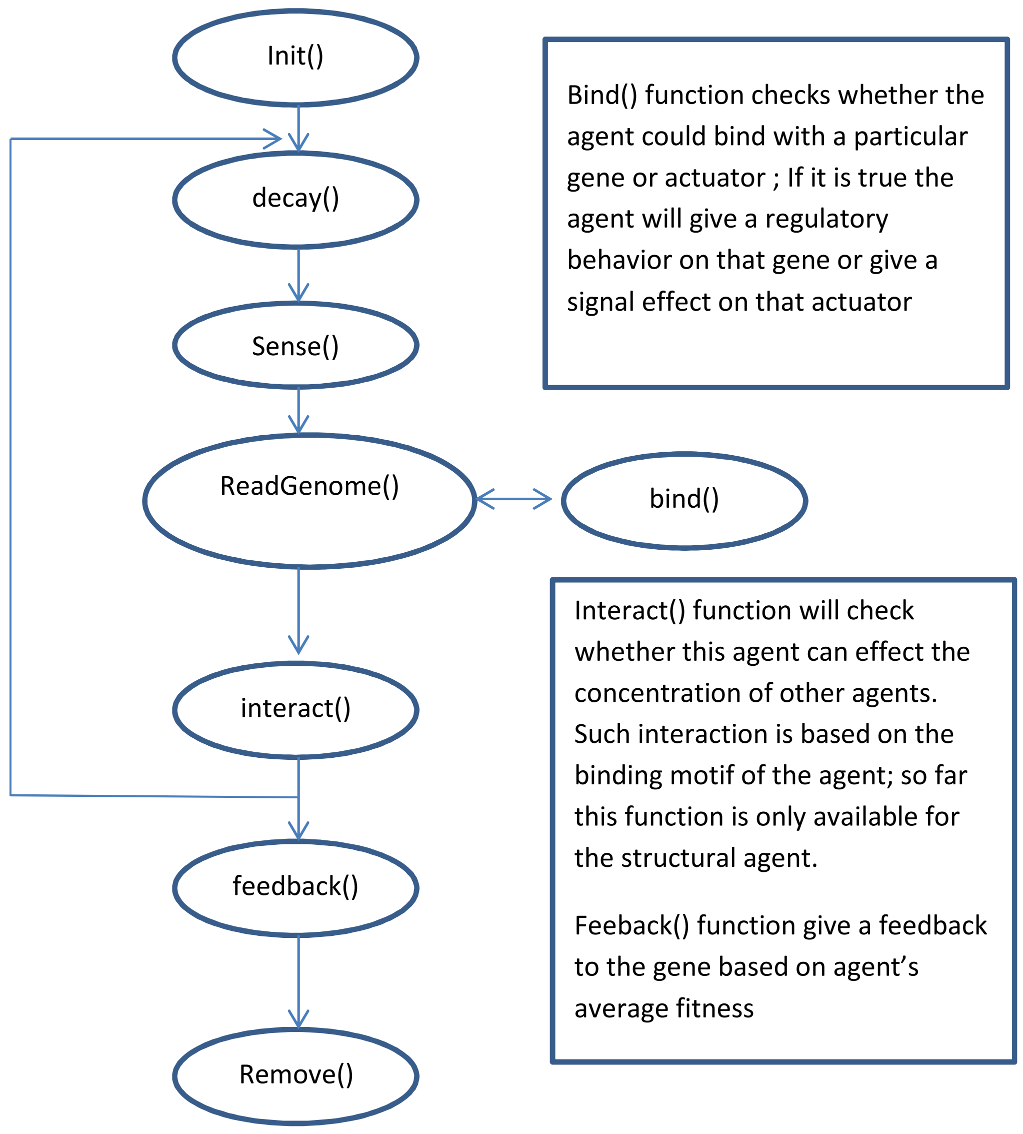

Init()
decay()
Sense()
ReadGenome()
bind()
interact()
feedback()
Remove()
Bind() function checks whether the agent could bind with a particular gene or actuator ; If it is true the agent will give a regulatory behavior on that gene or give a signal effect on that actuator
Interact() function will check whether this agent can effect the concentration of other agents. Such interaction is based on the binding motif of the agent; so far this function is only available for the structural agent.
Feeback() function give a feedback to the gene based on agent's average fitness

Figure S.1 The life circle process of an agent in GRN

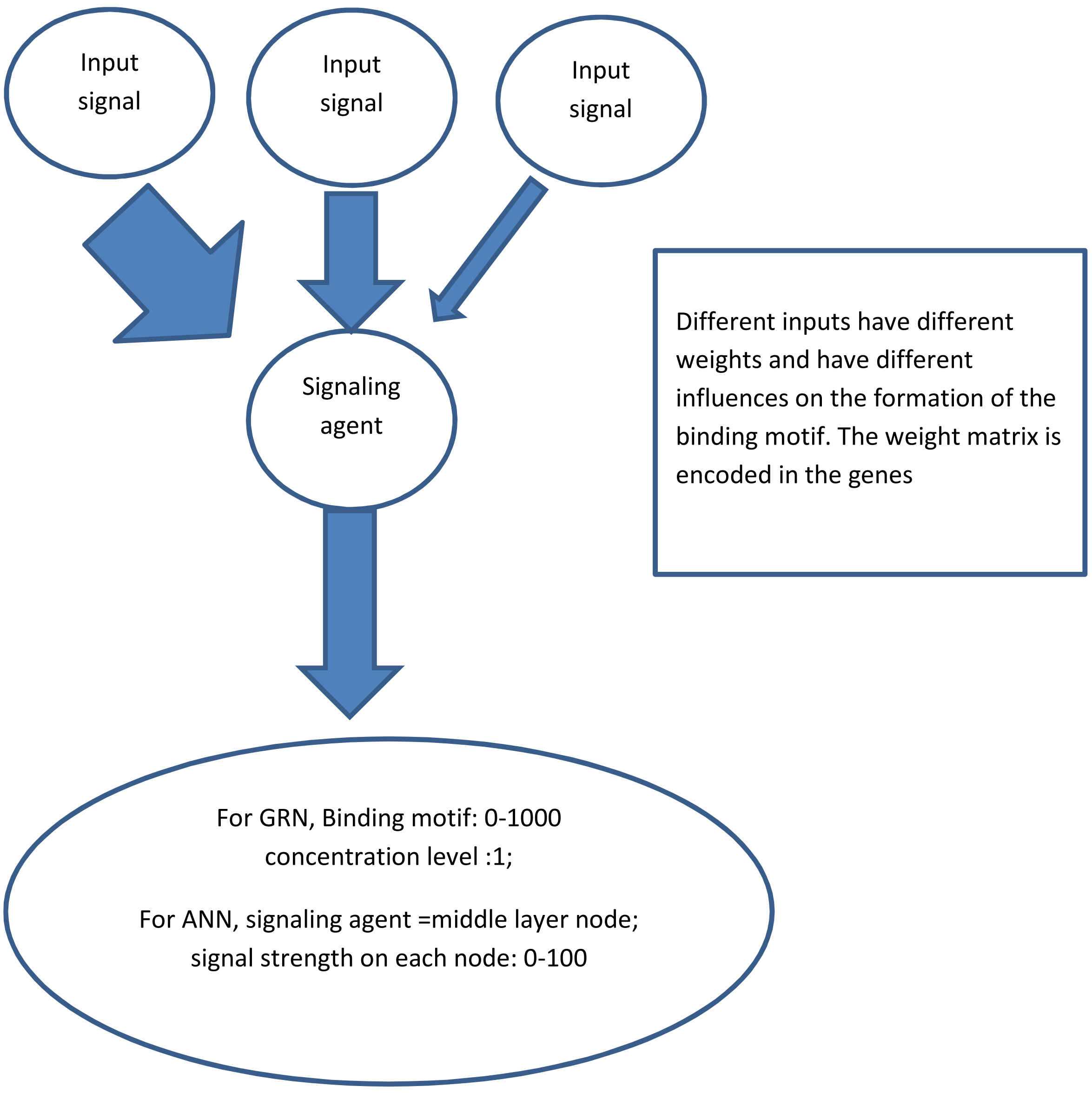

Input signal
Input signal
Input signal
Signaling agent
Different inputs have different weights and have different influences on the formation of the binding motif. The weight matrix is encoded in the genes
For GRN, Binding motif: 0-1000 concentration level :1;
For ANN, signaling agent =middle layer node; signal strength on each node: 0-100

Figure S.2 An example of the signal path in the GRN

Appendix D
Pseudo codes:

Pseudo code for the single robot in artificial life simulation:

```
Run LoadGenome() function random create or load artificial genome from other robot;

If loaded the genome from others, running a mutation operation on the genome;

While condition:the robot is alive;

    1.  Updating the fitness of robot based on the previous performance;
    2.  Sensing the environment and get sensor inputs $S_1,S_2…S_{10}$ ;
    3.  Creating the signaling agents based on the  sensor inputs;
    4.  Running all agent sub-loops in the robot ;
    5.  Checking the output signals: $T_{1,}T_{2….}T_{14;}$
    6.  Using the output signals to set the parameters of actuators;
    7.  Checking whether the robot has enough energy preform the behavior of
        actuators;
    8.  If energy is ok
    9.  Then implement the behavior and cost energy;
    10. Else do nothing;
    11. Check the energy and record the behaviors;
    12. Calculating the fitness of robot for next step;
    13. Updating the information of robot to simulation;
    14. Checking whether the robot has enough energy to live or life time has reach
        the threshold;
    15. If true out of while loop, robot is dead;
End while;

Delete robot and release memory;
```

Pseudo code for the artificial life simulation loop:

```
Initialize the robots and foods on the map;

While conditions: there are alive robots and the simulation is not over ;

    1. Check the simulation time and change the environmental
       parameters every 400 time steps (the energy cost for live and move,
       the food grow speed);
    2. Checking each remain food and decide whether the food can
       replicate itself based on the grow speed;
    3. Running all robot sub-loops and updating the situation of foods and
       robots;
    4. Referee the interaction behavior of all robots;
    5. Read the updates of each robots and store the information;
    6. Output the robot information to the files at every 10 time steps;
End while;
```

Pseudo code for the self-organizing simulation loop (for the simulation in section 4.2):

```
Initialize the robots on the map;

While conditions: there are robots waiting for aggregation and the simulation
is not over ;

    1.  Check the simulation time and change food distribution;

    2.  Running all robot sub-loops and updating the situation of robot;

    3.  If the position of aggregated part of robot organisms are overlapped
        with the food distributed position, the organism can have a certain
        energy reward;

    4.  If the swarm robot is in the range for aggregation;

    5.  Then the swarm robot will become aggregated robot;

    6.  All aggregated robot organisms cost a certain energy based on the
        number of aggregated robots

    7.  If the aggregated organism has too low energy;

    8.  Then its all aggregated robots become swarm robots again and the
        organism will be removed;

    9.  Read the updates of each robots and store the information;

    10. Output the robot information to the files at every 10 time steps;

End while;
```

References

1. Floreano D DP, Mattiussi C (2008) Neuroevolution: from architectures to learning. EvolIntel 1: 47-62.
2. Zihlman A (2001) The evolution of culture: An interdisciplinary view. American Journal of Archaeology 105: 538-538.
3. Kendall G (2014) Evolutionary Computation in the Real World: Successes and Challenges. Knowledge and Systems Engineering (Kse 2013), Vol 2 245: 5-5.
4. Bennett KH, Bradley S, Glover G, Barnes D (2004) Software evolution in an interdisciplinary environment. Eleventh Annual International Workshop on Software Technology and Engineering Practice, Proceedings: 199-203.
5. Antweiler C (1989) Long-Term and Directed Cultural-Change, an Overview of Interdisciplinary Evolutionist Trends with Regard to Ethnological Research on Cultural-Evolution. Anthropos 84: 564-569.
6. Dickins TE, Rahman Q (2012) The extended evolutionary synthesis and the role of soft inheritance in evolution. Proceedings of the Royal Society B-Biological Sciences 279: 2913-2921.
7. Shafiei E, Thorkelsson H, Asgeirsson EI, Davidsdottir B, Raberto M, et al. (2012) An agent-based modeling approach to predict the evolution of market share of electric vehicles: A case study from Iceland. Technological Forecasting and Social Change 79: 1638-1653.
8. A. E. Eiben SK, Evert Haasdijk (2012) Embodied artificial evolution. Evolutionary Intelligence 5: 261-272.
9. Gero JS, Fujii H (2000) A computational framework for concept formation for a situated design agent. Knowledge-Based Systems 13: 361-368.
10. Hamilton WD (1963) The evolution of altruistic behavior. The American Naturalist 97: 354-356.
11. Hamilton WD (1964) The genetical evolution of social behaviour. Journal of Theoretical Biology 7: 17-52.
12. Pittendrigh CS (1958) Adaptation, natural selection and behavior.
13. Slobodki.Lb (1966) Williams Gc - Adaptation and Natural Selection . A Critique of Some Current Evolutionary Thought. Quarterly Review of Biology 41: 191-&.
14. Novak VJA (1983) The Selfish Gene by Dawkins,R. Zhurnal Obshchei Biologii 44: 415-419.
15. Mcdonald JF (1983) The Extended Phenotype - the Gene as the Unit of Selection - Dawkins,R. Bioscience 33: 718-718.
16. Gould SJ (1997) Darwinian fundamentalism. New York Review of Books 44: 34-37.
17. Ahmed E EA, Hegazi AS (2005) An Overview of Complex Adaptive Systems. Mansoura J Math 32: 6059.
18. Lovelock JE, Margulis L (1974) Atmospheric Homeostasis by and for Biosphere - Gaia Hypothesis. Tellus 26: 2-10.
19. MacLennan B (2007) Evolutionary psychology, complex systems, and social theory (Stephen Turner). Soundings 90: 169-189.
20. Moller-Levet CS, Archer SN, Bucca G, Laing EE, Slak A, et al. (2013) Effects of insufficient sleep on circadian rhythmicity and expression amplitude of the human blood transcriptome. Proceedings of the National Academy of Sciences of the United States of America 110: E1132-E1141.
21. Lopes M, Guilleminault C, Roizenblatt S, Poyares W, Tufik S (2007) Cyclic alternating pattern expression during nrem sleep across puberty. Sleep 30: A86-A86.
22. Morris JJ, Lenski RE, Zinser ER (2012) The Black Queen Hypothesis: Evolution of Dependencies through Adaptive Gene Loss. Mbio 3.
23. Farmer JD, Packard NH, Perelson AS (1986) The Immune-System, Adaptation, and Machine Learning. Physica D 22: 187-204.
24. Maher BA (2002) Uprooting the tree of life. Scientist 16: 26-27.
25. Ranade SS, Lin YC, Zuccolo A, Van de Peer Y, Garcia-Gil MD (2014) Comparative in silico analysis of EST-SSRs in angiosperm and gymnosperm tree genera. Bmc Plant Biology 14.

26. Ben Ali A, De Baere R, De Wachter R, Van de Peer Y (2002) Evolutionary relationships among heterokont algae (the autotrophic stramenopiles) based on combined analyses of small and large subunit ribosomal RNA. Protist 153: 123-132.
27. Monteagudo A, Santos J (2007) Simulated evolution of the adaptability of the genetic code using genetic algorithms. Bio-Inspired Modeling of Cognitive Tasks, Pt 1, Proceedings 4527: 478-487.
28. Floreano D (1997) Reducing human design and increasing adaptability in Evolutionary Robotics. Evolutionary Robotics-From Intelligent Robots to Artificial Life.
29. Mosincat A, Binder W, Jazayeri M (2014) Achieving runtime adaptability through automated model evolution and variant selection. Enterprise Information Systems 8: 67-83.
30. Gahagan SM, Herrmann JW (2001) Improving simulation model adaptability with a production control framework. Wsc'01: Proceedings of the 2001 Winter Simulation Conference, Vols 1 and 2: 937-945.
31. Hu XL (2006) Context-dependent adaptability in crowd behavior simulation. IRI 2006: Proceedings of the 2006 IEEE International Conference on Information Reuse and Integration: 214-219.
32. Mayhew PJ (2006) Discovering Evolutionary Ecology: Bringing Together Ecology and Evolution. Oxford University Press, United Kingdom & Europe,.
33. Harms MJ TJ (2014) Historical contingency and its biophysical basis in glucocorticoid receptor evolution. Nature 512: 203-207.
34. Traulsen A, Nowak MA (2006) Evolution of cooperation by multilevel selection. Proceedings of the National Academy of Sciences of the United States of America 103: 10952-10955.
35. Yip KY, Patel P, Kim PM, Engelman DM, McDermott D, et al. (2008) An integrated system for studying residue coevolution in proteins. Bioinformatics 24: 290-292.
36. Zhang F, Hui C, Pauw A (2013) Adaptive Divergence in Darwin's Race: How Coevolution Can Generate Trait Diversity in a Pollination System. Evolution 67: 548-560.
37. Shoval O, Sheftel H, Shinar G, Hart Y, Ramote O, et al. (2012) Evolutionary Trade-Offs, Pareto Optimality, and the Geometry of Phenotype Space. Science 336: 1157-1160.
38. Duckworth RA (2009) The role of behavior in evolution: a search for mechanism. Evolutionary Ecology 23: 513-531.
39. Dobzhansky T (2013) Nothing in Biology Makes Sense Except in the Light of Evolution. American Biology Teacher 75: 87-91.
40. Hogeweg P (2011) The Roots of Bioinformatics in Theoretical Biology. Plos Computational Biology 7.
41. Hoban S, Bertorelle G, Gaggiotti OE (2012) Computer simulations: tools for population and evolutionary genetics. Nature Reviews Genetics 13: 110-122.
42. Carvajal-Rodriguez A (2008) Simulation of genomes: A review. Current Genomics 9: 155-159.
43. Hudson RR (2002) Generating samples under a Wright-Fisher neutral model of genetic variation. Bioinformatics 18: 337-338.
44. Liang LM, Zollner S, Abecasis GR (2007) GENOME: a rapid coalescent-based whole genome simulator. Bioinformatics 23: 1565-1567.
45. Peng B, Kimmel M (2005) simuPOP: a forward-time population genetics simulation environment. Bioinformatics 21: 3686-3687.
46. Balloux F (2001) EASYPOP (version 1.7): a computer program for population genetics simulations. J Hered 92: 301-302.
47. De Jong H (2002) Modeling and simulation of genetic regulatory systems: A literature review. Journal of Computational Biology 9: 67-103.
48. Arkin A, Ross J, McAdams HH (1998) Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected Escherichia coli cells. Genetics 149: 1633-1648.
49. Friedman N, Linial M, Nachman I, Pe'er D (2000) Using Bayesian networks to analyze expression data. Journal of Computational Biology 7: 601-620.
50. A W (1998) Genomic regulation modeled as a network with basins of attraction. Pac Symp Biocomput: 89-102.

51. Thomas R, Thieffry D, Kaufman M (1995) Dynamical Behavior of Biological Regulatory Networks .1. Biological Role of Feedback Loops and Practical Use of the Concept of the Loop-Characteristic State. Bulletin of Mathematical Biology 57: 247-276.
52. Smolen P, Baxter DA, Byrne JH (2000) Modeling transcriptional control in gene networks - Methods, recent results, and future directions. Bulletin of Mathematical Biology 62: 247-292.
53. Meyers S, Friedland P (1984) Knowledge-Based Simulation of Genetic-Regulation in Bacteriophage-Lambda. Nucleic Acids Research 12: 1-9.
54. Tang Y, Valocchi AJ (2013) An improved cellular automaton method to model multispecies biofilms. Water Res 47: 5729-5742.
55. Nuno Gracias HP JAL, Agostinho Rosa (1997) Gaia:An artificial Life Environment for ecological systems simulation. Artificial Life: 124-134.
56. Wilke CO, Wang JL, Ofria C, Lenski RE, Adami C (2001) Evolution of digital organisms at high mutation rates leads to survival of the flattest. Nature 412: 331-333.
57. Lenski RE, Ofria C, Pennock RT, Adami C (2003) The evolutionary origin of complex features. Nature 423: 139-144.
58. Dada JO, Mendes P (2011) Multi-scale modelling and simulation in systems biology. Integr Biol (Camb) 3: 86-96.
59. Fogel LJ (1990) The Future of Evolutionary Programming. Twenty-Fourth Asilomar Conference on Signals, Systems & Computers, Vols 1 and 2: 1036-1038.
60. Booker LB, Goldberg DE, Holland JH (1989) Classifier Systems and Genetic Algorithms. Artificial Intelligence 40: 235-282.
61. Back T (1996) Evolution strategies: An alternative evolutionary algorithm. Artificial Evolution 1063: 3-20.
62. Ablay P (1991) 10 Theses Regarding the Design of Controlled Evolutionary Strategies. Lecture Notes in Artificial Intelligence 565: 457-481.
63. Barricelli NA (1963) Intelligence Mechanisms Behind Biological Evolution. Scientia 98: 176-&.
64. Fraser AS (1958) Monte-Carlo Analyses of Genetic Models. Nature 181: 208-209.
65. Bonizzoni P, Cooper SB, Lowe B, Sorbi A (2009) Computational Paradigms from Nature Foreword. Theoretical Computer Science 410: 283-284.
66. Paun G (2005) Bio-inspired computing paradigms (natural computing). Unconventional Programming Paradigms 3566: 155-160.
67. Schlachter F, Meister E, Kernbach S, Levi P (2008) Evolve-ability of the robot platform in the Symbrion project. Sasow 2008: Second Ieee International Conference on Self-Adaptive and Self-Organizing Systems Workshops, Proceedings: 144-149.
68. Stauffer A, Mange D, Tempesti G (2006) Bio-inspired computing machines with self-repair mechanisms. Biologically Inspired Approaches to Advanced Information Technology, Proceedings 3853: 128-140.
69. Camazine S, Deneubourg, J.L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E. (2003) Self-Organization in Biological Systems Princeton University Press.
70. Dover GA (1993) The Origins of Order - Self-Organization and Selection in Evolution - Kauffman,Sa. Nature 365: 704-706.
71. Durr P, Mattiussi C, Floreano D (2006) Neuroevolution with Analog Genetic Encoding. Parallel Problem Solving from Nature - Ppsn Ix, Proceedings 4193: 671-680.
72. Stanley KO, Miikkulainen R (2002) Evolving neural networks through augmenting topologies. Evolutionary Computation 10: 99-127.
73. Bell JE, Griffis SE (2010) Swarm Intelligence: Application of the Ant Colony Optimization Algorithm to Logistics-Oriented Vehicle Routing Problems. Journal of Business Logistics 31: 157-175.
74. al-Rifaie MM, Bishop JM, Caines S (2012) Creativity and Autonomy in Swarm Intelligence Systems. Cognitive Computation 4: 320-331.
75. Stanley KO, Miikkulainen R (2003) A taxonomy for artificial embryogeny. Artificial Life 9: 93-130.
76. Hurst LD (1995) Reinventing Darwin - the Great Evolutionary Debate - Eldredge,N. Nature 377: 693-694.

77. Hodgkin J (1998) Seven types of pleiotropy. Int J Dev Biol 42: 501-505.
78. Pagel M (2014) Arrival of the Fittest: Solving Evolution's Greatest Puzzle. Nature 514: 34-34.
79. Bak P, Tang C, Wiesenfeld K (1987) Self-organized criticality: An explanation of the 1/f noise. Phys Rev Lett 59: 381-384.
80. Smith P (1991) The Butterfly Effect. Proceedings of the Aristotelian Society, New Series, Vol 91 91: 247-267.
81. Froyland J, Alfsen KH (1984) Lyapunov-Exponent Spectra for the Lorenz Model. Physical Review A 29: 2928-2931.
82. Llibre J, Messias M, Da Silva PR (2010) Global Dynamics of the Lorenz System with Invariant Algebraic Surfaces. International Journal of Bifurcation and Chaos 20: 3137-3155.
83. Gellmann M (1992) Complexity and Complex Adaptive Systems. Evolution of Human Languages 11: 3-18.
84. Holland JH (1992) Complex Adaptive Systems. Daedalus 121: 17-30.
85. Kozlov YV, Gladskikh AI, Baraboshkin VV (1979) Gas-Analysis Complex as an Adaptive System. Measurement Techniques 22: 335-337.
86. Zhao L, Yang G, Wang W, Chen Y, Huang JP, et al. (2011) Herd behavior in a complex adaptive system. Proceedings of the National Academy of Sciences of the United States of America 108: 15058-15063.
87. Schaaff KP, Bossio FT (1996) Warfare as a complex adaptive system. Milcom 96, Conference Proceedings, Vols 1-3: 299-302.
88. Lenton TM, van Oijen M (2002) Gaia as a complex adaptive system. Philosophical Transactions of the Royal Society of London Series B-Biological Sciences 357: 683-695.
89. Noor E, Milo R (2012) Efficiency in Evolutionary Trade-Offs. Science 336: 1114-1115.
90. Pigliucci M MG (2010) Evolution - the Extended Synthesis. The MIT Press.
91. Mcshea DW (1991) Complexity and Evolution - What Everybody Knows. Biology & Philosophy 6: 303-324.
92. Furusawa C, Kaneko K (2000) Origin of complexity in multicellular organisms. Physical Review Letters 84: 6130-6133.
93. Adami C, Ofria C, Collier TC (2000) Evolution of biological complexity. Proceedings of the National Academy of Sciences of the United States of America 97: 4463-4468.
94. Adami C (2002) What is complexity? Bioessays 24: 1085-1094.
95. Werner E (2007) Life: An introduction to complex systems biology. Nature 446: 493-494.
96. Grimm V, Revilla E, Berger U, Jeltsch F, Mooij WM, et al. (2005) Pattern-oriented modeling of agent-based complex systems: Lessons from ecology. Science 310: 987-991.
97. Ofria C, Wilke CO (2004) Avida: A software platform for research in computational evolutionary biology. Artificial Life 10: 191-229.
98. Pauly D, Christensen V, Walters C (2000) Ecopath, Ecosim, and Ecospace as tools for evaluating ecosystem impact of fisheries. Ices Journal of Marine Science 57: 697-706.
99. Favetto A, Osella A (1999) Numerical simulation of currents induced by geomagnetic storms on buried pipelines: An application to the Tierra del Fuego, Argentina, gas transmission route. Ieee Transactions on Geoscience and Remote Sensing 37: 614-619.
100. Evert Haasdijk NB, Stefano Nolfi, A. E. Eiben (2014) Evolutionary robotics. Evolutionary Intelligence 7: 69-70.
101. Vito Trianni SN, Marco Dorigo (2008) Evolution, Self-organization and Swarm Robotics. Swarm Intelligence Natural Computing Series. pp. 163-191.
102. Varenne F, Chaigneau, P., Petitot, J., Doursat, R. (2015) Programming the Emergence in Morphogenetically Architected Complex Systems. . Acta Biotheoretica: 1-14.
103. Floreano D, Keller L (2010) Evolution of Adaptive Behaviour in Robots by Means of Darwinian Selection. Plos Biology 8.
104. Hooker CA (1995) Adaptation in Natural and Artificial Systems - Holland,Jh. Philosophical Psychology 8: 287-299.

105. Oksanen L, Fretwell SD, Arruda J, Niemela P (1981) Exploitation Ecosystems in Gradients of Primary Productivity. American Naturalist 118: 240-261.
106. Mertz DB, Wade MJ (1976) Prudent Prey and Prudent Predator. American Naturalist 110: 489-496.
107. Bijma P, Muir WA, Van Arendonk JAM (2007) Multilevel selection 1: Quantitative genetics of inheritance and response to selection. Genetics 175: 277-288.
108. Whitham TG, Lonsdorf E, Schweitzer JA, Bailey JK, Fischer DG, et al. (2005) "All effects of a gene on the world": Extended phenotypes, feedbacks, and multi-level selection. Ecoscience 12: 5-7.
109. Wilson DS, Van Vugt M, O'Gorman R (2008) Multilevel selection theory and major evolutionary transitions - Implications for psychological science. Current Directions in Psychological Science 17: 6-9.
110. Moore AJ, Brodie ED, Wolf JB (1997) Interacting phenotypes and the evolutionary process .1. Direct and indirect genetic effects of social interactions. Evolution 51: 1352-1362.
111. McGlothlin JW, Moore AJ, Wolf JB, Brodie ED (2010) Interacting Phenotypes and the Evolutionary Process. Iii. Social Evolution. Evolution 64: 2558-2574.
112. Woese CR (2004) A new biology for a new century. Microbiology and Molecular Biology Reviews 68: 173-+.
113. Deroin P (2012) Evolution A new butterfly effect. Biofutur: 19-19.
114. Orrell D, Smith L, Barkmeijer J, Palmer TN (2001) Model error in weather forecasting. Nonlinear Processes in Geophysics 8: 357-371.
115. Rand DA, Wilson HB (1993) Evolutionary Catastrophes, Punctuated Equilibria and Gradualism in Ecosystem Evolution. Proceedings of the Royal Society B-Biological Sciences 253: 137-141.
116. Buchanan A, Triant M, Bedau MA (2004) The flexible balance of evolutionary novelty and memory in the face of environmental catastrophes. Artificial Life Ix: 297-302.
117. Yao Y, Marchal K, Van de Peer Y (2014) Improving the Adaptability of Simulated Evolutionary Swarm Robots in Dynamically Changing Environments. Plos One 9.
118. Young ID (2005) Medical genetics. Oxford University Press, United Kingdom & Europe,.
119. ennifer Hallinan JW (2004) Evolving Genetic Regulatory Networks Using an Artificial Genome. SECOND ASIA-PACIFIC BIOINFORMATICS CONFERENCE (APBC2004) 29.
120. Vega J, Perdices E, Canas JM (2013) Robot evolutionary localization based on attentive visual short-term memory. Sensors (Basel) 13: 1268-1299.
121. Reil T (1999) Dynamics of gene expression in an artificial genome - Implications for biological and artificial ontogeny. Advances in Artificial Life, Proceedings 1674: 457-466.
122. Covert AW, 3rd, Lenski RE, Wilke CO, Ofria C (2013) Experiments on the role of deleterious mutations as stepping stones in adaptive evolution. Proc Natl Acad Sci U S A 110: E3171-3178.
123. Clune J, Mouret JB, Lipson H (2013) The evolutionary origins of modularity. Proc Biol Sci 280: 20122863.
124. Helbing D, & Balietti, S. (2013) How to do agent-based simulations in the future: From modeling social mechanisms to emergent phenomena and interactive systems design. . In: Helbing D, editor. Social Self-Organization. Berlin: Springer. pp. 25-70.
125. Abdelmotaleb A SM, Davey N, Steuber V, Wróbel B (2013) From evolving artificial gene regulatory networks to evolving spiking neural networks for pattern recognition. BMC Neuroscience 14(1): 423.
126. Kassahun Y, Edgington M, Metzen JH, Sommer G, Kirchner F (2007) A Common Genetic Encoding for Both Direct and Indirect Encodings of Networks. Gecco 2007: Genetic and Evolutionary Computation Conference, Vol 1 and 2: 1029-1036.
127. Gauci J, Stanley K (2007) Generating Large-Scale Neural Networks Through Discovering Geometric Regularities. Gecco 2007: Genetic and Evolutionary Computation Conference, Vol 1 and 2: 997-1004.

128. Mendao M, Timmis J, Andrews PS, Davies M (2007) The immune system in pieces: Computational lessons from degeneracy in the immune system. 2007 IEEE Symposium on Foundations of Computational Intelligence, Vols 1 and 2: 394-400.
129. Pugh JK, Stanley KO (2013) Evolving Multimodal Controllers with HyperNEAT. Gecco'13: Proceedings of the 2013 Genetic and Evolutionary Computation Conference: 735-742.
130. Jin YC, Guo HL, Meng Y (2012) A Hierarchical Gene Regulatory Network for Adaptive Multirobot Pattern Formation. Ieee Transactions on Systems Man and Cybernetics Part B-Cybernetics 42: 805-816.
131. Zeeman EC (1976) Catastrophe Theory. Scientific American 234: 65-&.
132. Esquinas J, Lopezgomez J (1987) Optimal Results in Local Bifurcation-Theory. Bulletin of the Australian Mathematical Society 36: 25-37.
133. Mitri S, Xavier JB, Foster KR (2011) Social evolution in multispecies biofilms. Proceedings of the National Academy of Sciences of the United States of America 108: 10839-10846.
134. M ´onica Garc´ıa EA (2014) Multi-scale modeling of dynamic systems for evolution. Frontiers in Ecology, Evolution and Complexity.
135. West SA, Griffin AS, Gardner A (2007) Social semantics: altruism, cooperation, mutualism, strong reciprocity and group selection. Journal of Evolutionary Biology 20: 415-432.
136. Tinker MT, Bentall G, Estes JA (2008) Food limitation leads to behavioral diversification and dietary specialization in sea otters. Proceedings of the National Academy of Sciences of the United States of America 105: 560-565.
137. Tsuda ME, Kawata M (2010) Evolution of Gene Regulatory Networks by Fluctuating Selection and Intrinsic Constraints. Plos Computational Biology 6.
138. Bennett MR, Pang WL, Ostroff NA, Baumgartner BL, Nayak S, et al. (2008) Metabolic gene regulation in a dynamically changing environment. Nature 454: 1119-1122.
139. Kashtan N, Noor E, Alon U (2007) Varying environments can speed up evolution. Proceedings of the National Academy of Sciences of the United States of America 104: 13711-13716.
140. Wischmann S, Floreano D, Keller L (2012) Historical contingency affects signaling strategies and competitive abilities in evolving populations of simulated robots. Proceedings of the National Academy of Sciences of the United States of America 109: 864-868.
141. Gorochowski TE, Matyjaszkiewicz A, Todd T, Oak N, Kowalska K, et al. (2012) BSim: An Agent-Based Tool for Modeling Bacterial Populations in Systems and Synthetic Biology. Plos One 7.
142. Langton Ce (1989) Artificial Life Addison: Wesley.
143. Bedau MA (2003) Artificial life: organization, adaptation and complexity from the bottom up. Trends in Cognitive Sciences 7: 505-512.
144. Ruiz-Mirazo K, Pereto J, Moreno A (2004) A universal definition of life: Autonomy and open-ended evolution. Origins of Life and Evolution of Biospheres 34: 323-346.
145. Farmer JD, Belin AD (1992) Artificial Life - the Coming Evolution. Artificial Life Ii 10: 815-840.
146. Kim KJ, Cho SB (2006) A comprehensive overview of the applications of artificial life. Artif Life 12: 153-182.
147. Dugatkin LA, Bekoff M (2003) Play and the evolution of fairness: a game theory model. Behavioural Processes 60: 209-214.
148. Thomas V (2005) Evolutionary Game Theory, Natural Selection, and Darwinian Dynamics. Cambridge University Press: 72-87.
149. Huang X, Lee J, Sun TH, Peper F (2013) Self-adaptive self-reproductions in cellular automata. Physica D-Nonlinear Phenomena 263: 11-20.
150. Davies KJ, Green JEF, Bean NG, Binder BJ, Ross JV (2014) On the derivation of approximations to cellular automata models and the assumption of independence. Mathematical Biosciences 253: 63-71.
151. Golabczak M, Konstantynowicz A, Golabczak A (2014) Use of Cellular Automata for Modelling of the Carbon Nano layer Growth on a Light Alloy Substrate. Journal of Nano Research 26: 159-167.

152. Bie ZJ, Han Q, Liu C, Huang JJ, Song LP, et al. (2014) Chaotic Behavior of One-Dimensional Cellular Automata Rule 24. Discrete Dynamics in Nature and Society.
153. Svyetlichnyy DS, Mikhalyov AI (2014) Three-dimensional Frontal Cellular Automata Model of Microstructure Evolution Phase Transformation Module. Isij International 54: 1386-1395.
154. Xavier JB, Foster KR (2007) Cooperation and conflict in microbial biofilms. Proceedings of the National Academy of Sciences of the United States of America 104: 876-881.
155. Bucci V, Nadell CD, Xavier JB (2011) The Evolution of Bacteriocin Production in Bacterial Biofilms. American Naturalist 178: E162-E173.
156. T. Blackwell JB, and X. Li. (2008) Particle Swarms for Dynamic Optimization Problems. Swarm Intelligence: 193-217.
157. Yeom K (2010) Bio-inspired self-organization for supporting dynamic reconfiguration of modular agents. Mathematical and Computer Modelling 52: 2097-2117.
158. Whitacre JM, Rohlfshagen P, Bender A, Yao X (2010) The Role of Degenerate Robustness in the Evolvability of Multi-agent Systems in Dynamic Environments. Parallel Problems Solving from Nature - Ppsn Xi, Pt I 6238: 284-293.
159. Yao YB, G; Van de Peer, Y. A bio-inspired agent-based system for controlling robot behaviour; 2011; Pairs. pp. 1 - 8.
160. Gerkey BP, Vaughan RT, Howard A (2003) The Player/Stage project: Tools for multi-robot and distributed sensor systems. Proceedings of the 11th International Conference on Advanced Robotics 2003, Vol 1-3: 317-323.
161. F. Mondada MB, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz,S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli (2009) The epuck,a robot designed for education in engineering In Proc of the 9th Conference on Autonomous Robot Systems and Competitions. pp. 59-65.
162. F. Mondada EF, and P. Ienne. (1993) Mobile robot miniaturization: A tool for investigation in control algorithms. In Proc of the Third International Symposium on Simulation on Experimental Robots. pp. 501-513.
163. A. Berlanga PI, A. Sanchis, and J. M. Molina (1999) Neural networks robot controller trained with evolution strategies. In Proc of the 1999 IEEE Congress on Evolutionary Computation. pp. 413–419.
164. R. Calabretta SN, and D. Parisi (1997) Investigating the role of diploidy in simulated populations of evolving individuals Proc of the Fourth European Conference on Artificial Life. Brighton, UK: The MIT Press.
165. Calabretta R, Nolfi S, Parisi D, Wagner GP (2000) Duplication of modules facilitates the evolution of functional specialization. Artificial Life 6: 69-84.
166. Calabretta R, Nolfi S, Parisi D, Wagner GP (1998) A case study of the evolution of modularity: Towards a bridge between evolutionary biology, artificial life, neuro- and cognitive science. Artificial Life Vi: 275-284.
167. Calabretta R, Galbiati R, Nolfi S, Parisi D (1996) Two is better than one: A diploid genotype for neural networks. Neural Processing Letters 4: 149-155.
168. Niebank K (2003) Self-organization in biological systems. Zeitschrift Fur Klinische Psychologie Psychiatrie Und Psychotherapie 51: 83-85.
169. Levi PaK, S. (2010) Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution: Springer-Verlag.
170. Kornienko S, Kornienko O, Nagarathinam A, Levi P (2007) From real robot swarm to evolutionary multi-robot organism. 2007 Ieee Congress on Evolutionary Computation, Vols 1-10, Proceedings: 1483-1490.
171. Schmeck H, Muller-Schloer C, Cakar E, Mnif M, Richter U (2010) Adaptivity and Self-Organization in Organic Computing Systems. Acm Transactions on Autonomous and Adaptive Systems 5.
172. Kernbach S, Hamann H, Stradner J, Thenius R, Schmickl T, et al. (2009) On adaptive self-organization in artificial robot organisms. 2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns: 33-43.

173. Dorigo M, Trianni V, Sahin E, Gross R, Labella TH, et al. (2004) Evolving self-organizing behaviors for a swarm-bot. Autonomous Robots 17: 223-245.
174. Yim M, Duff, D.G., Roufas, K.D. PolyBot: a modular reconfigurable robot. Robotics and Automation 2000. pp. 514-520.
175. Ampatzis C, Tuci E, Trianni V, Christensen AL, Dorigo M (2009) Evolving Self-Assembly in Autonomous Homogeneous Robots: Experiments with Two Physical Robots. Artificial Life 15: 465-484.
176. Nolfi SaF, D. (2000) Evolutionary Robotics: The Biology Intelligence, and Technology of Self-Organizing Machines. Cambridge, MA: MIT Press.
177. Michael Rubenstein AC, Radhika Nagpal (2014) Programmable self-assembly in a thousand-robot swarm. Science 345: 795.
178. Canedo-Rodriguez A, Iglesias R, Regueiro CV, Alvarez-Santos V, Pardo XM (2011) Self-organized Multi-agent System for Robot Deployment in Unknown Environments. Foundations on Natural and Artificial Computation 6686: 165-174.
179. Shen WM, Chuong CM, Will P (2002) Simulating self-organization for multi-robot systems. 2002 Ieee/Rsj International Conference on Intelligent Robots and Systems, Vols 1-3, Proceedings: 2776-2781.
180. Darken R, McDowell P, Johnson E (2005) The Delta3D open source game engine. Ieee Computer Graphics and Applications 25: 10-12.
181. Krzak L, Mielniczuk R, Worek C (2008) Software-defined digital radio baseband processor using Blackfin DSP. Icses 2008 International Conference on Signals and Electronic Systems, Conference Proceedings: 523-525.
182. Ohno S (1970) Evolution by gene duplication: Springer-Verlag.
183. Stoltzfus A (2006) Mutation-biased adaptation in a protein NK model. Molecular Biology and Evolution 23: 1852-1862.
184. Weng JY, McClelland J, Pentland A, Sporns O, Stockman I, et al. (2001) Artificial intelligence - Autonomous mental development by robots and animals. Science 291: 599-600.
185. Floreano D, Urzelai J (2001) Evolution of plastic control networks. Autonomous Robots 11: 311-317.
186. SA K (1991) The sciences of complexity and ‘‘Origins of order’’. Proceedings of the Biennial Meeting of the PHILOS. pp. 299-322.
187. Linksvayer TA, Fewell JH, Gadau J, Laubichler MD (2012) Developmental evolution in social insects: regulatory networks from genes to societies. J Exp Zool B Mol Dev Evol 318: 159-169.
188. Jacob F MJ (1961) Genetic regulatory mechanisms in the synthesis of proteins. J MolBiol 3: 318–356.
189. Douglas HC, Hawthorne DC (1966) Regulation of genes controlling synthesis of the galactose pathway enzymes in yeast. Genetics 54: 911-916.
190. Yao X, Liu Y (1998) Towards designing artificial neural networks by evolution. Applied Mathematics and Computation 91: 83-90.
191. Hornby GS PJ (2002) Creating high-level components with a generative representation for body-brain evolution. Artificial Life 8.
192. Bentley PJ KS (1999) The ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999) 35–43.
193. Guo H, Meng Y, Jin Y (2009) A cellular mechanism for multi-robot construction via evolutionary multi-objective optimization of a gene regulatory network. Biosystems 98: 193-203.
194. Mattiussi C DP, Marbach D, Floreano D (2011) Beyond graphs: A new synthesis. J ComputSci 2: 165– 177.
195. Flamm C, Endler L, Muller S, Widder S, Schuster P (2007) A minimal and self-consistent in silico cell model based on macromolecular interactions. Philosophical Transactions of the Royal Society B-Biological Sciences 362: 1831-1839.
196. Rosenberg SM (2001) Evolving responsively: adaptive mutation. Nat Rev Genet 2: 504-515.

197. Polvichai J BU (2011) The survival robots: An artificial life. Computer Science and Software Engineering (JCSSE), 2011 Eighth International Joint Conference: 166 - 169.
198. James HA SC, Hawick KA (2004) A framework and simulation engine for studying artificial life. Res LettInf MathSci 6: 143-155.
199. Grefenstette JJ DR (1996) Methods for competitive and cooperative co-evolution. In Adaptation, Co-evolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Symposium: 45-50.
200. Tanner CJ, Jackson AL (2012) Social structure emerges via the interaction between local ecology and individual behaviour. Journal of Animal Ecology 81: 260-267.
201. Bredeche N, Haasdijk E, Eiben AE (2010) On-Line, On-Board Evolution of Robot Controllers. Artificial Evolution 5975: 110-121.
202. Tan BSA-H (2009) A self-organizing neural network architecture for intentional planning agents. AAMAS '09 Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems 2: 1081-1088.
203. Yu JB, Wang SJ, Xi LF (2008) Evolving artificial neural networks using an improved PSO and DPSO. Neurocomputing 71: 1054-1060.
204. M C (1995) The Price of Programmability Computerkultur: 261-281.
205. Mitchell A, Romano GH, Groisman B, Yona A, Dekel E, et al. (2009) Adaptive prediction of environmental changes by microorganisms. Nature 460: 220-U280.
206. GP Wagner LA (1996) Perspective: Complex adaptations and the evolution of evolvability

Evolution 967-976.
207. Lynch M, Abegg A (2010) The Rate of Establishment of Complex Adaptations. Molecular Biology and Evolution 27: 1404-1414.
208. J B (2002) Evolving Modular Genetic Regulatory Networks. Evolutionary Computation, CEC '02 Proceedings 2: 1872 – 1877.
209. Draghi JA, Plotkin JB (2011) MOLECULAR EVOLUTION Hidden diversity sparks adaptation. Nature 474: 45-46.
210. Kurakin A (2009) Scale-free flow of life: on the biology, economics, and physics of the cell. Theor Biol Med Model 6: 6.
211. Tononi G, Sporns O, Edelman GM (1999) Measures of degeneracy and redundancy in biological networks. Proceedings of the National Academy of Sciences of the United States of America 96: 3257-3262.
212. Kizer JB (1978) Evolution of Complex Adaptations. Speculations in Science and Technology 1: 513-514.
213. Applin S, Fischer M (2013) Asynchronous Adaptations to Complex Social Interactions. Ieee Technology and Society Magazine 32: 35-44.
214. Gould F (1988) Evolutionary Biology and Genetically Engineered Crops. Bioscience 38: 26-33.
215. Williamson M (1992) Environmental risks from the release of genetically modified organisms (GMOs)-the need for molecular ecology. Molecular Ecology 1: 3-8.
216. Prakash CS (2001) The genetically modified crop debate in the context of agriculture evolution. Plant Physiology 126: 8-15.
217. Wagner A (2008) Gene duplications, robustness and evolutionary innovations. Bioessays 30: 367-373.
218. Edwards AWF (2000) The Genetical Theory of Natural Selection. Genetics 154: 1419-1426.
219. Orr HA (2000) Adaptation and the cost of complexity. Evolution 54: 13-20.
220. Gunter P. Wagner JPK-H, Mihaela Pavlicev, Joel R. Peck, David Waxman, James M. Cheverud (2008) Pleiotropic scaling of gene effects and the 'cost of complexity'. Nature 452: 470-473.
221. Moore MJ (1995) The Cost of Complexity. Proceedings of the Institution of Mechanical Engineers Part a-Journal of Power and Energy 209: 251-251.

222. Ulieru M, & Doursat, R. (2010) Emergent engineering: a radical paradigm shift. International Journal of Autonomous and Adaptive Communications Systems 4: 39-60.
223. Conforth M, Meng Y (2008) Reinforcement Learning for Neural Networks using Swarm Intelligence. 2008 Ieee Swarm Intelligence Symposium: 89-95.
224. Kamiya A, Kimura H, Yamamura M, Kobayashi S (1998) Power plant start-up scheduling: a reinforcement learning approach combined with evolutionary computation. Journal of Intelligent & Fuzzy Systems 6: 99-115.
225. Sigmund K, Young HP (1995) Evolutionary Game-Theory in Biology and Economics - Introduction. Games and Economic Behavior 11: 103-110.
226. Mangel M (1998) Game theory and animal behaviour. Nature 395: 32-32.
227. Lynch KE, Kemp DJ (2014) Nature-via-nurture and unravelling causality in evolutionary genetics. Trends in Ecology & Evolution 29: 2-4.
228. Amit Gupta SA (2012) Insights from Complexity Theory: Understanding Organisations better. IIM Bangalore.
229. Di Bernardo M (2010) Natural Selection and Self-organization in Complex Adaptive Systems. Rivista Di Biologia-Biology Forum 103: 89-110.
230. Levin SA (1998) Ecosystems and the biosphere as complex adaptive systems. Ecosystems 1: 431-436.
231. Margulis L, Bermudes D, Obar R, Tzertzinis G (1986) Symbiosis in Evolution - Status of the Hypothesis of the Spirochete Origin of Undulipodia. Origins of Life and Evolution of the Biosphere 16: 319-319.
232. Duffin M (2006) Symbiosis and evolution. Natural History 115: 10-10.
233. Watson RA, Pollack JB (1999) How symbiosis can guide evolution. Advances in Artificial Life, Proceedings 1674: 29-38.
234. Numaoka C (1995) Symbiosis and co-evolution in animats. Advances in Artificial Life 929: 261-272.
235. Webster NS (2014) Cooperation, communication, and co-evolution: grand challenges in microbial symbiosis research. Frontiers in Microbiology 5.
236. Watson RA, and Jordan B. Pollack (2001) Symbiotic composition and evolvability Advances in Artificial Life: Springer Berlin Heidelberg. pp. 480-490.
237. Rosenfield I, Ziff E, Kirschner MW (2006) The plausibility of life: Resolving Darwin's dilemma. New York Review of Books 53: 12-+.
238. Gerhart J, Kirschner M (2007) The theory of facilitated variation. Proceedings of the National Academy of Sciences of the United States of America 104: 8582-8589.
239. Axelrod DI, Bailey HP (1968) Cretaceous Dinosaur Extinction. Evolution 22: 595-&.
240. Brasier MD (1992) Background to the Cambrian Explosion. Journal of the Geological Society 149: 585-587.
241. Castrodeza C (1979) Non-Progressive Evolution, the Red Queen Hypothesis, and the Balance of Nature. Acta Biotheoretica 28: 11-18.
242. Cohen IR, Harel D (2007) Explaining a complex living system: dynamics, multi-scaling and emergence. Journal of the Royal Society Interface 4: 175-182.
243. Al-Tawil K, Moritz CA (2001) Performance modeling and evaluation of MPI. Journal of Parallel and Distributed Computing 61: 202-223.