2021

On the relation between robustness, evolvability and phenotypic complexity: Insight from Artificial Evolutionary Experiments

Milano, Nicola

UNIVERSITY OF
PLYMOUTH

On the relation between robustness, evolvability and phenotypic complexity: Insight from Artificial Evolutionary Experiments

by

Nicola Milano

A thesis submitted to the University of Plymouth
in partial fulfilment for the degree of

DOCTOR OF PHILOSOPY

School of Computing, Electronics and Mathematics

February 2021

On the relation between robustness, evolvability and phenotypic complexity: Insight from Artificial Evolutionary Experiments by Nicola Milano

Abstract

This thesis addresses the study of evolutionary methods to achieve robustness and evolvability in artificial systems. Chapter 1 introduces the research area, reviews the state of the art, discusses promising research directions, and presents the two major scientific objectives of the thesis. The first objective, which is covered in Chapter 2, is to verify and exploit the impact of the environmental variations on the evolvability of an artificial system. This is accomplished through the use of two experimental setup: the simulation of digital circuits and the simulation of a robotic agent situated in an external environment. Digital circuits are used to considers the variation as internal to the system, modelled as fault in circuit gates; agent-based simulation instead consider the variation in the external environment where the robot performs. The second objective, which is targeted in Chapter 3, presents the design of a new algorithm and a more efficient selection mechanism that exploits the characteristics of robustness and neutrality of the digital circuit domain. Due to its relative simplicity quantitative measures of phenotypic complexity, robustness and evolvability are obtained. Such information on the search space composition is then used to design a novel evolutionary algorithm that outperforms previously methods and to propose a selection mechanism that takes into account the phenotypic complexity of the genotypes. Chapter 4 summarizes the results obtained and describes the major contributions of the thesis.

Author's declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award without prior agreement of the Doctoral College Quality Sub-Committee. Work submitted for this research degree at the University of Plymouth has not formed part of any other degree either at the University of Plymouth or at another establishment.
Relevant scientific seminars and conferences were regularly attended at which this work was presented. Four articles have been accepted for publication in refereed journals and two articles have been presented in international conferences and then published in the proceedings.
This study was carried out in collaboration with the Istituto di Scienze e Tecnologie della Cognizione (ISTC) - Consiglio Nazionale delle Ricerche (CNR), Rome.
This thesis contains works being the result of collaborations with other researchers. The author contribution to the reported works over the total was about 80% for the work described in chapter 4 and 80% for the work described in chapter 5.

Word count for the main body of this thesis: **32220**

Publications:

Milano, Nicola, and Stefano Nolfi. "Robustness to faults promotes evolvability: Insights from evolving digital circuits." *PloS one* 11.7 (2016).

Milano, Nicola, Paolo Pagliuca, and Stefano Nolfi. "Robustness, evolvability and phenotypic complexity: insights from evolving digital circuits." *Evolutionary Intelligence* 12.1 (2019): 83-95.

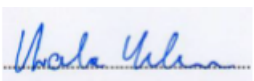
Milano, Nicola, Jônata Tyska Carvalho, and Stefano Nolfi. "Moderate environmental variation across generations promotes the evolution of robust solutions." *Artificial life* 24.4 (2019): 277-295.

Pagliuca, Paolo, Nicola Milano, and Stefano Nolfi. "Maximizing adaptive power in neuroevolution." *PloS one* 13.7 (2018).

Presentations at conferences:

Milano, Nicola, Jônata Tyska Carvalho, and Stefano Nolfi. "Environmental variations promotes adaptation in artificial evolution." *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2017.

Carvalho, Jônata Tyska, Nicola Milano, and Stefano Nolfi. "Evolving Robust Solutions for Stochastically Varying Problems." *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018.

Sign:  date: 02/02/2021

Contents

List of Figures

List of tables

Chapter 1. Introduction

Robustness and evolvability are fundamental properties of biological systems. Robustness is the invariance of phenotypes in the face of perturbation; these perturbations can be internal to the system, being originated from genetic mutations, or external, results of environmental changes (de Visser et al. 2003). Evolvability is the propensity of a system to produce adaptive heritable phenotypic variations as a result of mutations.

Both robustness and evolvability are quantitative properties. For instance, a system can be more or less robust in response to a given number of mutations or to a certain amount of environmental variations. Both definitions apply to different levels of organization, including RNA and protein molecules, small genetic circuits, genome networks and even whole organisms. Focusing on the appropriate level of organization there are different type of variations and system's response, i.e. amino acids can change in protein or variations in the temperature of organism's ecosystem. Genetic mutations act on the genome of individuals and are inherited directly by new generations; instead, mutations of the environment act on a different level: environmental factors that initiate evolutionary novelties can be either new building blocks or cues that influence regulation in new ways (West-Eberhard 2003, de Visser et. al. 2003). They give rise to a process of genetic accommodation that acts on the whole population instead of the single individual. Environmentally induced novelties may have greater evolutionary potential than mutational induced ones. They can be immediately recurrent in a population; are more likely than mutational novelties to correlate with particular environmental conditions and be subjected to directional selection; and, being relatively immune to selection, are more likely to persist even though initially disadvantageous. (West-Eberhard 2003)

Biological systems that are able to cope with the effect of variations are naturally selected by evolution since they are capable of maintaining their adaptive capabilities despite the mutations received and/or since they are capable of operating effectively in varied environmental conditions. Again, developing very robust systems for a certain environment or for a recurrent genetic mutation, can prevent the evolution of novel property or functions of a system, limiting the possibility of a system to evolve and to adapt when a change occur. Biological systems also display evolvability, i.e. the capacity to improve and/or to adapt to variations, this is the case of genetic mutations that discovers new functionalities or environmental variations that permits the expression of a trait previously hidden.

To exemplify the nature of robustness and evolvability let's compare biological organisms with human designed computer programs (Wagner and Altenberg, 1996). Both are encoded in strings of characters: DNA in organisms, binary code in programs. However, they widely differ in terms of robustness and evolvability. Indeed, the former tends to preserve their function and to adapt as a result of random mutations of their sequence. The latter instead, tend to completely lose their functionalities and have literally zero probability to improve as a result of random mutations. More generically, human-design systems, if not designed specifically to be robust like fault tolerant software i.e., lack the robustness and evolvability properties of natural systems. Natural systems are not designed to be robust or evolvable, but they are emerging properties of the evolutionary process.

Aim of this thesis is to use evolutionary algorithms and artificial evolution for the analysis and the synthesis of evolvability and robustness in artificial systems. To this purpose I will use two experimental scenarios: the evolution of digital Boolean circuits selected for the ability to perform a logic function and the evolution of neuro-controlled agents situated in an external environment for the ability to solve a given task. The use of the former scenario is motivated by the fact that it constitutes a classic setup for studying the effect of neutrality and evolvability, and by the fact that it supports a simple quantitative measure of complexity (i.e. the number of connected gates forming the circuit). The use of the second scenario is motivated by its relative simplicity, its wide spread usage, and by the fact that it permits to study the effect of variations affecting the external environment or the relation between the agent and the environment.

The following chapter is dedicated to an extensive introduction about robustness and evolvability and to a literature's review on them, both from a biological and artificial point of view. Then artificial evolution, evolutionary algorithms and the principal experimental methods used in this thesis are introduced in Chapter 3. The following chapters, 4 and 5, represents the experimental part of the thesis, devoted respectively: to verify whether the need to operate in varying environment (i.e. the need to be robust to environmental variation) promotes evolvability; and to verify the correlation between complexity and evolvability and to identify the mechanisms that can promotes evolvability by favoring complexity.
Finally chapter 6 and 7 are the general conclusion and future works.

Chapter 2. On the relation between robustness and evolvability

At a first sight, robustness and evolvability have an antagonistic relationship. From a genetic point of view, the higher the robustness of a system, the lower is the probability that it will vary as a result of genetic mutations, and consequently the lower the evolvability of the system is. Indeed, mechanisms that prevent changes such as proofreading and DNA repair enhance robustness but reduce phenotypic variability (Lenski, Barrick and Ofria, 2006; Masel and Trotter, 2010). On the other hand, robustness to mutations facilitates the retention of mutations that permits the population to spread over large regions of the genotype space, the space of all possible genotype expressible by a system (Wagner, 2008). This combined with the fact that the phenotypes located on distant regions of the genetic space are much more varied than the phenotypes located in nearby regions of the genetic space, increases the differentiation of the phenotypes that can be produced, at the level of the population, through genetic variations (Wagner, 2008).

A second reason that suggests an antagonistic relationship between robustness and evolvability is that environmental factors can favor the expression of some hidden traits that remain unexpressed in static environments, so individuals can improve their functionalities and innovate under the drive of environmental variations. Prolonged exposure to static environments can cause the expression of genes that are suitable only in that particular environment and the consequent loss of other traits, developing robust individuals in that particular environment but not evolvable or adaptable (West-Eberhard, 2003, de Visser et al., 2003).

The influence of genetic robustness on evolvability also depends on whether robustness is achieved through the development of parsimonious (phenotypically simple) solutions that minimize the number of genes having a functional role (de Visser et. al. 2003) or through mechanisms capable of buffering the effect of mutations. Buffer means to be able to cope with mutations not being affect by them also if that occurs. Mechanisms able to buffer the effect of mutations are redundancy: the presence of multiple components playing the same function (i.e. different genes having the same functions); and degeneracy: i.e. the interaction between multiple components playing multiple functions (Tononi, Sporns, and Edelman, 1999; Edelman and Gally, 2001). The achievement of robustness through these mechanisms requires less parsimonious (phenotypically more complex) solutions; the complexity of these solutions plays a fundamental role for the evolvability, because

the accumulation of genetic diversity in functional genes without altering the phenotype is essential to allow exploration of the genotype space and find innovations.

We can use genotype networks to illustrate the relation between robustness and evolvability (Wagner 2008). Networks are graphs, i.e. mathematical objects that consist of nodes and of edges that link these nodes. A graph is connected if one can reach any node from any other node by traversing a path of edges, and disconnected otherwise. The genotype space can be viewed as a graph whose nodes are genotypes. Edges connect nearest (1-mutant) neighbors in this space. Different genotypes can give rise to the same phenotype, because a lot of genes are the so called "junk genes", they not have a functional role, so don't contributes to the phenotypic expression (Wagner 2008). Nodes indicated with the same symbol (e.g. circles) correspond to identical phenotypes. Figure 1.1 shows and exemple of a genotype network, nodes indicated with different symbols, instead, correspond to genotype that give rise to functionally different phenotypes. In Figure 1 The shape and the shadowing of the nodes indicate the type of the phenotype. The subnetwork including the elements with the same shape represent a phenotype network. For example, the network including all circular elements represent a phenotype network. The figure illustrates that many different novel phenotypes can be accessed from a connected genotype network that spreads far through genotype space

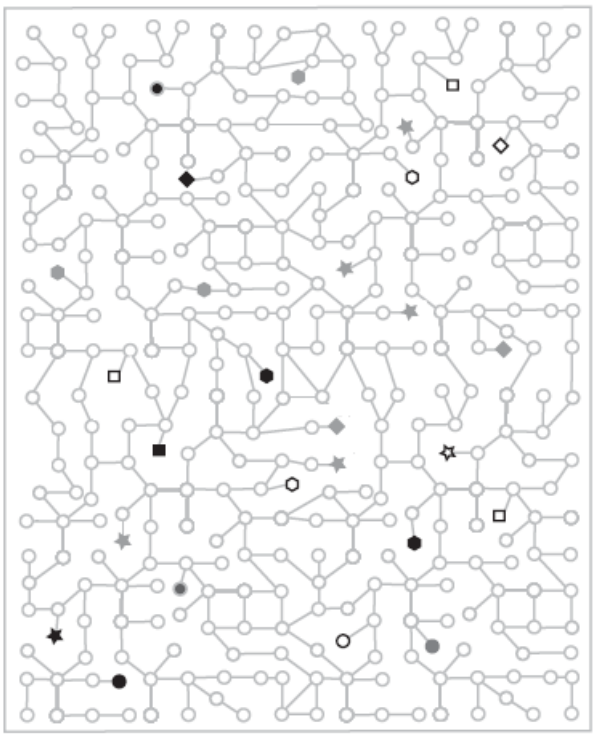


Figure 2.1. A schematic representation of a genotype network. The shape and the shadowing of the nodes indicate the type of the phenotype. The subnetwork including the elements with the same shape represent a phenotype network. For example, the network including all circular elements represent a phenotype network. The figure illustrates that many different novel phenotypes can be accessed from a connected genotype network that spreads far through genotype space. Adapted from Wagner 2008

A biological system's robustness is the ability to preserve phenotype in the face of change. This property is fundamental to spread over a genotype space that is connected as in Figure 2.1. The genotypes evolving on it may give rise to different novel phenotypes (the stars, squares and filled circles present in Figure 2.1), and in the same time preserve their own phenotype for a long time until reaching innovations. Thus, a genotype with many neutral neighbors is to some extent robust to genetic change but on the other hand leads to potential new fronts of phenotypic innovation(Wagner 2008, Raman and Wagner 2011, Hu et al. 2012). As claimed by (Wagner, 2008), the apparent conflict between robustness and evolvability can be resolved by considering that the large majority of the neighbors of a genotype belong to the same genotype network and by considering that phenotype networks often span large portions of the entire genotype space. When a genotype as many neutral neighbors the large connectedness of the phenotype networks ensure the possibility to reach distant areas of the genotype space preserving the phenotype and its functionalities, permitting the access to different sub-network confining with the initial ones.
Such characteristics are shared by many biological systems such as molecules, regulatory circuits and metabolic networks(Wagner 2008, Tononi, Sporns, and Edelman, 1999). First, their phenotypes are typically organized into vast phenotype networks that traverse a large fraction of genotype space. Second, different neighborhoods on these networks contain very different novel phenotypes.

To quantify this consider a single mutation acting on a genotype G (an empy grey circle in figure 2.1), so all the genotype accessible in its neighbor are 1-mutation far away from G. What is particularly important here is the neutral neightbornood, i.e. the fraction of neighbors that share the same phenotype of G (empty gray circles connected by lines Figure 2.1), call this fraction v . It can be viewed as a measure of mutational robustness: if $v = 0$ there is no neighbor with the same phenotype and robustness is minimal; if $v = 1$ all neighbors have the same phenotype and robustness is maximal. So, as stated before, some degree of robustness $v > 0$ is sufficient for the presence of a genotype network, and the existence of these network are essential for evolvability, because they allow access to great quantity of phenotypic variation, i.e. squares filled circles and stars in Figure 2.1, preserving existing phenotypes; Robustness is so necessary to ensure that an evolving system could explore different phenotypes, that in turn is the necessary condition for evolvability. As I

already say at a first sight robustness and evolvabilty are anticorrelated. To illustrate this point, consider the extreme case of a minimally robust genotype *G* (*v*=0). It certainly contains more novel phenotypes than the neighborhood of a genotype with greater robustness (v>0). The minimally robust genotype with 1-mutation can have access to a great number of novel phenotypes immediately, because all its neighbor have different phenotype by definition; so the lower a genotype's robustness, the higher its phenotypic variability in response to mutations. This argument has two fundamental flaws. Firstly, the vast majority of mutations with new phenotype are deleterious (Lynch, 1999; Li 1997); secondly if we assume that a single neutral neighbour always has another neutral neighbour, the number of possible novel phenotype encountered exponentially grows. Robustness can reduce the number of new phenotypes accessible in the immediate neighbors of a genotype, but it make possible access to its neutral neighbor, their neutral neighbors and so forth, thus exponentially expanding accessible variation. Finally the naive argument that robustness prevents evolvability doesn't take into account that robustness brings together vastly connected genotype networks, and gives rise to an enormous number of possible phenotypic variations; so, individual genotype with a robust phenotype have more neutral neighbors, and thus fewer novel phenotypes in their immediate neighborhood. Robust phenotypes will more strongly affect populations encountering fewer and not more, novel phenotypes. This is the ideal situation depicted in Figure 2.1, but often the genotype space can be composed of very different sub-networks, such zones of the genotype space can hide the innovation and the discovery of new phenotypes. Figure 2.2 gives three typical scenarios of the different sub-networks composing the genotype space.

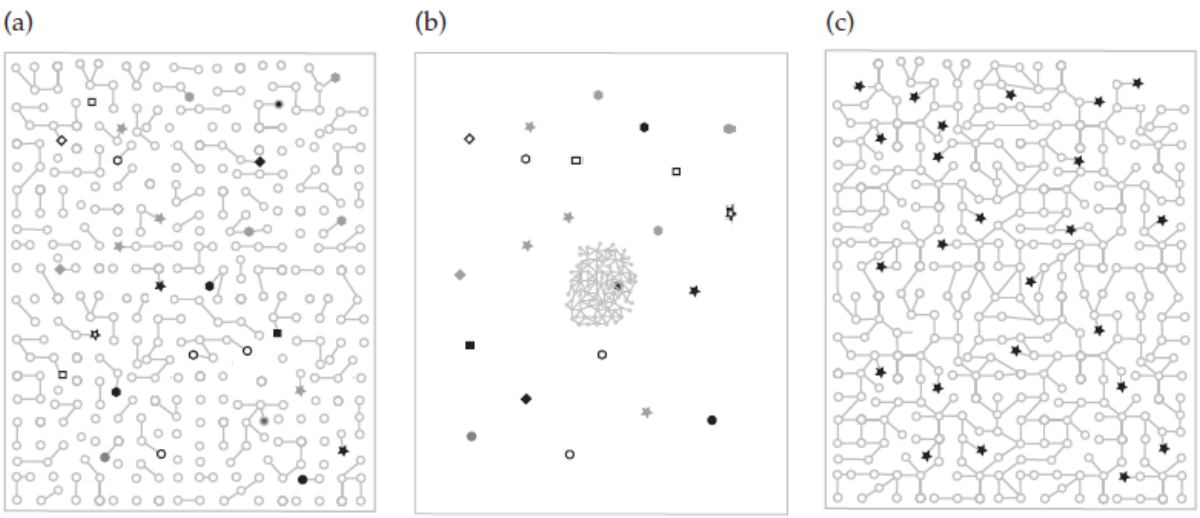


Figure 2.2. Three possible organizations of phenotype networks. See caption of Figure 1. (a) Disconnected genotype network, (b) Localized genotype network, and (c) genotype network surrounded by nodes belonging to different phenotype networks. Adapted from Wagner 2008

In the first scenario (Figure 2.2a) the phenotype networks span over a large portion of the genotype space as in Figure 2.1 but is composed by several unconnected sub-networks. A situation of this type does not facilitate the exploration of the genotype space, because genotypes that share the same phenotypes (empty gray circles) are disconnected and so not reachable trough 1-mutation. Figure 2.2b shows a case in which the phenotype network, composed by gray circles, forms a single connected network that is confined in a small portion of the search space. Also this situation does not favor the exploration of the search space since many 1-mutation always produce the same phenotype, preventing the access to phenotypes located in different areas of the search space.
Finally, Figure 2.2c exemplifies a scenario in which the phenotype network spans over a large portion of the search space, and every node is connected with at least another one, forming a huge connected network. However in this situation the novel phenotype accessible are just of one kind (black stars), so the search space doesn't not enable the access to qualitatitevely different phenotypes.

I have depicted until now how robustness work and how biological systems gain advantage from it, but how is it possible incorporate and exploit this property into artificial designed systems?
The way to achieve robustness is typically based on the relation between genotype and phenotype, this relation determines robustness as a systemic property. By acting on the genotype-phenotype map it is possible to reproduce and study the effects and the dynamics of evolutionary innovation in artificial systems, and eventually take advantage from them. Several studies have been proposed and I will next analyze relevant experiments from artificial system that have been useful for the theoretical and practical foundations of this thesis. Typical in robotics artificial studies the phenotypes are represented through the fitness function (Nolfi and Floreano 2000, Nolfi et al. 2016), that directly measure the performance of the artificial system. Differently, in genetic programming approach to artificial evolution, or in models derived from biology, the phenotype is not the fitness but the physical form of an individual, i.e. the way its component are wired or the topology of a network (Wagner 2008, Koza 1992). In both cases there is a mapping between the genotype composing the artificial individual and the its phenotype. Newman and Enghelhart (1998) modeled the genotype-fitness map with tunable robustness, as the number of genotype with the same fitness. They found that evolutionary search encountered genotypes with higher fitness more

readily when robustness is high. Van Nimwegen and Crutchfield (2000), examined one scenario typically encountered in fitness landscapes, crossing a fitness valley of inferior genotype to reach better fitness (phenotype). They show how robust population able to explore the genotype network neutrally found new phenotypes faster by orders of magnitude respect of population where neutral neighbors were absent. Elena and Sanjouan (2008) studied self-replicating logical circuits whose "phenotype" consisted in their ability to compute a set of logic functions. In long evolutionary searches, programs more robust to random changes in their computing instructions can be more effective in discovering new phenotypes. The same framework of logic functions computed with logical circuits is at the base of the work of Hu et al. (2012). They analyzed in an exhaustive manner the space of 2^{28} logic circuits (genotypes) and of 16 corresponding logic functions (phenotypes). From this analysis, they were able to reconstruct the genotype space and the neutral network the genotypes can span over the space. The space was found to be divided into 16 different fully connected neutral network corresponding to 16 phenotypes. Therefore, they found that the genotypes located in the inner core of networks are characterized by a high level of robustness to genetic variation and by a low level of phenotypic variability respect to genotype far away from the center of the net, having much variability and accessibility to novel phenotypes.

2.3 Environmental variations and evolvability

In the previous section, I analyzed how the probability of synthesizing better solutions during an evolutionary process depends on the topology of the genotype space and how this can be illustrated by using the genotype network formalism. As stressed by West-Eberhard 2003, however, phenotypic variations arise not only as a result of genetic variations but also as a result of environmental variations. "Environmentally induced phenotypic changes can give rise to adaptive evolution as readily as mutational induced changes; both are equally subject to genetic accommodation." (West-Eberhard, 2003, pp. 498). The evolutionary utility of environmental induced novelties may be greater than genetically induced ones since: they can affect all the population immediately (West-Eberhard, 2003; Whitley, 2001; Sgrp et al., 2004), can correlate with environmental conditions subjected to directional selection, and tend to persist even when they reduce the fitness of the agent, at least initially. Interaction between the expression of genetic variation and environmental condition influences the evolutionary dynamics; some expressed trait in one environment could be totally useless in a different one and will never be discovered without variations (Janssen et al., 2016). Consequently, environmental variations influence evolutionary

trajectories in populations (Wierstra et al., 2009).

The relation between environmental variation and evolvability can also be exemplified by using the genetic network framework. Consider agents situated in an environment that varies periodically between environment 1 and environment 2 denote respectively, the environment 1 genotypes with full circles and environment 2 with empty circles(Figure 2.3, top and middle panel). Finally, let assume that the genotypes filled in gray are viable in both type of environments (Figure 2.3, bottom panel).

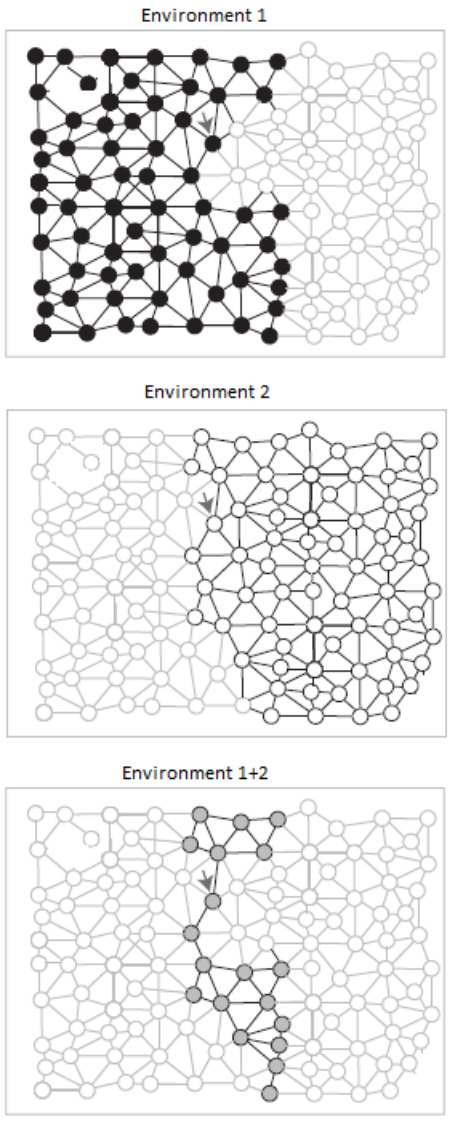


Figure 2.3. Each rectangle represents the same part of a hypothetical genotype space. The filled black circles in the upper panel and the open black circles of the middle panel correspond to parts of two genotype networks that are viable in environments 1 and 2, respectively. The gray circles in the lower panel correspond to the intersection of these genotype networks, and thus to genotypes viable in both environments. Gray arrows highlight a genotype in this intersection. This genotype, together with its neighbors on the same genotype network, is also shown to the right of each panel. Adapted from Wagner 2008

Individuals robust to environmental variations are those that develop viable phenotypes in multiple environments and are able to perform optimally in different conditions. Figure 2.3 is a simply approximation in only two environments but the consideration can be easily extended to a number n of environments. In this case the intersection between viable n-environment phenotypes became progressively smaller, however keeping in mind that the genotype space are extremely large the amount of n-viable phenotype is still very large (Wagner, 2009; Wagner, 2008).

In the context of evolutionary computation, the study of adaptation in varying environmental conditions is usually indidicated with the term dynamic optimization (Branke, 2002). Jin and Branke (2005) pointed out how these studies address different type of variations. In some cases, the evolutionary process should deal with variations caused by the fact that the calculation of the fitness is noisy. In other cases, the fitness measure constitutes only an approximation of the fitness value. In a third set of cases, the fitness function and consequently the problem to be solved vary over generations. Finally, in the fourth set of cases, the characteristics of the environment and/or of the agent vary across generations.

Previous research demonstrated how exposing evolving candidate solutions to varied environmental condition during multiple evaluation episodes can successfully lead to robust solutions in different domains: electronic circuits robust to temperature variations (Thompson and Layzell, 2002), fault tolerant neural networks (Sebald and Fogel, 1992), job shop scheduling (Tjornfelt, Jensen and Hansen, 1999), flight control under changing conditions (Blythe, 1998), robot control in varying environmental conditions (Nolfi et al., 1994; Jacoby, 1997). Individuals able to operate effectively in multiple environments can also adapt more easily to new environments, never experienced before. This since they display a greater number of traits. Consequently, they have a greater probability to posses traits that can be re-used in the new environmental conditions (Samal et al., 2010, Wagner, 2008, Crombach and Hogeweg, 2008).

Several studies analyzed the impact of environmental variations on evolutionof artificial systems. Draghi and Wagner (2009) observed that gene regulatory networks evolved in varying environmental conditions, i.e. realized by varying the fitness function, are more likely to adapt to new environmental variations. Similar results were found by Crombach and Hogeweg (2008) on another study involving gene regulatory networks. They found that networks evolved in two environmental conditions that alternated periodically over generations outperformed individuals evolved in stable environmental conditions.

Kashtan et al. (2007) showed that the exposure to varying environmental conditions, realized by varying the fitness function, speeds-up evolution expecially when the new fitness function varies within limits while preserving most of its objectives.

Moreover, the problem of adapting to varying environmental conditions change depending on whether the individuals have access to states that provide information about the current conditions or not. In the latter case, the adaptive individuals can only try to select strategies that degradate as little as possible in varied environmental conditions. In the former case, the adaptive individuals can also develop an ability to adjust their strategy to the current condition.

Individuals able to operate effectively in multiple environments can also adapt more easily to new environments, never experienced before. This, since they display a greater number of traits. Consequently, they have a greater probability to posses traits that can be re-used in the new environmental conditions (Samal et al., 2010, Wagner, 2008, Crombach and Hogeweg, 2008).

1.4 Phenotypic complexity and evolvability

Another factor that can influence evolvability is the complexity of the evolving phenotype. The indirect relationship between the genotype of evolving agents and the behavior exhibited by the agents implies that the same behavior can be generated by genotype of different complexity(Branke and Jin 2003, Raman and Wagner 2011). From the point of view of the fitness gained by the agent the possession of larger or smaller phenotype might make no difference, provided that the behavior produced by the agent is the same. From the point of view of the propensity of the agent to generate better offspring, instead, the possession of a larger phenotype might be advantageous with respect to the possession of a smaller phenotype.

An evidence of the correlation between the complexity of the phenotype and evolvability has been reported in the study of Raman and Wagner (2010) who analyzed the property of logic circuits composed by nodes performing logic functions. By systematically analyzing randomly sampled circuits of different size, the authors observed a correlaction between the size of the circuit and its evolvability defined as the number of functionally different circuits that can be generated by exploring the neutral network of the original circuit.

Wagner (2010) also pointed out a possible relation between variability of the environmental conditions, complexity of the agents, and evolvability. More specifically the author pointed out how the need to operate in variable environmental conditions can promote the development of more

complex phenotype capable of displaying multiple behaviors that are adapted to the specific environmental conditions they might encounter. This, in turn, promotes the development of more complex phenotypes that can benefit from an enhanced evolvability.

An example of how complexity is fundamental for biological system comes from bacteria and their metabolic reactions. Bacteria that live in fixed environment develop simple metabolic networks that rely on only a few different type of metabolic reactions. Instead, bacteria robust to changing environments develop more complex metabolic networks, and the chemical reactions they perform are robust to failures in one or more part of the reactions network (Wagner 2010). Complex metabolic networks are also more evolvable, i.e. have a greater probability to discover how to metabolize new nutrients as a result of genetic variations.

After the theoretical introduction of robustness evolvability and complexity, in the next chapter I will introduce in a formal way the evolutionary algorithms and the technique used in this thesis to reproduce and study how these biological properties can be reproduced and analyzed in artificial systems.

Chapter 3. Evolutionary algorithms

Evolutionary algorithms (EAs) are a "class of direct, probabilistic search and optimization algorithms gleaned from the model of organic evolution" (Back, 1996). The introduction of Evolutionary algorithms begins in the 60s, the aim is to reproduce the human brain's capability of solving problems. They are based on Darwinian's theory of evolution, which states that adaptive changes occur as result of natural selection, according to the principle of "survival of the fittest".

General-purpose algorithms can be grouped into three main areas: genetic algorithms (Holland, 1975) evolutionary strategies (Rechenberg, 1973; Schwefel, 1977; Hansen and Ostermeier, 2001; Igel, 2003; Wierstra, Schaul, Peters and Schmidhuber, 2008), and evolutionary/genetic programming algorithms (Koza, 1992; Miller and Thomson, 2000).

The first formulation of Genetic algorithms was devoted to Holland (1975), his aim was to reproduce in computer programs the natural phenomenon of genetic evolution. In Holland Algorithm, strings of fixed length, encoding individual solutions to a given problem, composes a population of solutions. This population is evolved for a certain number of generations and is subjected to selection and recombination, just as in the Darwin's theory of the evolution. Selection depends on the candidate solutions' fitness function, fitter individuals will be selected and passed to the next population; recombination consist in combining individuals' genetic strings in order to generate new offspring solutions for the next generation.

Evolutionary strategies are optimization methods belonging to the class of evolutionary algorithms,

they apply selection, recombination and mutations on a population of individual solutions to evolve always more effective solutions (Beyer, 2007). Evolutionary strategies were developed in the 70s by Rechenberg and Schwefel, and then extended by other authors (Hansen and Ostermeier, 2001; Igel, 2003; Wierstra, Schaul, Peters and Schmidhuber, 2008). Harvey (2001) and Whitley (2001) demonstrated the superiority of evolutionary strategy methods (it uses small populations, steady state selection, and rely only on mutations to generate variations) over genetic algorithms (these uses large populations, generational selection, and rely basically on recombination to generate variations). This is explained by the fact that, considering evolution, any phenotype can be

generated by a large number of alternative genotypes. Consequently, evolution can explore the search space and improve the fitness of the population over the long term (Harvey, 2001).

Genetic programming is a technique where a whole set of computer programmes instruction are evolved for a given number of generations/evaluations in order to solve a task. In genetic programming, computer programmes' instruction are encoded as set of genes that are modified by operators of mutation and crossover. Cartesian genetic programming (CGP) (Miller and Thomson, 2000) is an extension of genetic programming where individuals are encoded in strings representing coordinate of nodes connected by links. Each node is defined by three integers specifying the inputs of the node and the function the node computes. Cartesian genetic programming is the experimental core of this thesis, so, the theoreotical fundaments the mainly implementation and a literature review are the subject of the following sections.

3.1 Cartesian genetic programming

Cartesian Genetic Programming (CGP, Miller and Thomson, 2000; Miller, 2011) is a form of Genetic Programming (Koza, 1992; Langdon and Poli, 2002). It is usually used to evolve acyclic computational structures of nodes (graph) indexed by their Cartesian coordinates but can also be extended to evolve recurrent (cyclic) structures (Turner and Miller, 2014). The method has been successfully applied to evolve digital circuits (Miller, Job and Vassilev, 2000 a-b), robots' controllers (Harding and Miller, 2005), Atari games players (Wilson, Cussat-Blanc, Luga and Miller, 2018), neural networks (Khan, Ahmad, Khan and Miller, 2013), image classifier (Harding, Graziano, Leitner and Schmidhuber, 2012), molecular docking (Garmendia-Doval, Morley and Juhos, 2003) and regression programs (Harding, Miller and Banzhaf, 2009).

CGP use a vector of integer to encode a graph constituted by nodes and connections among the inputs and the outputs of the nodes. The properties of each node are encoded in a tuple of integers (genes) that encode the function of the node, chosen from a list of available functions, and the indexes of the input of the nodes. Nodes can take input from either the problem inputs or any node preceding them in the chromosome. A final set of tuples including a single integer specifies the index of the nodes that are used to produce the output of the circuit (Figure 3.1).

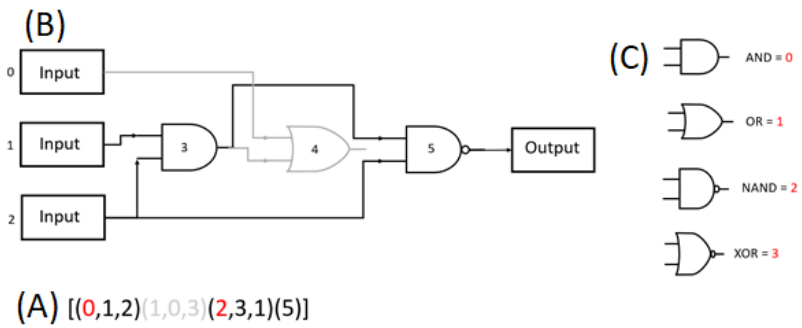


Figure 3.1. Example of a chromosome and of a corresponding CGP program with three inputs, three nodes, and one output. The sub-Figures illustrate the chromosome (A), the circuit (B) and the list of available node type (C). The first three-elements tuples of the chromosome encode the function and the input of the three corresponding nodes. The last one-element tuple encodes the index of the node that is used as output. The functional and non-functional genes/nodes are shown in black and grey, respectively. Indexes are constituted by integer numbers and are used to identify the inputs of the circuit, the nodes, and the output of the circuit.

The number of inputs and outputs are fixed and are determined based on the nature of the problem. For example, in the case of the eight-bit parity problem the number of inputs can be set to 8 and the number of output to 1. The number of gates and the functions that gates can assume is also determined manually by the experimenter based on the nature of the problem. For example, the binary operators shown in Figure 3.1 are a suitable choice for a parity problem while a list of mathematical operators such as addition, subtraction, multiplication and division, can be a suitable choice for a continuous control problem, like pole-balancing, in which the input and output states are continuous (Turner and Miller, 2015).

The initial chromosome is generated randomly. More specifically the value of the genes encoding the function of each node are generated randomly with a uniform distribution in the range [0, NGF] where NGF is the number of alternative functions included in the functions' list. The value of the genes encoding the input of a node are generated randomly with a uniform distribution in the range [0, G-1], where G is the index of the corresponding node. Offspring are generated by creating a mutated copy of the parent chromosome. Mutations are realized by replacing a fraction of randomly selected genes with a new integer selected randomly with a uniform distribution in the ranges described above. Nodes can eventually be arranged in pre-determined layers by limiting the range of the input indexes to the nodes contained in the previous layer. Usually only a small fraction of the nodes participates to the generation of the output values, because circuits tend to have a great non-functional part, gates not directly involved in output computation (Raman and Wagner 2011, Wagner 2008). Indeed, although each node has inputs, the output of a node does not necessarily have to be used by the inputs of later nodes that projected their output to the output nodes directly or indirectly. The nodes with unused outputs are non-functional since they do not influence the overall output of the program. Similarly, genes encoding nodes with un-used outputs are non-functional since they do not affect the performance of the program (functional and non-functional nodes and genes are indicated in black and gray in Figure 3.1). Non-functional nodes can be eliminated from a program without altering its behavior. However, non-functional nodes and genes that are non-functional in a circuit can become functional in the offspring of that circuit because of mutations affecting "downstream" genes. Similarly, formerly functional genes can become non-functional because of mutations affecting downstream genes (Miller and Smith, 2006). The definition of non-functional nodes can be extended to include also the nodes that project connections to the output nodes indirectly but that do not alter the state of the output. I restrict my analysis to the former type of non-functional node only.

Digital circuits can be realized in hardware or simulated in a computer. In standard electronic digital circuit the number and type of gates and the way in which they are wired is hardwired and hand-designed. In reconfigurable electronic digital circuits (such as the FPGA, see (Balch M. 2003)), instead, the logic function computed by each gates and the way in which gates are wired can be varied. In evolvable hardware applications and in simulated evolving circuits the logic functions computed by each gates and the way in which gates are wired are encoded in artificial genotypes and evolved (Koza J. 1992, Thompson A. et al. 1999, Miller J.F et al. 2000,Miller J.F. et al. 2006). Evolving circuits are selected based on their fitness which is usually computed by measuring how well the function computed by a circuit approximates a given target function.

As in several related works (Thompson et al 1999, Raman and Wagner 2011, Hartmann and Haddow 2004), I choose to provide digital circuits with a fixed number of gates since this enables us to use a simple encoding schema. For alternative approaches in which the number of gates is variable, see (Macia and Sole 2009, Miller and Hartmann 2001). However, notice that the usage of a fixed number of logic gates only limits the maximum size of the circuits. Indeed, as I will see, evolving circuits typically use a much smaller number of gates than the maximum number(Gadja and Sekanina 2010, Raman and Wagner 2011), i.e. they include several non-functional gates that do not contribute to the function computed by the circuit itself. In other words, the size of the evolving circuits varies in any case within the limit imposed by the maximum number of gates (Miller and Smith 2006). In other words, the functional size of the evolving circuits can vary freely, within the upper limit imposed by the maximum number of gates. This also implies that the number of genes encoding phenotypical components (gates) playing a functional role can also vary freely during evolution.

I decided to use digital circuits since they have been widely used in artificial evolutionary studies (Koza 1992, Thompson et al. 1999, Miller et al. 2000) and since they share with natural systems (e.g. proteins, RNA, regulatory circuits and metabolic networks) the following properties (Wagner 2011, Raman and Wagner 2011): (i) any phenotype (i.e. any circuit computing a given logic function) can originate from many different genotypes, (ii) these genotypes, giving rise to the same phenotype, can vary significantly among themselves, (iii) these genotypes span over vast genotype networks or neutral networks (Schuster et al 1994, Wagner 2008), i.e. genotypes giving rise to the same phenotype connected through single locus variation links, (iv) genotypes typically have many neighbors with the same phenotype and are thus robust to some extent to mutations, (v) the neighborhood of genotypes belonging to the same neutral network includes genotypes that give rise to rather different phenotypes.

3.1.1 Mutations

A fundamental aspetcs in CGP is the way in which an individual is mutated in order to get novel programs. In CGP mutation can be either point or probabilistic mutation. In the former, the user decides the percentage of the total number of genes of a parent genotype to be mutated to create an offspring. In probabilistic mutation every gene is considered for mutation according to a user-defined probability (Miller, 2020).

Due to its property starting from a single CGP genotype a very large variety of phenotypes can be found after a mutation. Because there is a lot of reduntant genes, often mutations occurs in non-functional zone, so, the mutated genotype has the same phenotypic expressions of the parent.

One of the very interesting aspects of the CGP representation is that a large variety of phenotypes can be found applying mutation.

Goldman and Punch, 2013, proposed a strategies to compensate the great number of mutations occurring in non-functional parts of the genotype. They proposed a single active mutation that change only the genes of functional node. Kalkreuth et al, 2015, introduced new pair of mutation operators: inactive node activation, insertion, and active node deactivation, deletion. Insertion mutation chooses an inactive node and changes one or more connection genes in the genotype to make it active. Deletion alters connections, and an active node becomes inactive.

Find the proper mutation in order to improve the evolvability of CGP is an active research topic and will be the subject of the fifth chapter of this thesis.

3.1.2 Bloat

One of the most known limit of genetic programming is called bloat, it consists in evolving programs that becomes larger over the evolutionary time (Silva et al 2007). Bloat has been described as “program growth without (significant) return in terms of fitness”, (Poli et al. 2008). CGP isn’t affected by this phenomenon as several studies showed (Turner and Miller 2014, Miller and Thompson 2000)

The presence of a large number of inactive genes, that are necessary to navigate the genotype network, could be the reason of the absence fo bloat in CGP (Turner and Miller 2014). Having large size circuit is a limitation to find promiseful and evolvable zone of the networks, as stated in chapter 2, because most of the mutation could be maladaptive. The development of large functional circuits able to navigate the network in a neutral manner is the purpose of paragraph 4.1 and the chapter five of this thesis.

This chapter was devoted to a brief introduction of evolutionary algorithms and a more extensive presentation of the Cartesian genetic programming, method that constitutes the core of the experimental part of this thesis. The next two chapters are dedicated to the experiments performed using CGP, evolutionary algorithm and artificial evolution methods: to study, exploit and analyze robustness evolvability and complexity in artificial systems

Chapter 4. Exposure to environmental variations promotes the evolution of better solutions

4.1 Introduction

Several evolutionary studies in the last decades have demonstrated how the need to cope with environmental variations produces solutions that are more evolvable, robust and responsive to novel environments. One crucial aspect behind innovation is constituted by the organism's capacity to generate phenotypic variations in response to genetic and environmental variations. In that respect it is important to consider that phenotypic variation arises not only as a result of genetic variations but also as a result of environmental variations (West-Eberhard, 2003) and that the effect of genetic and environmental variations on the phenotype depends on the organization of the phenotype itself (West-Eberhard, 2003; Wagner and Altenberg, 1996; Krischner and Gerhart, 2005).The interaction between environmental conditions and the expression of genetic variation influences the evolutionary dynamics. Genes influencing a trait in one environment may not be important in a different one (Wilke 2001). Mutations often have environment-dependent effects (Keymeulen et al. 2000, Nimwegen et al. 1999). The environmental conditions influence the genetic interactions among traits, i.e., the correlation between the genetic influences on a trait and the genetic influences of another trait, which are known to influence the evolutionary dynamics (Rana et al. 1996). For instance, the genetic correlations among certain traits can be positive in an environment and negative in another one. Consequently, environmental variations influence evolutionary trajectories in populations (Rana et al. 1996).

As already stated phenotypic variation arises not only as a result of genetic variations but also because of environmental variations. "Environmentally induced phenotypic changes can give rise to adaptive evolution as readily as mutational induced changes; both are equally subject to genetic accommodation." [West-Eberhard 2003, pp. 498].

In this chapter, I use two different domains (boolean circuit and evolving agents) to modelize the effects of environmental variations in artificial evolutionary experiments.

In Section 4.2, I demonstrate how the need to cope with operational faults in circuits leads to finding better solutions than those found in control experiments in which circuits are not subjected to faults. The analysis of the results obtained in different experimental conditions also indicates that, in absence of faults, evolution tends to select circuits that are small and have low phenotypic

variability and evolvability. The need to face operation faults, instead, drives evolution toward the selection of larger circuits that are truly robust with respect to genetic variations and that have a greater level of phenotypic variability and evolvability.

In Section 4.3 I analyze the impact of environmental variations on the evolution of neuro-agents situated in an external environment. More specifically, I study the conditions that promote the evolution of agents robust to environmental variations. I demonstrate how agents evolved in environments that vary across generations outperform agents evolved in environments that remain fixed. Moreover, I demonstrate that best performance is obtained when the environment varies at a moderate rate across generations, i.e. when the environment does not vary every generation but every N generations. The advantage of exposing evolving agents to environments that vary across generations at a moderate rate is due, at least in part, to the fact that this condition maximizes the retention of changes that alter the behavior of the agents, which in turn facilitates the discovery of better solutions. Finally, I demonstrate that moderate environmental variations are advantageous also from an evolutionary computation perspective, i.e. from the perspective of maximizing the performance that can be achieved within a limited computational budget.

4.2 The role of random-fault in the evolution of digital circuits

4.2.1 Introduction

The objective of the work described in this section is to verify experimentally whether the need to cope with environmental variations promotes evolvability, i.e. whether digital circuits evolved in variable environmental conditions discover more fit solutions than circuits evolved in stable environmental conditions. In particular, I investigate whether the need to cope with internal variation caused by component faults can promote evolvability.

Here I define *phenotypic variability* as the propensity of an individual or of a population to generate different unique phenotypes because of genetic variation. Variability should not be confused with variation, which refers to actual variations occurring between the individuals of a population (e.g. the heterozygosity or the degree of polymorphism of a population). Moreover, I define the term *evolvability* as the propensity of genetic variations to sometime produce phenotypic adaptations. Phenotypic variability and evolvability indicate the potential or the propensity to vary and to improve, respectively, and "thus belong to the group of 'dispositional' concepts, such as solubility" (Wagner GP and Altenberg 1996). Notice that although phenotypic variability likely correlates with evolvability, a high phenotypic variability does not necessarily imply a high evolvability. Multiple definitions of the term evolvability are in use (Wagner GP and Altenberg 1996,

Krischner and Gerhart 2005, Wagner 2008, Sniegowsky and Murphy 2006). I adopted the definitions reported above since they enable us to distinguish between variations and adaptive variations and since they can be clearly operationalized in the case of my experiments. Indeed, they can be measured by generating from an evolving circuit a large number of genetically varied circuits and by counting the fraction of different unique phenotypes and the fraction of fitter phenotypes (for more details see below).

The relation between robustness to genetic variations, robustness to fault tolerance, and phenotypic variability in evolving digital circuits has already been investigated in several studies. By exploring the space of 10^{45} logic circuits (genotypes) and of 10^{19} corresponding logic functions (phenotypes), (Raman and Wagner 2011) observed that: (i) the robustness of circuits with respect to mutation and to faults is high on the average, (ii) different circuits with the same phenotype have a broad distribution of robustness to genetic variations, with some circuits being much more robust than others, and (iii) larger circuits are more robust to mutations than smaller circuits. Larger circuits are more robust than smaller circuits also with respect to mutations that affect the components of the circuit that actively contribute to the output of the overall circuit. Moreover, the authors observed how neutral evolution tends to select circuits that have a high robustness with respect to mutation but a low phenotypic variability (Raman and Wagner 2011). The neutral evolutionary process was realized by choosing a circuit computing a given function, generating an initial population composed of identical copies of the same circuit, generating varied copies of the circuits, and selecting the variations that preserve the function computed by the original chosen circuit. Hu et al 2011 and Hu et al 2012, instead, analyzed in an exhaustive manner the space of 2^{28} logic circuits (genotypes) and of 16 corresponding logic functions (phenotypes). The circuits were constituted by 2 inputs, 4 gates and 1 output. Their analysis revealed that: (i) the genotype space is divided into only 16 fully connected neutral networks corresponding to 16 phenotype networks, (ii) the size of the networks varies significantly, and (iii) the genotypes located in the innermost core of the networks are characterized by a high level of robustness to genetic variation and by a low level of phenotypic variability. Moreover, by analyzing the course of random walk exploration processes carried out from a randomly selected genotype belonging to a given network toward the first encountered genotype belonging to a different network, the authors observed that: (i) different networks have rather different accessibility levels (i.e. probability to be reached through neutral and/or adaptive variations), (ii) the time necessary to reach a network through neutral and/or adaptive variations is correlated with the accessibility of the network (iii) the accessibility of a network is correlated with the robustness to genetic variations of the genotypes forming the

network, and (iv) genotypes robust to genetic variations are more likely to be reached than less robust genotypes.

Finally, in a series of studies conducted by evolving digital circuits for the ability to perform a given target function, the authors demonstrated how forcing the circuits to operate in the presence of failures of circuit components lead to the evolution of circuits that are more robust against these faults (Thompson et al 1999, Keymeulen et al 2000, Macia and Sole 2009). The robustness of evolving circuits is not achieved through the development of redundant solutions, i.e. solutions that include multiple copies of components and in which the failure of one component is compensated by the activity of another identical component. Rather robustness is achieved through degeneracy, i.e. through the ability of structurally different components to perform the same function (Tononi et al 1999, Edelman and Gally 2001).

In general terms, as discussed by Wagner (2008), the relationship between robustness to genetic variations and phenotypic variability is characterized by both antagonistic and synergetic factors. In fact, from the perspective of a specific genotype, the higher robustness to genetic variation of the genotype is, the lower phenotypic variability of the genotype is. On the other hand, from the perspective of a specific phenotype (i.e. from the perspective of the neutral network that includes all the genotypes connected through single genetic variations that give rise to the phenotype), the higher the robustness of the genotypes forming the neutral network, the higher the variety of the phenotypes that can be accessed in the neighborhood of the neutral network of the corresponding phenotype is (Wagner 2008). Apparently, in some cases, the interplay of these factors can have an overall negative effect on phenotypic variability while in other cases it can have a positive effect. Indeed, by studying simulated RNA molecules that evolve toward a predefined target shape in a constant environment, (Ancel and Fontana 2000) observed a dramatic loss of variability throughout generations that ultimately traps populations in regions where most genetic variation is phenotypically neutral (a phenomena named “neutral confinement”). On the other hand, the systematic exploration of the genotype and phenotype space of simulated RNA molecules reported in (Wagner 2008) indicates that populations with robust phenotypes have a higher phenotypic variability than populations with less robust phenotypes.

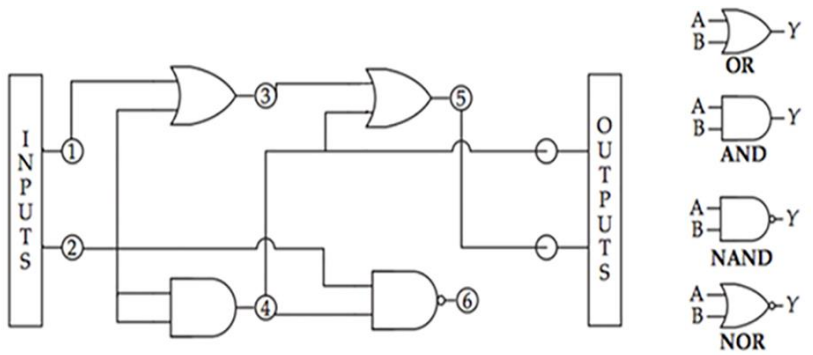
In this section I analyze whether the occurrence of faults drives the evolutionary process toward circuits with a high phenotypic variability and with a high evolvability. Moreover, I analyze whether the occurrence of faults enables the evolutionary process to find more fit circuits with respect to a control condition in which circuits are not subjected to faults. For sake of clarity, I define phenotype

as the logic function computed by a given circuit, as in (Raman and Wagner 2011). This must be pointed out, because in many works related to genetic programming the phenotype represent the phisycal form of the circuit (Koza 1992, Miller and Thompson 1999). Anyway, in this thesis I will follow the terminology adopted in (Wagner 2008, Wagner 2010, Raman and Wagner 2011) where the function that any one circuit computes is an analogue to a biological phenotype.

The results demonstrate that the need to cope with faults promotes the selection of phenotypically variable and evolvable circuits, and this, in turn, speeds up the evolution of effective circuits.

4.2.2 Method

Digital circuits (Figure 4.1) are systems that compute logic functions, such as the multiplication of digital numbers, by receiving as input two or more binary (Boolean) values and by producing as output one or more binary values, as described in chapte 3. In the experiments reported in this section I evolved simulated digital circuits with four inputs, 256 logic gates divided into 16 layers of 16 gates, and one output for the ability to compute a 4-bit even parity function (i.e. to produce as output 1 when there is an even numbers of 1 in the input pattern and 0 otherwise), this configuration cames from (Raman and Wagner), figure 4.1 give an exemplification of the circuit.



1	2	0	2	2	1	3	4	0	2	4	2	4	5

Figure 4.1. Top. A digital circuit with two inputs, two outputs, and four gates. The right side of the panel shows the four symbols that correspond to the four kinds of permissible logic gates. The numbers 1-2 indicate the binary states that are provided as inputs to the circuit (input pattern). The numbers 3-6 indicate the output computed by the four corresponding logic gates. The output of the circuit corresponds to the output of the two logic gates that are wired to the output pattern (4 and 5). The lines indicate the wiring of the circuit. Circuits can be composed by different numbers of rows and columns, each gates can receive inputs only by gates of the previous columns. **Bottom**. The genotype of the circuit is shown. The first 4 vectors of 3 numbers encode the characteristics of the four gates (from top to bottom and from left to right). The blue numbers encode the input states of each gate. The black numbers encode the function

computed by each of the four gates (i.e. 0=OR, 1=AND, 2=NAND, 3=NOR). The last two numbers shown in green encode the output states of the circuit.

This particular configuration of the circuits and the task are chosen to maintain the problem parameters reasonably small. Choosing larger circuits or more complex task would have been led to a genotype and phenotype space exponentially larger and the analysis of these spaces would have been more difficult and computationally much more expensive.

To evolve the circuits I start with an initial population of 20 randomly generated genotypes encoding 20 corresponding digital circuits. Each circuit was allowed to produce an offspring, i.e. a mutated copy of the parent genotype in which genes were replaced with a number randomly generated with an uniform distribution in the appropriate range, according to a mutation rate (*MutRate*). Each mutant's fitness was then compared to the lowest fitness in the parent population, if the mutant was better or equally good, it replaced the parent. The selection and reproduction process were repeated for a certain number of generations. More formally :

Algorithm

1. Generate random population of 20 individuals (Parents P_i)
2. Evaluate Parents and retrieve fitness($F(P_i)$)
3. For each member of the population create an offspring by mutation(M_i) and evaluate them ($F(M_i)$)
4. Order Parents and Offrsping by fitness, the best 20 individuals are the new parents
5. Until generation limit reached go to step 2

Circuits are evaluated for the ability to map the 2^n possible input patterns into the corresponding desired outputs (i.e. 1 for input patterns with an even number of 1 and 0 otherwise). I choose this function since it constitutes a rather difficult problem for evolving circuits including OR, AND, NAND, and NOR logic gates (Miller and Thomson, 2000). More specifically the fitness is calculated based on the following equation:

$$F = 1 - \frac{1}{2^n}\sum_{j=1}^{2^n}\left|O_j - E_j\right| + \varepsilon \qquad (1)$$

Where n is the number of inputs of the circuit, j is the number of the input pattern varying in the range $\{1, 2^n\}$, O_j is the output of the circuit for pattern j, E_j is the desired output for pattern j and ε is a noise value randomly selected in the range {-*Stochasticity*, *Stochasticity*} with a uniform distribution. The maximum values of Stochasticity varies for different experimental settings. Due to the particular fitness of the task, that is discrete and with a fixed step landscape, I chose to have a uniform distribution of noise because this ensure the tuning of how much is possible to jump for an individual to a new fitness level. In this domain Stochasticity prevents neutral dift but enables explorations, so I decided to add it to the algorithm. As showed in the results section adding it is a boost for the performance, hower comparison with settings where stochasticity is absent and neutral drift is possible are reported.

The function of noise is that to make the selection process probabilistic (Jin 2005). I decided to use this technique, rather than probabilistic selection operators such as roulette wheels or tournament selection, since its impact can be tuned quantitatively by varying a single parameter and since it is qualitatively similar to the stochastic variation of fitness caused by uncontrolled variations occurring in non-deterministic settings. As examples of non-deterministic setting consider the case of circuits subjected to random operation faults in which the effect of the faults is stronger or weaker depending on the malfunctioning gates or the case of evolutionary robotics experiments in which the fitness scored by a robot depends also on aspects that are variable (e.g. the initial position/orientation of the robot in the environment and/or the positions of the obstacles in the environment, see (Nolfi and Floreano 2000).
In the fault experimental condition I take in consideration the condition where each logic gate can fails with a certain probability, i.e. responds to its input by producing the wrong output. I don't consider faults in wiring among gates to don't complexify the problem too much.

To promote the evolution of circuits robust to faults affecting different gates and combinations of gates, I evaluated the circuit for one trial without faults and for 200 trials with faults; the number of 200 trials is derived empirically looking at how much trials are needed to ensure a good amont of faults in circuit of different functional size. Each trial includes all possible 2^n input patterns. The fitness is computed by averaging the fitness scored in the no-fault trial (calculated based on Eq. 2) and the average fitness scored on the worst trials with faults, so giving the same importance to both components:

$$F_t = \frac{F_n + F_f}{2} \tag{2}$$

$$F_f = \frac{1}{W_t}\sum_{j=1}^{Wt} F_j \ , W_t \in [0{,}200] \tag{3}$$

Where W_t is the number of worst trials considered to calculate the fitness scored on fault trials, i.e. only the trials that affects performance, trials with faults in non-functional part of the circuits are not considered so the number can be variable within 0 and 200. F_n is the fitness scored during the no-fault trials calculated in the basis of eq.1, F_f is the average fitness scored during the worst fault-trials, and *200* is the total number of trials carried out with faults. Finally, to promote the evolution of progressively more robust solutions, I called fault rate the probability f_r that a logic gate has a fault (that is initially set to 0.3) is increased or decreased at the end of each generation when the average fitness scored during the worst trials is larger or smaller than ½ of the fitness scored in the no-fault trial, this ensure to maintain an appropriate fault pressure during all the evolution. So :

$$f_r \ \pm f_r * \left(\frac{F_f}{F_n} - \frac{1}{2}\right)/v_f \ , \qquad 0.04 \leq f_r \leq 0.8 \tag{4}$$

Where f_r **,** that is constrained in the range {0.04, 0.8} and start from 0.3, is the rate that gates undergo to faults and v_f is a constant that determines the variation rate of f_r . The rationale behind this variation of the fault probability is that it enables to maintain a more constant level of variation to select upon. Comparative experiments performed by using constant fault probabilities led to worse performance (results not shown). The reason why I decided to use only the worst fault trials

for the calculation of the fitness instead than all trials is that this increases the selective pressure toward the selection of circuits that are robust with respect to operation faults. A circuit is considered robust to fault when after a failure in one or more of its functional gates it still performs as in absence of faults.

I will use the term behavior to indicate the outputs produced by a circuit in response to each possible input pattern. Moreover, I will use the term functional size to indicate the number of gates that actively contribute to the outputs produced by the circuit. Notice that circuits having the same fitness might differ at the level of the behavior produced. Indeed, circuits producing different outputs can produce the same number of correct and incorrect responses. Notice also that circuits displaying the same behavior might differ at the level of the circuit's components. Indeed, circuits characterized by different type of gates or different wiring can produce the same outputs. The fact that the number of circuits that differ with respect to the type of the gates and/or the way in which the gates are wired is much greater than the number of circuits that differ at the level of the fitness implies that a large portion of genetic variations are neutral, i.e. produce variations at the level of the circuit and/or at the level of behavior that do not alter the fitness of the circuit.

4.2.3 Results

In this and in the following section I report the results obtained in the no-fault experimental condition. The results obtained by subjecting evolving circuits to faults are reported later in the section.

To verify the role of the mutation rate and of stochasticity I measured the percentage of experiments that lead to optimal solutions within 6,000 generations for different values of the parameters (Table 2.1). As can be seen, in the case of the best combination of parameters (Stochasticity=0.05 and MutRate=0.02), evolvid circuits find optimal solutions in 50% of the replications of the experiment at the end of the evolution. By continuing the evolutionary process for 300,000 generations (Fig 2.2) evolving circuits manage to achieve optimal performance sooner or later. However, the discovery of optimal solutions might require a rather long evolutionary time, i.e. up to 300,000 generations.The data reported in Table 2.1 also show that the introduction of moderate level of noise in the selection process leads to better results (for similar results see Bäck and Hammel 1994, Levitan and Kauffman 1994, Rana Whitlev and Cogswell 1996). This can be explained by considering that the addition of noise enables the selection of a limited number of less fit individuals, i.e. it enables a reduction on selection pressure. This in turn increases the variation

among the individuals of the population and reduces the risk of premature convergence. When stochasticity is zero many genotypes will have the same fitness as the offspring of parents may only differ in non-coding region, to this reason I skipped the evalution of same functional offspring ensuring that all the experiment had the same number of evalutation

	MutRate 0.01	MutRate 0.02	MutRate 0.03	MutRate 0.04	MutRate 0.05
Stochasticity 0.0	26.66%	33.33%	33.33%	36.66%	36.66%
Stochasticity 0.05	33.33%	50%	46.67%	40%	36.66%
Stochasticity 0.08	26.66%	26.66%	33.33%	36.66%	30%
Stochasticity 0.1	20%	30%	30%	30%	26.66%

Table 4.1. Percentage of evolutionary experiments that achieve maximum fitness in the no-fault condition in 20 experiments carried out with different mutation rate and stochasticity range. Each experiment has been replicated 30 times and continued for 6000 generations.

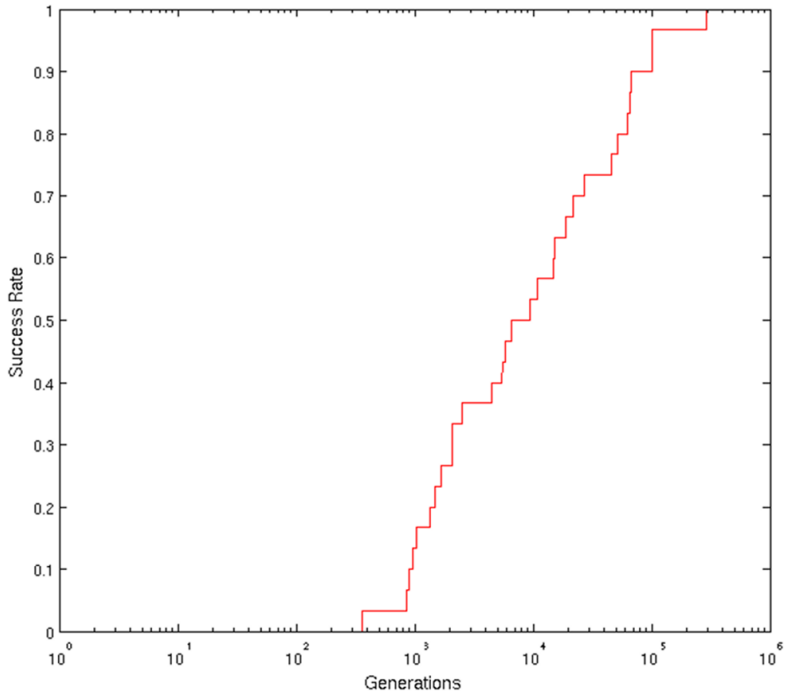


Figure 4.2. Percentage of evolutionary experiments that achieve maximum fitness throughout generations in the no-fault condition. Data obtained by running 30 replications of the experiment for 300,000 generations with Stochasticity=0.05 and MutRate=0.02.Noticethat the x-scale is logarithmic.

4.2.3.1 Evolution leads to the selection of small circuits with low evolvability

The analysis of evolved circuits in the no-fault condition indicates that, overall, they are very robust with respect to mutations. Indeed, by analyzing the effect of single point mutations I observed that the large majority of them are neutral (Table 2.2). This high robustness, however, is largely due to the fact that the size of the evolved functional circuit is small, as showed in table 2.2 the 95% of mutations are neutral because for vast majority them occurs in non-functional part. Indeed, restricting the analysis to the mutations that affect the functional part of the circuits (i.e. to the mutations that alter the characteristics of the logic gates that actively contribute to the output of the overall circuit), the percentage of neutral mutations drops from 95% to 9%.

	% Neutral Mutations
Overall	95%
Functional	9%

Table 4.2. Percentage of single point mutations that do not alter the fitness of the circuit. Data obtained by subjecting the best circuits of generation 6,000 to 10,000 single point mutations. The “functional” data refer to the mutations affecting the genes of functional gates (i.e. the gates that actively contribute to the output of the circuit). In this and in

the following Tables/Figures of this section I refer to the 30 replications of the evolutionary experiment carried out in the no-fault condition with Stochasticity=0.05 and MutRate=0.02.

The factor that drives evolution toward the selection of small circuits (i.e. circuits with a small functional part) is the fact that smaller is a circuit with respect to the other individuals of the population, the larger the probability that its offspring will receive mutations that do not affect its functional gates, and consequently, the higher is the probability that its offspring will have a relative higher fitness. This is similar to the protection hypothesis that postulates that non-functional coding regions of the genotype might protect the evolving individuals from the deleterious effect of crossover (Miller and Thomson, 2000).

This is demonstrated by the fact that the number of logic gates that actively contribute to the output of the circuit is only 30.4 and 18.3, on the average, in the case of circuits evolved for 6,000 generations displaying optimal or sub-optimal performance, respectively. Moreover, it is demonstrated by the fact that in most of the cases the size of the functional part of evolved circuits decreases with increasing generations, when they are subjected to a neutral evolutionary process in normal conditions (Fig 4.3, top, Wilcoxon Rank Sum Test, $p < 0.001$) while the size increases in a control condition in which selected offspring are used to replace only their own parent (Figure 4.3, bottom, Wilcoxon Rank Sum Test, $p < 0.01$). Notice that the latter condition corresponds to a situation in which the population is divided into 20 different single-individual sub-populations that evolve independently without competing with each other during selection. The tendency to select functionally small solutions, therefore, originates because of the selection process (only the best individuals reproduce) and as the result of the fact that smaller circuits have a greater probability to generate viable offspring than larger circuits.

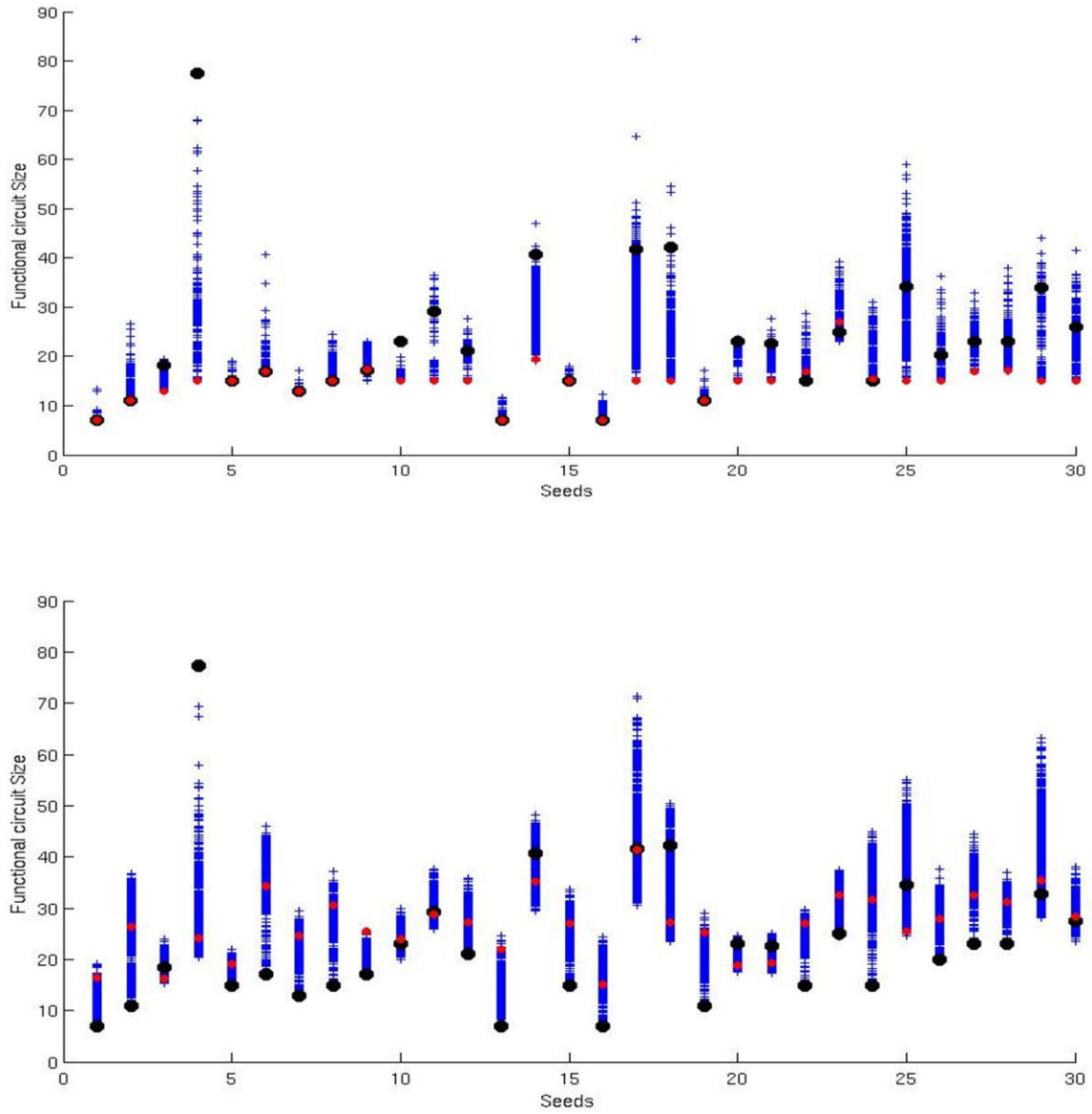


Figure 4.3. Variation of the functional size of evolved circuits while they are subjected to a neutral evolutionary phase for 100,000 generations. During the neutral phase, individuals are evaluated on the basis of a binary fitness function that assigns a fitness of 1.0 to circuits that compute the same function of the initial circuits and a fitness of 0.0 otherwise. Circuits with fitness 1.0 and 0.0 are always selected or discarded, respectively. The black disks represent the average size of the evolved circuits before neutral evolution. The blue crosses indicate the average size every 100 generations during neutral evolution. Finally, the red disks represent the average size of circuits at the end of the neutral evolutionary phase. The top figure shows the results obtained in a normal condition in which selected offspring are used to replace one of the parents chosen randomly (all parents compute the same logic function and have the same fitness). The bottom figure shows the results obtained in a control condition in which offspring are used to replace their own parent. Data obtained by subjecting to neutral evolution 30 populations each composed of 20 circuits evolved for 6,000 generations.

Unfortunately, however, the size of the evolving circuits strongly correlates with the phenotypic variability of the circuits, i.e. with the number of new unique phenotypes that can be found in the neighborhoods of the circuits (Fig 2.4, Spearman Test, rho 0.90229, phi $1.1153 * 10^{-220}$ n = 600). This implies that evolution in normal conditions leads to the selection of functionally small circuits

with low phenotypic variability and low evolvability (see also the results of the comparison with circuits evolved in the fault condition described below).

Raman and Wagner 2011 have already found a correlation between circuit size and phenotypic variability. In their case, the correlation was observed by comparing randomly generated circuits of different size that computed the same logic function. The correlation, therefore, seems to characterize all circuits, irrespectively from whether they were evolved or not and irrespectively from the function that they compute.

The fact that the growth of non-coding regions of the genotype can protect evolving individuals from deleterious genetic variations has already been pointed out in previous works (Miller and Thomson 2000). Here I show that this protection can be achieved by shrinking the coding regions at the cost of a reduced phenotypic variability and evolvability.

The tendency of the population to move toward functionally small circuits during neutral evolutionary phases is a consequence of: (i) the fact that under neutral evolution the population tends to concentrate toward highly connected parts of the neutral network that correspond to individuals that are relatively robust against mutations (Nimwegen et al. 1999, Newman 2010, Wilke 2001), and (ii) the fact that individuals that are robust against mutations generally correspond to functionally small circuits

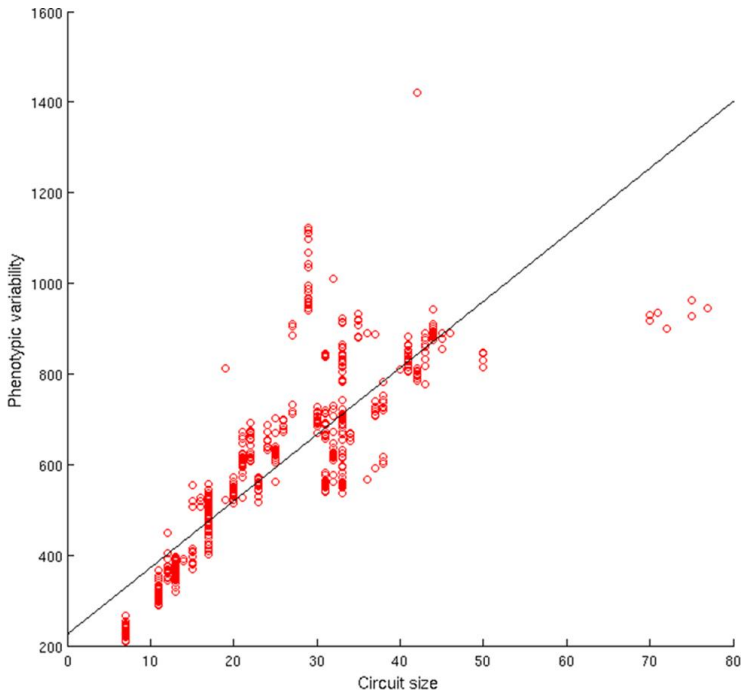


Figure 4.4. Distribution of the number of new unique functions found in the neighborhoods of evolved circuits and of the size of the circuits. Data obtained by analyzing the neighborhood of 600 evolved circuits obtained by running 30 evolutionary experiments for 6,000 generations. For each circuit, phenotypic variability has been calculated by counting the number of new unique functions computed by sample circuits found in the neighborhood of the original circuit. Sample circuits have been selected by performing for 10 times a 1,000 steps function-preserving random walk from the original circuit. During each step I generated a varied circuit, through the application of a single point mutation, that was included in the sample set and used to generate additional varied circuits or discarded, depending on whether it computed the same function of the original circuit or not.

The analysis of the relation between robustness to genetic variation and evolvability reveals a strong negative correlation with respect to overall variations (Fig 4.5, top, Spearman Test rho –0.87538, phi $6.4189 * 10^{-191}$ n = 600) and a strong positive correlation with respect to variations affecting the functional components of the circuits only (Fig 2.5, bottom, Spearman Test rho 0.70301, phi $6.4189 * 10^{-191}$ n = 600). This can be explained by considering that the circuits that are more robust to genetic variations overall are generally circuits with a small number of functional gates (see Figure 4.5). The robustness of these circuits with respect to genetic variations, therefore, is due primarily to their small size and not to a genuine ability to compensate the effects of variations. On the contrary, the circuits that are robust with respect to variations affecting their functional components are larger and are genuinely robust, i.e. are able to compensate largely the effects of genetic variations affecting functional gates (Fig 4.5, bottom). Larger circuits that are genuinely robust to genetic variations are also more evolvable with respect to smaller circuits that are robust only thanks to their small size.

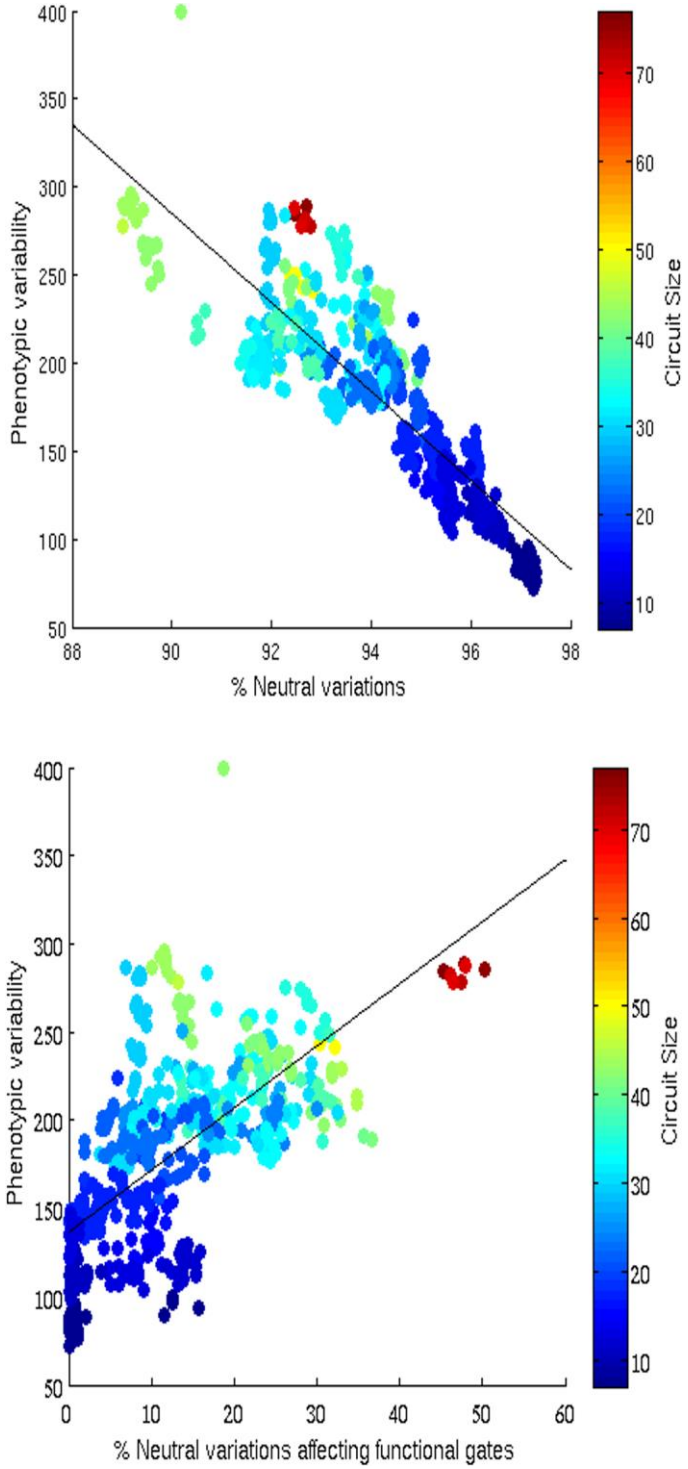


Figure 4.5. Distribution of phenotypic variability and robustness in evolving circuits. The top and the bottom figures show the correlation between phenotypic variability and robustness to variations overall and to variations affecting the functional gates only, respectively. The color of each dot indicates the size of the corresponding circuit. Phenotype variability corresponds to the number of neighborhood circuits computing different unique functions and has been calculated by using the same procedure described in the caption of Figure 2.4.

4.2.3.2 The need to face faults promote evolvability

In this section, I report the results obtained by evolving circuits in the fault condition, i.e. in a condition in which each circuit gates, is subjected to fault during operation with a certain probability.

To identify the optimal value of the two additional parameters that characterizes the evolutionary process in the fault condition I systematically varied the fraction of worst trials considered in the fitness function and the rate of variation of the fault frequency across generations. As shown in Table 4.3, the values of the parameters that maximize the percentage of replications leading to optimal performance in 6,000 generations are WTrials=25% and vFaultRate=5 (see Table 4.3). For the mutation rate I kept the same value that resulted optimal in the no-fault condition (MutRate = 0.02). The stochasticity range was set to 0.0 given that in the fault condition the fitness measure is already subjected to the stochastic effects caused by the randomly occurring faults, i.e. by the fact that the fitness loss caused by faults depends on the specific gates that are affected by faults. Indeed, using two different form of stochasticity result in very noisy behavior, so, in this case the usage of a stochasticity range greater than 0.0 is counterproductive (see Table 4.4) and not beneficial as in the case of the no-fault condition (Table 4.1).

	vFaultRate=1	vFaultRate=3	vFaultRate=5	vFaultRate=7	vFaultRate=9
WTrials=35%	46.66	43.33	50	53.33	43.33
WTrials=30%	50	50	**60**	53.33	40
WTrials=25%	53.33	53.33	**60**	46.66	40
WTrials=20%	50	46.66	53.33	40	36.66
WTrials=15%	46.66	46.66	50	43.33	36.66
WTrials=10%	33.33	30	33.33	30	26.66

Table 4.3. Percentage of evolutionary experiments that achieved maximum fitness with different values of the vFaultRate and wTrial parameters. The MutRate and Stochasticity parameter are set to 0.02 and 0.0, respectively. Data obtained by running 30 replications, each lasting 6,000 generations, for each combination of parameters. The percentage of success refers to the performance achieved during the trial in which the circuits are not subjected to operational faults.

Stochasticity=0.0	Stochasticity=0.01	Stochasticity=0.02	Stochasticity=0.03	Stochasticity=0.04	Stochasticity=0.05
60	36.66	30	33.33	23.33	20

Table 4.4. Percentage of evolutionary experiments that achieve maximum fitness in experiments carried out with different values of the Stochasticity parameter. Data obtained by running 30 replications lasting 6,000 generations for each value of the parameter. The Wtrial, vFaultRate, and MutRate parameters have been set 25%, 5, and 0.02, respectively. The percentage of success refers to the performance achieved during the trials in which the circuits are not subjected to operational faults.

The comparison of the results indicates that, as expected, the circuits evolved in the fault condition are more robust with respect to genetic variations affecting their functional components than the circuits evolved in the no-fault condition (Table 4.5). Indeed, the percentage of variations affecting functional gates that do not produce any loss in performance is 2.07% and 0.99% on the average, in the case of fault and no-fault circuits, respectively. The circuits evolved in the no-fault conditions are more robust with respect to overall variations than the circuits evolved in the fault condition. As discussed above, however, this does not reflect a genuine robustness but simply the fact that circuits evolved in the no-fault conditions are smaller than the circuits evolved in the fault condition.

The circuits evolved in the fault condition are larger than those evolved in the no-fault condition from generation 6,000 on (see Fig 4.6, left).

The circuits evolved in the fault condition have a greater phenotypic variability (Table 4.5).

Finally, the circuits evolved in the fault condition have a greater evolvability. This is demonstrated both by the fact that the percentage of genetic variations leading to improvements is higher in the case of the circuits evolved in the fault condition (Table 4.5) and by the fact that the circuits evolved in the fault condition achieve better performance from generation 6,000 on with respect to circuits evolved in the no-fault condition (Fig 4.6, right). As expected, the probability that random genetic variations lead to improvement (evolvability) is rather low in both cases (Table 4.5). However, it is higher for circuit evolved in the fault than in the no-fault condition. Notice that the evolvability is necessarily 0 for circuits displaying optimal performance, because I have defined it as the propensity to generate better solution,so, by definition a circuit displaying optimal performanc cannot improve. Since the number of optimal circuits evolved in the fault condition is greater than the number of optimal circuits evolved in the no-fault condition, the difference in evolvability between the two conditions is even greater than that reported in Table 4.4. This strengthens my conclusion that circuits evolved in the fault condition have a greater evolvability than circuits evolved in the no-fault condition.

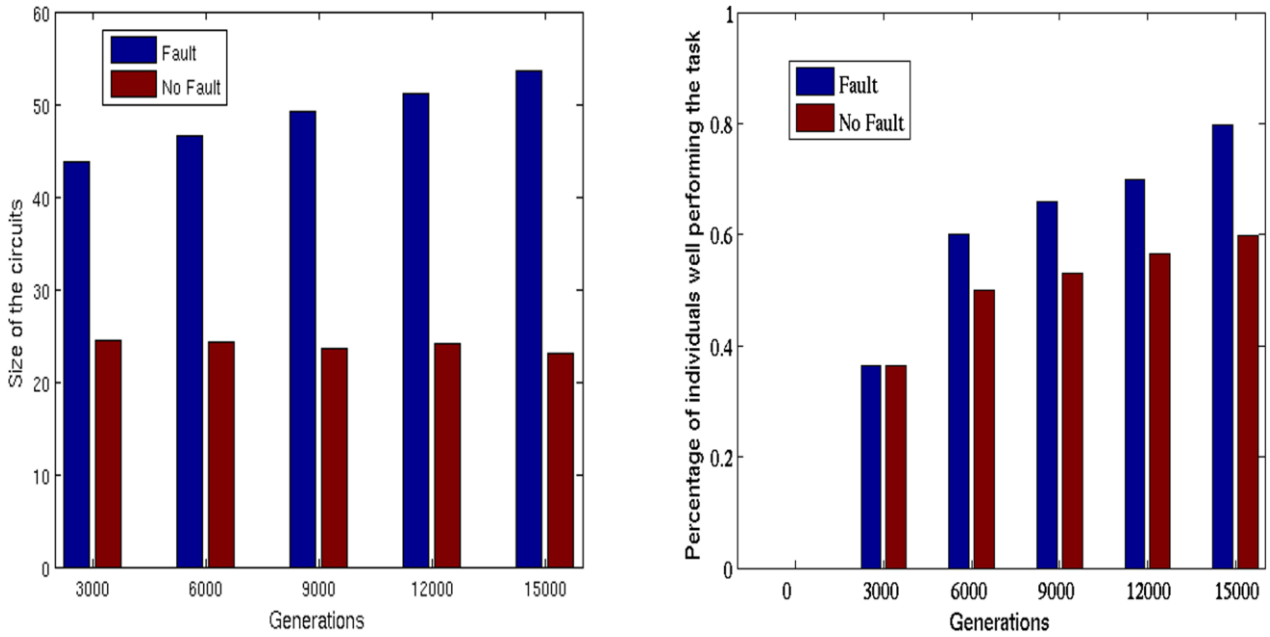


Figure 4.6. Left: Average size of functional circuits throughout generations in the no-fault and fault conditions. **Right**: Fraction of replications achieving optimal performance throughout generations in the no-fault and fault conditions.

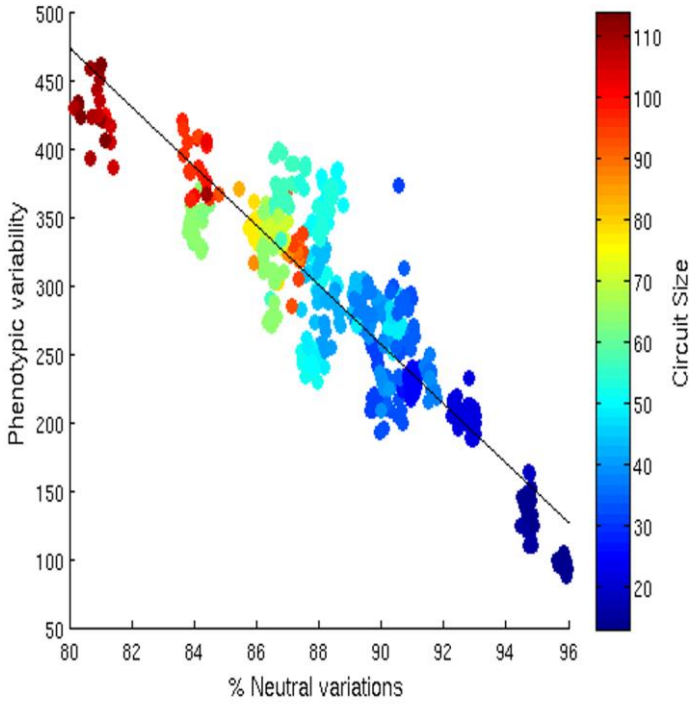
Data obtained by running 30 replications in each condition for 15,000 generations with the best parameters (i.e. Mutrate = 0.02 andStochasticity= 0.05 in the no-fault condition; Mutrate = 0.02, stochasticity= 0.0, Wtrial=25% and vFaultRate=5, in the fault condition). The experiments conducted in the fault condition are considered successful when the evolving circuits display optimal performance during the trial in which the circuit component are not subjected to fault. In other words, the data displayed in the right figure are not influenced by the level of robustness of the circuits with respect to operational faults. Wilcoxon rank sum tests p-value < 10^{-63}.

	No-fault	Fault	p-value (Wilcoxon Rank Sum test)
% neutral variations	94.2601	89.3427	$<10^{-13}$
% maladaptive variations	5.7389	10.6559	$< 10^{-13}$
% adaptive variations(evolvability)	0.0010	0.0014	$< 10^{-5}$
% functional neutral variations	0.9986	2.0705	$< 10^{-5}$
% functional maladaptive variations	4.0953	7.6177	$<10^{-14}$
% functional adaptive variations	0.0008	0.0011	$< 10^{-4}$
number of new unique phenotypes (phenotypic variability)	583.42	914.80	$<10^{-9}$

Table 4.5. Characteristics of the neighborhoods of circuits evolved in the no-fault and fault conditions. Functional variations refer to variations affecting the gates that actively contribute to the output of the circuits. Data obtained by analyzing 600 evolved circuits obtained by running 30 evolutionary experiments for 6000 generations in each condition. Each circuit was subjected for 100 times to a 1,000 steps function preserving random walk (see caption of Figure 4.4).

The analysis of the relation between robustness to genetic variations and phenotypic variability reveals, also in this case, a strong negative correlation with respect to overall variations (Fig 4.7, top, Spearman Test rho –0.8679, phi $7.2153 * 10^{-184}$ n = 600) and a positive correlation with respect to

variations affecting the functional gates of the circuits only (Fig 2.7, bottom, Spearman Test rho 0.54292, phi 2.6872$*$ 10^{-47} n = 600).



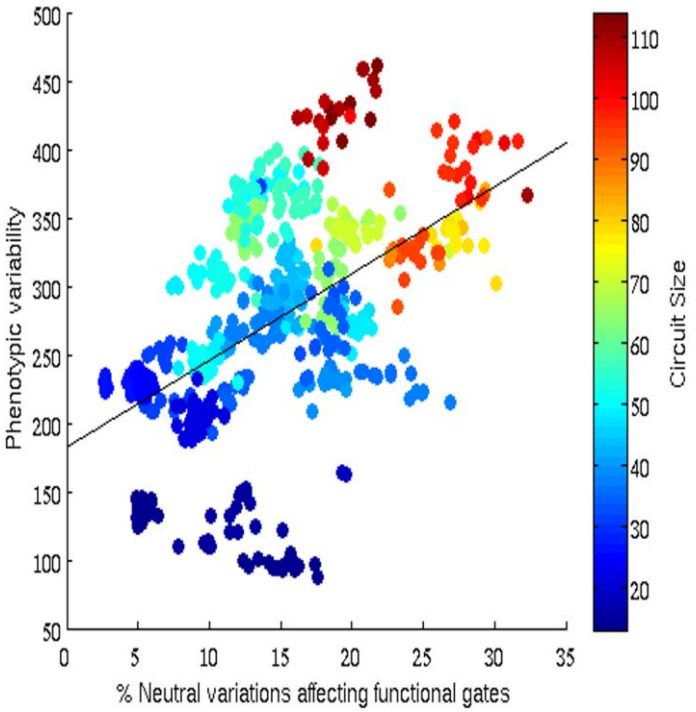


Figure 4.7. Distribution of phenotypic variability and robustness in evolving circuits. The top and the bottom figures show the correlation between phenotypic variability and robustness to variations overall and to variations affecting the functional gates only, respectively. The color of each dot indicates the size of the corresponding circuit. Data obtained by analyzing the neighborhood of 600 evolved circuits obtained by running 30 evolutionary experiments for 6,000 generations. Phenotypic variability has been calculated by using the same procedure described in the caption of Figure 4.4.

4.2.4 Discussion

The data collected demonstrate that circuits subjected to fault are more evolvable than the circuits that are not subjected to faults, with respect to both direct and indirect measures of evolvability (see Figure 4.6 right and Table 4.5). As far as I know, this is the first time that this effect is demonstrated through artificial evolutionary experiments. Moreover, the analysis of my results indicates why the need to cope with faults promotes evolvability. Here I restrict the analysis to only a 4-even bit parity problem, in order to use simple circuit and have a reasonably accurate search space description. The analysis of the circuits evolved in the no-fault condition indicates that they tend to be very robust with respect to genetic variations. Indeed, the large majority (95%) of the genetic variations affecting evolved circuits are neutral, i.e. do not alter the fitness of the circuits. This is because mutations enhancing robustness with respect to genetic variation tends to be retained in further generations even if they are adaptively neutral (i.e. even if they do not cause any fitness improvement). This in turn is because the offspring of individuals that are more robust to genetic variations, with respect to the other individuals of the population, have a lower probability to generate offspring affected by maladaptive mutations.

The tendency to select individuals that are robust with respect to genetic variations have both negative and positive effects (Wagner, 2008). The negative effect arises because of the fact that the robustness with respect to genetic variations is negatively correlated with phenotypic variability, i.e. with the number of different unique phenotypes that can be generated by mutating an individual genotype. The positive effect arises because of the fact that the robustness with respect to genetic variations enables the evolving population to retain a higher number of (neutral) variations that in turn enable the population to explore a larger portion of the neutral networks on which the evolving individuals are currently located. As claimed by Wagner (2008), the net effect of these negative and positive effects can be positive. This can be explained by considering that the increase in phenotypic variability gained from the population thanks to its enhanced ability to retain genetic variations is greater than the loss of phenotypic variability occurring at the level of the single individuals. This in turn can be explained by considering that what matters from the point of view of innovating is the possibility to access to new phenotypes not accessed before and that the neighborhoods of individuals located in different areas of the same neural networks tend to include different unique

phenotypes (Wagner 2008, Raman and Wagner, 2011).

Unfortunately, however, the selection of individuals robust with respect to genetic variation drives the evolutionary process toward the synthesis of minimal solutions, i.e. toward circuits that operate based on a minimum number of functional logic, gates (see Figures 4.3 and 4.6). In other words, it drives the evolutionary search toward a specific subarea of the neutral network. This might reduce and/or eliminate the advantage that can be gained by exploring different areas of the neutral networks, i.e. the advantage deriving from the fact that different areas of the neutral networks tend to include different unique phenotypes in their neighborhoods. Moreover, due to the negative correlation between robustness to genetic variation and phenotypic variability (Figure 4.5 top, (Raman and Wagner 2008)), the tendency to maximize robustness by selecting minimal solutions drives the evolving population toward a sub-area of the neutral network that is characterized by a low phenotypic variability. This implies that the selection of individuals that are robust with respect to mutations might drive the evolutionary process toward low evolvable solutions and eventually to evolutionary stagnation. Overall, this implies that robustness to genetic variations achieved in this way, i.e. achieved by simply minimizing the number of genes encoding functional traits, does not promote but rather reduces evolvability. These minimal circuits are actually not truly robust with respect to genetic variations. They are not able to tolerate genetic variations through redundancy or degeneracy.

Instead, the need to cope with faults drives the evolutionary process toward the selection of circuits that are not only robust against operation faults (as shown in (Thompson et al. 1999, Keymeulen et al 2000, MAcia and Sole 2009)) but that are also genuinely robust against genetic variations (i.e. that are able to tolerate a greater number of variations affecting their functional components with respect to circuits evolved in the no-fault condition, see Table 4.5). The synthesis of circuits characterized by this form of robustness reduces the tendency to select minimal solutions (Figure 4.6) and promotes the selection of solutions characterized by higher phenotypic variability (Figure 4.7 and Table 4.5) and higher evolvability (Table 4.5). In other words, it enhances the probability to generate new better phenotypes because of genetic variation both from the perspective of the single individuals and from the perspective of the population that is no more constrained toward the specific sub-area of the neutral network containing small circuits.

4.3 The role of environmental variations in the evolution of adaptive agents

4.3.1 Introduction

In this section, I analyze whether the occurrence of environmental variations across generations promotes the evolution of robust solutions. To study the effect of environmental variations I use simulated agents embedded in an external environment.. The obtained results demonstrated that indeed agents evolved in environmental conditions that vary across generations outperform agents evolved in environments that remain stable. With the term performance, I mean the ability of individuals to solve their adaptive problem in all possible environmental conditions. Performance can be estimated based on the averaged fitness obtained by the individuals during multiple post-evaluation tests carried in randomly different environmental conditions. The precision with which the fitness and/or the performance is estimated depends on the number of evaluations. Indeed, the higher the number of evaluations is, the higher the precision of the estimation is. Moreover, my results indicate that the performance is maximized when the environment varies across generation at a moderate rate (i.e. when the environment does not vary every generation but only every N generations).

As I will see, the advantage of exposing evolving agents to environments that vary across generation at a moderate rate is due, at least in part, to the fact that this condition maximizes the retention of changes that alter the behavior of the agents which in turn facilitates the discovery of better solutions.

This study is related to evolutionary dynamic optimization (Jin and Branke 2005, Kashtan et al 2007, Cruz et al 2011, Nguyen et al 2012, O' Donnell et al. 2014, Janssen et al 2016) namely the study of how an optimization algorithm can solve an optimization problem in a given period [t^{begin}, t^{end}] in which the underlying fitness landscape change and in which the optimization algorithm reacts to such change by providing new optimal solutions (adapted from definition 1 included in (Nguyen et al 2012)). However, the objectives of robust and dynamics optimization differ since the former aims to develop solutions capable of operating as effectively as possible in new environmental conditions without further adaptation. The latter, instead, aims to develop solutions that are not necessarily effective after the environment changes but that can adapt to the new environmental conditions as readily as possible. Consequently, also methodological issues differ. For example, the formalization of a method for measuring the speed with which agents adapt to the new environmental conditions is crucial for studying dynamic optimization (Cruz et al. 2012, Nguyen et al. 2012) but is not relevant for the study of robust agents.

Moreover, this research is related to the evolution of plasticity, i.e. the ability of agents to react to internal or external environmental states with a change in form, state, movement, or rate of activity (West-Eberhard M. J. 2003). Indeed, the agents described in this section have the possibility to detect the current environmental conditions and to vary their behavior in a context-dependent manner. For an analysis on the role of plasticity in artificial evolving agents, see (Nolfi S. and Floreano D. 1999, Di Paolo 2002, Carvalho and Nolfi 2016). In this section, however, I focus on the impact that environmental variation (and its rate of change) has on the course of the evolutionary process.

4.3.2 Method

To investigate the evolution of robust solutions I evolved a population of neuro-controlled agents for their ability to balance the two poles attached on their top through passive joints in varying environmental conditions (see Figure 2.8). The characteristics of the environment that vary are the inclination and the friction of the plane and the initial state of the cart and the poles. Consequently, the evolving agents should display an ability to balance the poles irrespectively from the characteristics of the plane and from the initial state of the cart and the poles.

This problem constitutes an extended version of the non-markovian version of the double-pole balancing problem introduced by (Wieland, 1991) that became a commonly recognized benchmark for nonlinear control (Igel, 2003, Khan et al., 2013). In the standard version of the problem, the plane is always orthogonal and the friction between the plane and the cart is always null.

Agents are provided with a three-layer neural network with five sensory inputs, ten internal neurons with recurrent connections, and one motor neuron (Figure 2.8, right). The sensory neurons encode the position of the cart (x), the angular position of the two poles (θ_1 and θ_2), the inclination of the plane (α), and the friction coefficient of the plane (μ_c). The activation state of the motor neuron is normalized in the range [-10.0, 10.0] N and is used to set the force (F) applied to the cart along the x axis. This is the classical setup for the double pole problem (Mikkulainein and Gomez 1997).

The connection weights of the neural network, that determine the behavior of the agents, are encoded in artificial genotypes and evolved through a steady state evolutionary algorithm, a method widely used to evolve embodied agents (Nolfi et al., 2016; Pagliuca et al., 2018).

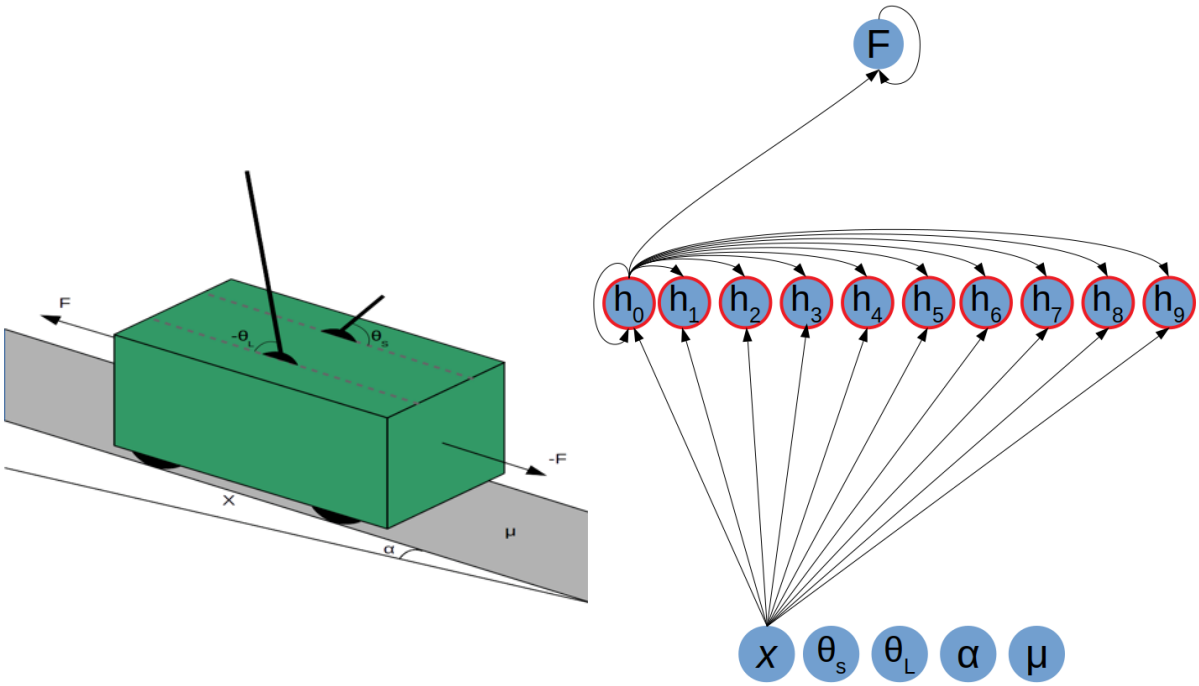


Figure 4.8. Left: Schematization of the extended double-pole balancing problem. See text for explanation. **Right**: The architecture of the neural network controller. The circles shown in the bottom, central and top part of the figure represents the sensory, internal and motor neurons, respectively. Red circles represent the biases. The arrows represent connections. For sake of clarity, only the connection departing from the first sensory and the first internal neurons are displayed.

4.3.2.1 The agent and the environment

The cart has a mass of 1Kg. The long pole and the short pole have a mass of 1.0 and 0.1 Kg and a length of 0.5 and 0.05 m, respectively. The cart can move along one dimension within a track of 4.8 m. It is provided with five sensors that encode the current position of the cart on the track (x), the current angle of the two poles (θ_L and θ_S) with respect to the cart, the angle of the inclined plane (α) and the friction coefficient (μ). The motor controls the force (F) applied to the cart along the x-axis. The goal of the agent is to move to maintain the angle of the poles and the position of the cart within a viable range (see below).

The behavior of the agent has been simulated based on equations 5-9. This is an extended version of the equations proposed by (Florian 2007) for the standard problem, developed by me, in which the inclination of the plane and the friction between the cart and the plane were not considered.

$$\ddot{x} = \frac{F + \mu_c \tilde{M} g + M_c g \sin\alpha + \sum_{i=1}^{n} \hat{F}_i}{M_c + \sum_{i=1}^{n} \tilde{m}_i} \qquad (5)$$

$$\ddot{\theta}_i = -\frac{3}{4l_i}\left(\ddot{x}\cos\theta_i - g\sin\theta_i + \frac{\mu_p\dot{\theta}_i}{m_i l_i}\right) \quad (6)$$

$$\hat{F}_i = \mu_c\left[\frac{3}{4}m_i g sin^2\theta_i - \frac{3\mu_p}{4l_i}\dot{\theta}_i sin\theta_i + m_i l_i\dot{\theta}_i^2 cos\theta_i\right] - \frac{3}{4}\left[m_i g sin\theta_i cos\theta_i + \mu_p\dot{\theta}_i cos\theta_i\right] + m_i l_i\dot{\theta}_i^2 sin\theta_i \quad (7)$$

$$\widetilde{m}_i = \frac{3}{4}[cos^2\theta_i - \mu_c cos\theta_i sin\theta_i] \quad (8)$$

$$\widetilde{M} = M_c cos\alpha + \sum_{i=1}^{n} m_i \quad (9)$$

where n is the number of poles on the cart, g is the acceleration due to gravity, m_i and l_i are the mass and the half length of the i^{th} pole, M_c is the mass of the cart, μ_c is the coefficient of friction of the cart on the track, μ_p is the coefficient of friction for the i^{th} hinge, F is the force applied to the cart, $\hat{F}_i$ is the effective force from the i^{th} pole on the cart, $\widetilde{m}_i$ is the effective mass of the i^{th} pole, and $\widetilde{M}$ is the effective mass of the cart.

4.3.2.2 The neural network controller of the agents

The controller of the agent is constituted by a neural network with five sensory neurons, ten internal neurons with recurrent connections, and one motor neuron. The sensory neurons are fully connected with the internal neurons, and the internal neurons are fully connected with the motor neurons and the internal neurons.

The sensory inputs gives the position of the cart (*x*) expressed in meters, the angular position of the

two poles (θ_1 and θ_2) radians, the inclination of the plane (α) radians, and the friction coefficient of the plane/cart (μ_c). The state of all sensors is normalized in the range [-0.5, 0.5]. The activation state of the motor neuron is normalized in the range [-10.0, 10.0] N and is used to set the force applied to the cart. The state of the sensors, the activation of the neural network, the force applied to the cart, and the position and velocity of the cart and of the poles are updated every 0.01s.

The neural network's architecture is fixed. The activation state of the internal and motor neurons is updated based on the logistic function. The connection weights and the biases of the network are encoded in agent's genome and evolved. More specifically each genome consists of a vector of 171 x 8 = 1368 bits that encode the 160 connection weights and the 11 biases of the corresponding neural network controller. The choice of using bits instead of floating point number is due to the robotic simulator used, FARSA that work with integers instead of floating poin numbers.

4.3.2.3 The evolutionary method

The evolutionary algorithm consists of a simple ($\mu + \mu$) evolutionary strategy (Rechenberg 1973), I used this algorithm because of its wide usage in the artificial life community. The algorithm operates based on populations formed by μ parents; during each generation, each parent generates one offspring, the parent and the offspring are evaluated, and the best μ individuals are selected as new parents. When the environmental conditions do not change with respect to the previous generation, the fitness of the parent is set equal to the fitness measured during previous evaluations and the evaluation of the parents is skipped to save time.

The genome of the initial population is composed by a matrix of (μ x 1368) bits that are initialized randomly. Each block of 8 bits is converted into a floating-point number in the range [-5.0, 5.0] that is used to set the weight of the corresponding connection or bias of the neural network controller. Offspring are generated by creating a copy of the genotype of the parent and by subjecting each bit to mutation with a *MutRate* probability. Mutations are realized by flipping the mutated bit.

The selection pressure is regulated by adding to the fitness of individuals a value randomly selected in the range [-Noise, Noise] with a uniform distribution (Jin and Branke 2005), where Noise corresponds to the theoretical maximum fitness multiplied by the value of the Stochasticity parameter. When Stochasticity is set to 0.0 only the best μ individuals are allowed to reproduce. The higher the level of stochasticity, the higher the probability that the worse individuals reproduce is.

This method requires to set two parameters: MutRate and Stochasticity. To identify the optimal values of the parameters I carried a series of control experiments in which the two parameters were

varied systematically (see the following Sections). The method operates well on small populations, e.g. populations formed by 100 individuals (Whitley, 2001).

Each agent is evaluated for *NT* trials,where *NT* is the actual number of the trials, that vary with respect to the characteristics of the plane and with respect to the initial state of the cart and the poles. More specifically, at the beginning of each trial the inclination of the plane (α), the friction coefficient between cart and plane (μ_c), the initial position of the cart on the plane (x), the velocity of the cart ($\dot{x}$), the angular position of the two poles (θ_1 and θ_2) and the angular velocity of the two poles ($\dot{\theta_1} \wedge \dot{\theta_2}$)are set to values selected randomly with a uniform distribution within the following ranges: [0.0, 0.2617], [0.0, 0.30], [-1.5, 1.5], [-1.2, 1.2], [-0.1047, 0.1047], [-0.1350, 0.1350], respectively.

Trials terminate after 1000 steps or when the angular position of one of the two poles exceeded the range [-0.628319,0.628319] rad and/or the position of the cart exceed the range [-2.4, 2.4] m. The fitness of the agent during a trial corresponds to the fraction of time steps in which the agent maintains the cart and the poles within the allowed position and orientation ranges. The total fitness is obtained by averaging the fitness obtained in the NT trials. The higher the number of NT is, the higher the accuracy of the estimation of the agent's performance in varying environmental conditions is.

The performance of evolved agents, i.e. the ability of an evolved agent to solve the problem in variable environmental conditions, is measured by post-evaluating them for 1000 trials in which the characteristics of the environment and the initial state of the cart are set randomly with a uniform distribution in the ranges described above. Although the number of different environmental conditions that an agent can encounter is practically infinite, measuring the performance of the agents on 1000 randomly different environmental conditions provides a good estimate of the performance of the agents, i.e. of the ability of the agents to solve the problem in all possible environmental conditions.

The experiments have been replicated in a fixed condition in which the environmental conditions do not vary across generations, in a varying condition in which the environmental conditions vary every generation, and in an intermediate condition in which the environmental conditions vary every N generations. The environmental conditions experienced by the agents are stored in an NTx8 matrix that encodes for each trial the inclination and the friction coefficient of the plane, the initial position and velocity of cart, and the initial position and velocity of the poles. The values of the matrix are generated randomly with a uniform distribution in the range described above. In the fixed condition, the *NT*x8 matrix is generated randomly once and is maintained constant during the

entire evolutionary process. In the varying condition, the NTx8 matrix is re-generated randomly every generation. Finally, in the intermediate condition, the matrix is re-generated randomly every N generations.

4.3.3 Results

Figure 2.9 displays the results of a series of experiments in which the environmental conditions remained fixed or varied every 5000, 2500, 1000, 500, 200, 100, 50, 25, 10, and 1 generation/s. In all experiments, the evolutionary process was continued for 50000 generations. Evolving agents were evaluated for 50 trials.

As can be seen, the best performance is achieved by agents evolved in environments that vary at a moderate rate across generations. Indeed, the agents evolved in the experiments in which the environment change every 100 generations achieve significantly better performance than the agents evolved in the experiments in which the environment remains fixed or change every generation (Kruskal-Wallis p-value <0.001, Wilcoxon test, p-value < 0.05 with Bonferroni correction in all cases, effect size: Cohen's d > 0.8 in all cases).

The agents evolved in environment that vary across generations outperform the agents evolved in non-varying environments. Indeed, the performance obtained in the fixed condition is significantly lower than the performance achieved in all other conditions (Kruskal-Wallis p-value <0.001, Wilcoxon test, p-value < 0.05 with Bonferroni correction in all cases, effect size: Cohen's d > 0.8 in all cases).

In the rest of the section, I will use the term intermediate condition to refer to the experiments in which the environment varies every 100 generations since it corresponds approximately to the average value of the best conditions.

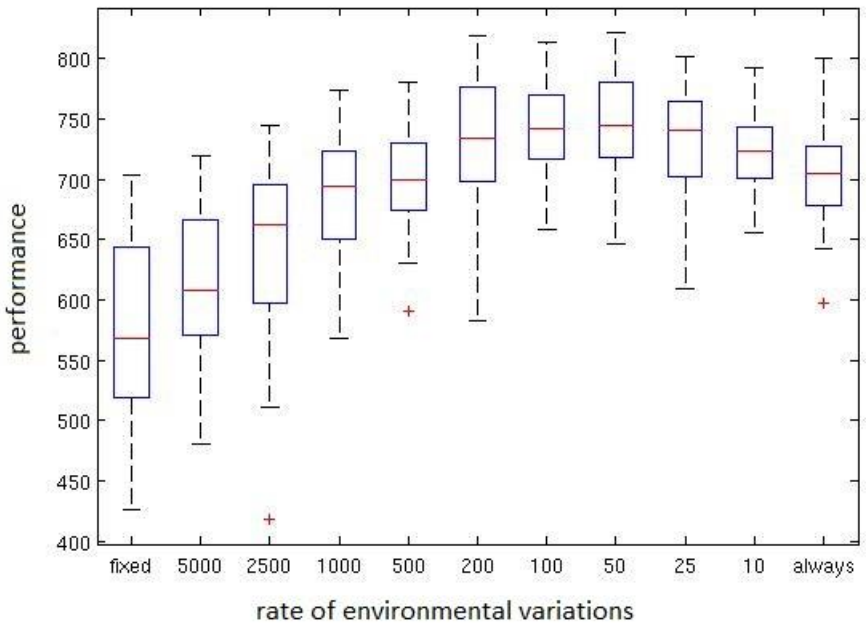


Figure 4.9. Performance in the fixed experimental condition, in eight intermediate conditions in which the environment varies every 5000, 2500, 1000, 500, 200, 100, 50, 25, and 10 generations (5000-10), and in the varying experimental condition in which the environmental conditions vary every generation. The number of trials (NT), the mutation rate, and the stochasticity parameters were set to 50, 1% and 0% in all cases. Boxes represent the inter-quartile range of the data and horizontal lines inside the boxes mark the median values. The whiskers extend to the most extreme data points within 1.5 times the inter-quartile range from the box. "+" indicate the outliers. The evolutionary process was continued for 50 million evaluations. Average results of 30 replications.

The beneficial effect of a moderate environmental variations across generation persists on the long term as demonstrated by the fact that the agents evolved in the intermediate condition outperform agents evolved in fixed and always varying conditions also after 100000 generations (Figure 4.10, Kruskal-Wallis p-value <0.001, Wilcoxon test, p-value < 0.05 with Bonferroni correction, effect size > 0.8).

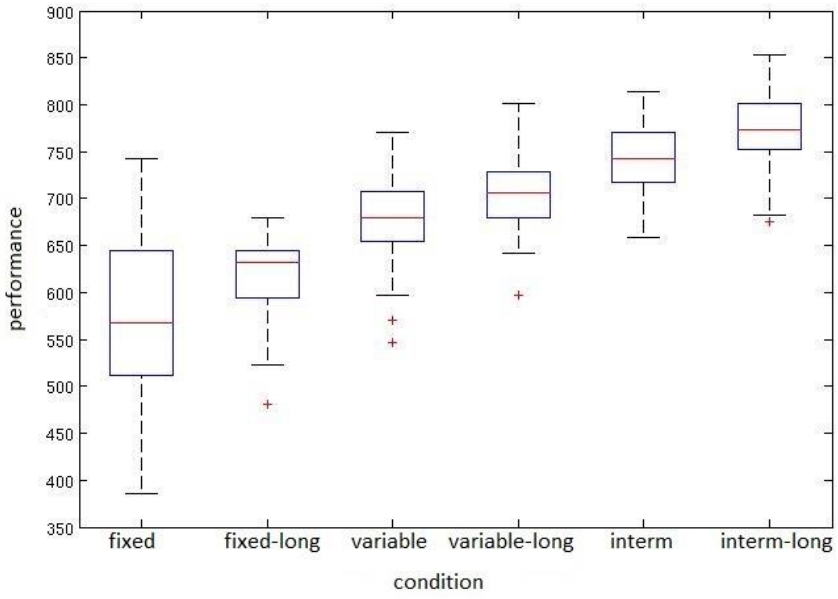


Figure 4.10. Performance of agents evolved in the fixed, variable and intermediate experimental conditions in which the environmental conditions remain stable, change every generation, and change every 100 generations, respectively. The boxes labelled "fixed", "variable" and "interm" display the performance obtained after 50000 generations (same data displayed in Figure 1.9). The boxes labelled "fixed-long", "variable-long" and "interm-long" display the performance obtained after 100000 generations. The number of trials, the mutation rate and the stochasticity parameters were set to 50, 1% and 0% in all cases. Boxes represent the inter-quartile range of the data and horizontal lines inside the boxes mark the median values. The whiskers extend to the most extreme data points within 1.5 times the inter-quartile range from the box. "+" indicate the outliers.

Tables 4.6, 4.7 and 4.8 report the results obtained by systematically varying the number of trials, the mutation rate and the stochasticity level in experiments carried out in the fixed, varying and intermediate experimental conditions in which the environment remains stable, varies every generation, and vary every 100 generations, respectively. The population size is always set to 100. The evolutionary process was continued for 50 million evaluations. Each number indicates the average results of 10 replications.

1 Trial	**Stoch 0%**	**Stoch 10%**	**Stoch 20%**	**Stoch 30%**	**Stoch 40%**
Mut 1%	78.9	85.6	95.3	86.3	75.3
Mut 2%	68.3	75.3	81.3	74.2	70.2
Mut 4%	55.3	63.2	53.3	56.8	55.2
50 Trials	**Stoch 0%**	**Stoch 10%**	**Stoch 20%**	**Stoch 30%**	**Stoch 40%**
Mut 1%	562.9	578.6	629.8	613.4	646.4

Mut 2%	606.9	628.4	625.7	645.2	619.6
Mut 4%	537.2	504.0	537.3	506.8	525.2
100 Trials	**Stoch 0%**	**Stoch 10%**	**Stoch 20%**	**Stoch 30%**	**Stoch 40%**
Mut 1%	625.6	595.9	622.0	615.9	634.1
Mut 2%	658.1	629.9	623.3	574.5	606.5
Mut 4%	479.2	442.6	506.1	477.0	463.0
150 Trials	**Stoch 0%**	**Stoch 10%**	**Stoch 20%**	**Stoch 30%**	**Stoch 40%**
Mut 1%	649.9	639.7	653.9	644.1	643.9
Mut 2%	601.6	605.3	624.2	629.8	602.9
Mut 4%	431.8	461.8	466.1	462.0	427.0
200 Trials	**Stoch 0%**	**Stoch 10%**	**Stoch 20%**	**Stoch 30%**	**Stoch 40%**
Mut 1%	641.6	644.2	634.0	**658.7**	636.8
Mut 2%	584.5	592.7	594.2	572.1	584.8
Mut 4%	424.5	442.7	444.2	432.1	427.8
300 Trials	**Stoch 0%**	**Stoch 10%**	**Stoch 20%**	**Stoch 30%**	**Stoch 40%**
Mut 1%	612.8	600.9	625.5	606.4	608.0
Mut 2%	571.6	546.9	551.9	568.7	539.5
Mut 4%	391.6	406.9	401.9	408.7	409.5

Table 4.6. Performance of the best agents evolved in the fixed environmental condition obtained by systematically varying the number of evaluation trials, the mutation rate, and the level of stochasticity. Each number indicates the average results of 10 replications. The evolutionary process was continued for 50 million evaluations. Data obtained by post-evaluating evolved agents for 1000 trials.

1 Trial	**Stoch 0%**	**Stoch 10%**	**Stoch 20%**	**Stoch 30%**	**Stoch 40%**
Mut 0.5	91.9	108.6	102.9	105.8	90.7
Mut 1%	119.9	130.2	118.9	111.8	103.7

Mut 2%	103.9	113.2	109.9	101.8	92.7
15 Trials	**Stoch 0%**	**Stoch 10%**	**Stoch 20%**	**Stoch 30%**	**Stoch 40%**
Mut 0.5	581.9	589.2	592.9	595.8	590.7
Mut 1%	669.9	670.2	678.9	681.8	673.7
Mut 2%	623.9	643.2	639.9	661.8	662.7
25 Trials	**Stoch 0%**	**Stoch 10%**	**Stoch 20%**	**Stoch 30%**	**Stoch 40%**
Mut 0.5	594.6	606.3	608.8	601.6	588.2
Mut 1%	696.9	707.6	709.5	700.5	698.5
Mut 2%	631.9	656.3	658.8	681.6	668.2
50 Trials	**Stoch 0%**	**Stoch 10%**	**Stoch 20%**	**Stoch 30%**	**Stoch 40%**
Mut 0.5	570.4	575.7	598.8	591.6	588.2
Mut 1%	665.6	675.6	698.9	697.8	675.7
Mut 2%	593.5	624.7	628.8	627.3	626.2
200 Trials	**Stoch 0%**	**Stoch 10%**	**Stoch 20%**	**Stoch 30%**	**Stoch 40%**
Mut 0.5	521.2	525.3	535.3	530.5	529.3
Mut 1%	604.2	615.2	617.2	613.5	611.6
Mut 2%	503.4	504.2	514.5	517.7	513.1

Table 4.7. Performance of the best agents evolved in the always-varying environmental condition obtained by systematically varying the number of evaluation trials, the mutation rate, and the level of stochasticity. Each number indicates the average results of 10 replications. The evolutionary process continued for 50 million evaluations. Data obtained by post-evaluating evolved agents for 1000 trials.

25 Trials	**Stoch 0%**	**Stoch 10%**	**Stoch 20%**	**Stoch 30%**	**Stoch 40%**
Mut 1%	760.2000	755.0000	731.3000	739.5000	725.5000
Mut 2%	714.4000	704.2000	680.0000	685.9000	673.9000
Mut 4%	582.4000	578.4000	571.0000	582.8000	581.5000
50 Trials	**Stoch 0%**	**Stoch 10%**	**Stoch 20%**	**Stoch 30%**	**Stoch 40%**
Mut 1%	745.2000	732.9000	741.6000	739.8000	726.8000
Mut 2%	696.3000	702.7000	694.8000	675.4000	668.3000
Mut 4%	546.7000	543.0000	547.0000	525.9000	527.7000
100 Trials	**Stoch 0%**	**Stoch 10%**	**Stoch 20%**	**Stoch 30%**	**Stoch 40%**
Mut 1%	712.5000	714.8000	718.9000	686.4000	682.4000
Mut 2%	636.0000	643.5000	624.4000	658.9000	650.8000
Mut 4%	520.1000	495.1000	495.1000	464.3000	458.1000
150 Trials	**Stoch 0%**	**Stoch 10%**	**Stoch 20%**	**Stoch 30%**	**Stoch 40%**
Mut 1%	716.4000	694.2000	687.6000	694.8000	691.5000
Mut 2%	627.0000	634.7000	602.6000	654.8000	652.5000
Mut 4%	480.5000	471.8000	435.8000	458.3000	455.5000
200 Trials	**Stoch 0%**	**Stoch 10%**	**Stoch 20%**	**Stoch 30%**	**Stoch 40%**
Mut 1%	667.4000	676.2000	678.6000	663.8000	652.5000
Mut 2%	601.0000	603.8000	598.4000	584.8000	572.4000
Mut 4%	460.5000	451.8000	425.8000	448.2000	435.6000

Table 4.8. Performance of the best agents evolved in the intermediate experimental condition obtained by systematically varying the number of evaluation trials, the mutation rate, and the level of stochasticity. The evolutionary process was continued for 50 million evaluations. Each number indicates the average results of 10 replications of each experiment. Data obtained by post-evaluating evolved agents for 1000 trials.

In the following two Sections I analyze why moderate environmental variation promote the evolution of better agents. I analyze whether moderate environmental variations are advantageous also from an evolutionary computational perspective, i.e. from the point of view of optimizing the

performance that can be achieved by using a limited computational budget.

4.3.3.1 On the role of fortunate specific environmental conditions

A possible hypothesis that could explain why agents evolved in varying environmental conditions achieve better performance than agents evolved in fixed conditions is that environmental variations enable evolving agents to encounter, sooner or later, favorable environmental conditions that promote the evolution of better solutions.

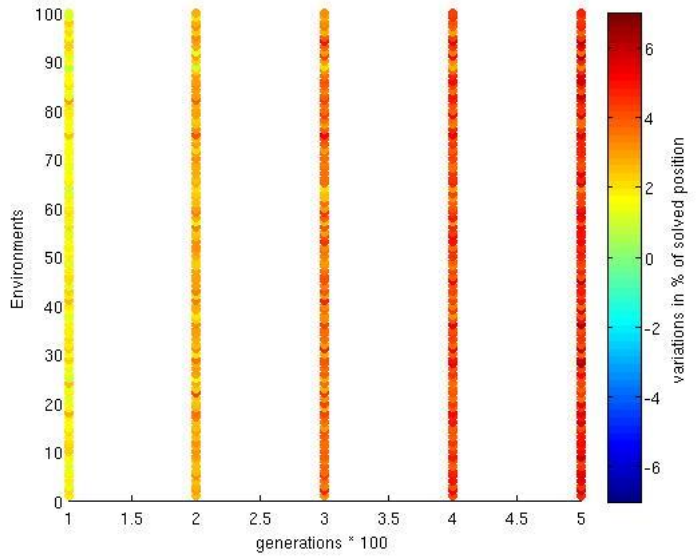
This hypothesis is based on evidences collected in incremental evolutionary experiments in which the problem and/or the environment become progressively more challenging throughout generations (Mikkulainen and Gomez 1997, Nolfi and Floreano 1999). The possibility to experience easier conditions during the first evolutionary phase leads to the synthesis of solutions for simple conditions that represent a stepping-stone for the synthesis of solutions for harder conditions. Moreover, the hypothesis is based on evidences collected in neural network learning literature that indicate that the relative distribution of qualitatively different stimuli in the training set affect the outcome of the learning process (Hare and Edelman 1995, Zhou and Liu 2006). Environmental variations in my experiments are stochastic and consequently cannot lead to a progressive complexification of the adaptive problem. On the other hand, one could hypothesize that the continuous variation of the environmental conditions can generate, eventually, sufficiently easy environmental conditions or favorable environmental conditions that can boost evolution.

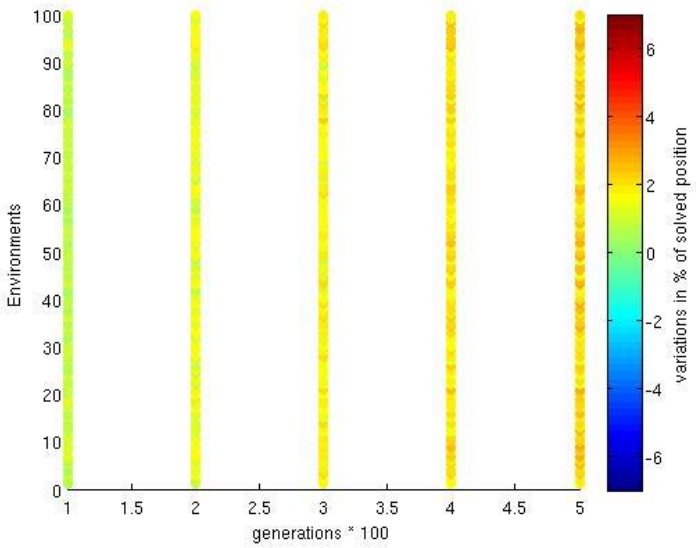
To verify this hypothesis, I analyzed the impact of different environmental conditions on the evolution of a population of evolving agents. The analysis was conducted on 30 populations of agents evolved in the intermediate experimental condition in which the environment varied every 100 generations after one, five, and then ten thousand generations. One hundred copies of these populations were evolved for 500 generations in 100 different corresponding environments for 50 trials. The performances of these populations were post-evaluated every 100 generations on 729 trials in which agents were exposed to systematically varied environmental conditions. Agents were evaluated for 3^6 = 729 trials during which the state of the six variables that encode the characteristics of the plane and the most important initial characteristics of the cart assumed one of the following values: α [-0.1385, 0.0, 0.1385], μ_c [-0.15, 0.0, 0.15], x [-0.75, 0.0, 0.75], $\dot{x}$ [-0.6, 0.0, 0,6], θ_1 [-0.05235, 0.0, 0.05235], $\dot{\theta_1}$ [-0.0675, 0.0, 0.0675]. The angular position and velocity of the shorter pole ($\theta_2 \wedge \dot{\theta}_2$), that is easier to control, were always set to 0.0.

As can be seen, the performance of the agents after one and five thousand generation's increases by a similar amount in all environmental conditions (Figure 4.11 top and middle). Later on, i.e. after

ten thousand generations, performance variations become much smaller and lead to both increase and decrease of performance (Figure 4.11 bottom, notice that the scale used for displaying variation in performance is one order of magnitude smaller in the case of the bottom picture).

Overall, this data does not show evidences of fortunate specific environmental conditions capable of boosting the evolutionary process. During the initial evolutionary phase, the evolving agents improve their ability of a similar amount in the large majority of environmental conditions. Later on, the evolving agents improve their ability by a similar small amount in the majority of the environmental conditions and worse their performance by a similar small amount in the remaining conditions.





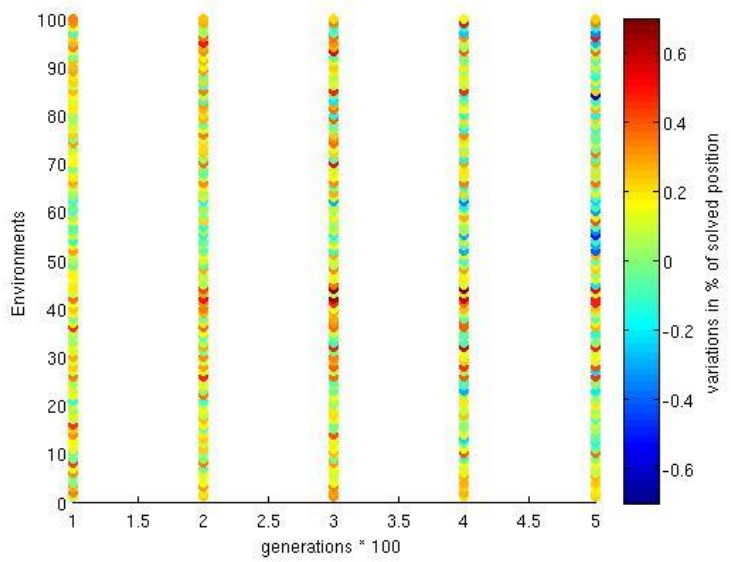


Figure 4.11. Fraction of additional trials solved every 100 generations by populations of agents evolved for 500 generations in 100 different environments. The top, middle, and bottom pictures show the results obtained by analyzing the agents evolved in the intermediate condition after one, five, and then ten thousand generations, respectively. Data averaged over 10 replications of the experiment. Columns correspond to data after 100, 200, 300, 400, and 500 generations. Lines indicate performance variations observed in each of the 100 different environments.

4.3.3.2 Environmental variation increases the rate at which evolving agents change across generations

Another possible reason that could explain why moderate environmental variation promotes the evolution of better solutions is that moderate environmental variations promotes evolutionary change.

The rationale behind this hypothesis is that adaption depends on the generation of phenotypic changes and on the retention of the changes that are adaptive. As stressed by (West-Eberhard 2003), phenotypic variations arise not only because of genetic variations but also because of environmental variations. Indeed, adaptations can arise both because of genetically induced changes and because of environmentally induced changes since both are subjected to genetic accommodation (West-Eberhard 2003). The sum of environmentally induced changes and genetically induced changes, therefore, produce more variation than genetically induced changes only and can thus facilitate the discovery of better solutions. The fact that moderate environmental variation promotes the evolution of better solutions than frequent environmental variation is because it represents an optimal tradeoff between the contrastive needs of variation and stability. Indeed, too infrequent environmental variations provide a limited opportunity for change. On the other hand, too frequent environmental variations prevent the stability that is necessary to enable genetic accommodation. This hypothesis is in line with West-Eberhard's claim that the most

important reason that explains why environmentally induced changes are evolutionarily important is indeed their time persistence:

Perhaps the most compelling argument for the superiority of environmental induction over mutations in term of recurrence and persistence has to do with the inexorable persistence of an environment immune to natural selection: environmental inducers might be not only immediately widespread without necessity for positive effects on fitness sufficient to spread them due to differential reproduction of their bearers (selection), but they are inexorably present.

West-Eberhard (2003), pp. 504

This hypothesis suggests that agents evolved in environmental conditions that vary at a moderate rate can accumulate more phenotypic variations across generations than agents evolved in non-varying environments and agents evolved in environments that vary every generation.

To verify how the rate of variation of the environment affects the rate at which the agents vary across generations, at behavioral level, I compared the best-evolved agent and its ancestor of 100 generations every 100 generations. The comparison was made by post-evaluating the agents for 729 trials during which they were exposed to systematically varied environmental conditions as described above and by counting the number of trials in which the agent manages to balance the poles while its ancestor fails or vice versa. The evolutionary process was continued for 5000 generations in all cases. As in the case of the analysis reported in Figure 4.9, NT was set to 50, the mutation rate wat set to 1%, and the stochasticity level to 0%.

As expected, the rate of variation decreases across generations because of the evolution of better and better agents (Figure 4.12). Interestingly, the behavior of agents evolved in the intermediate experimental condition, in which the environment varies every 100 generations (blue line), varies more across generations than the behavior of agents evolved in the non-varying environment (black line) and in the always-varying environment (red line) (Figure 4.12, Wilcoxon test, p-value <0.05 with Bonferroni correction, effect size > 0.8). The rate of variation of the behavior of the agents evolved in the fixed and always-varying experimental conditions, instead, does not differ statistically (Figure 4.12, Wilcoxon test, p-value > 0.5 with Bonferroni correction).

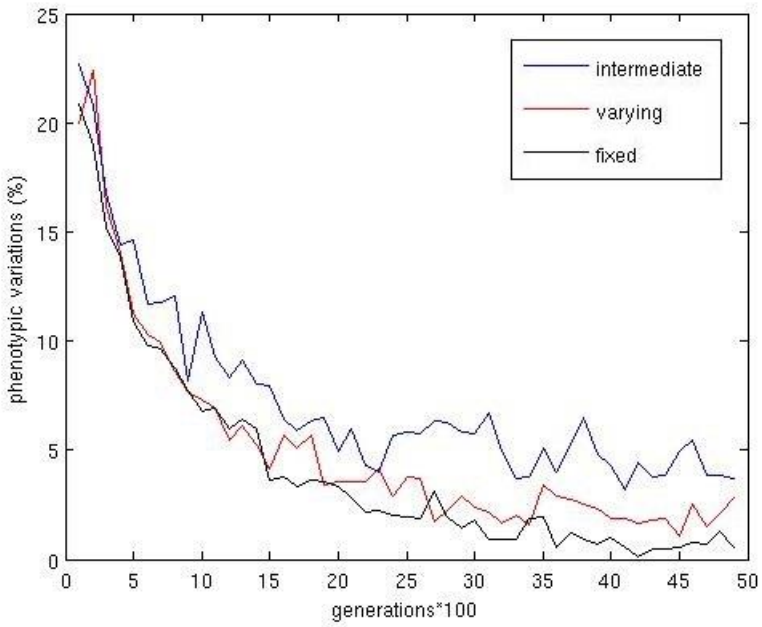


Figure 4.12. Fraction of trials in which the ability to solve the double-pole navigation problem varies every 100 generations. The black, blue, and red curves display the data obtained in the experiments carried out in the fixed, intermediate (in which the environment changes every 100 generations), and varying experimental conditions, respectively. Data obtained by analyzing the evolutionary lineage of the fittest evolved individual of each experiment. Each curve displays the average results of 25 replications continued for 5000 generations.

To verify how the rate of variation of the environment affects the rate at which the agents vary across generations, at the genetic level, I compared the fraction of genes that differ between the best evolved agent and its ancestor of 500 generations before every 500 generations. In addition, in this case the analysis was performed on experiments continued for 5000 generations. The agents evolved in the intermediate condition accumulate more genetic variations across generations than the agents evolved in the always-varying and fixed experimental conditions (Figure 4.13, Wilcoxon test, p-value < 0.05 with Bonferroni correction in both cases , effect size = 0.816). The agents evolved in the always-varying condition accumulate more changes than the agents evolved in the fixed condition (Figure 4.13. Wilcoxon test, p-value < 0.05 with Bonferroni correction, effect size = 0.744). Clearly, the rate at which agents change throughout generations at the genetic level is influenced by the mutation rate which, however, is always 1% in the experiments reported in this Section.

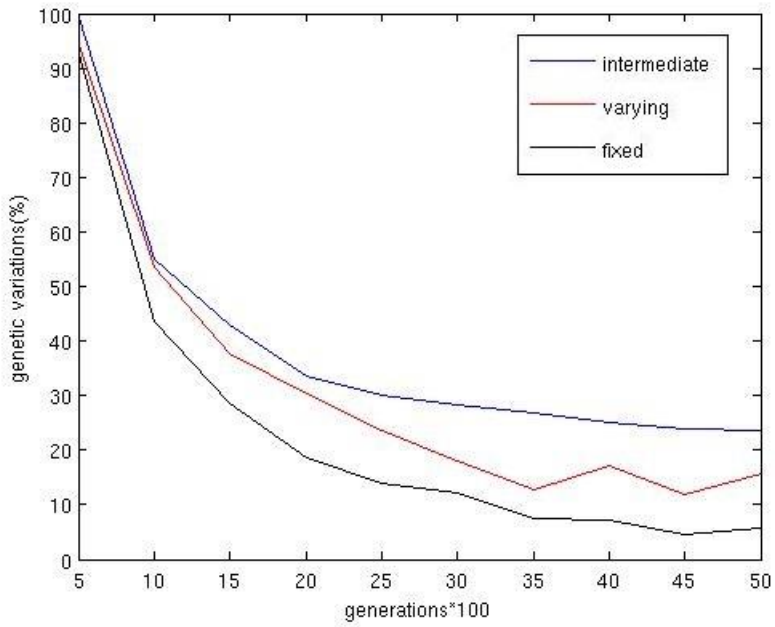


Figure 4.13. Fraction of genes that varies with respect to the 500th ancestor measured every 500 generations. The black, blue, and red curves display the data obtained in the experiments carried out in the fixed, intermediate, and always-varying experimental conditions, in which the environmental conditions do not vary, vary every 100 generations, and vary every generation, respectively. Data obtained by analyzing the evolutionary lineage of the fittest evolved individual of each experiment. Each curve displays the average results of 25 replications of each experiment.

The analysis of these data also indicates a significant correlation between the performance of evolving agents and the rate with which agents vary throughout generations at the behavioral (Spearman r= 0.52 p-value $<10^{-7}$) and genetic level (Spearman r= 0.53 p-value $<10^{-8}$).

Overall, these results indicate that moderate environmental variation promotes the retention of genetic variations that alter the behavior of the evolving agents, which, in turn, favors the discovery of better solutions.

4.3.3.3 On the advantage of moderate environmental variations from an evolutionary computation perspective

In this Section, I analyze whether moderate environmental variations are advantageous also from an evolutionary computation perspective, i.e. from the point of view of maximizing the performance that can be achieved with a limited computational budget.

Since the major computation, cost in these experiments is constituted by the evaluation of candidate solutions, in this Section I compare experiments in which the evolutionary process is continued until a maximum total number of evaluations is performed.

To identify the best value of the parameters, I run three series of experiments in the fixed, always-varying and intermediate experimental conditions in which I systematically varied the number of

trials experienced by evolving agents (NT), the mutation rate, and the stochasticity level. The total number of evaluations was set to 50 million in all cases. The analysis of this data indicates that the best performance is achieved by setting NT to 200, 25 and 25 in the fixed, always variable and intermediate conditions. Performance with the best parameter differ statistically in all cases: fixed condition (Table 4.6, Kruskal-Wallis p-value <0.001, Wilcoxon test, p-value < 0.05 with Bonferroni correction, effect size > 0.7); always-variable condition (Table 4.7, Kruskal-Wallis p-value <0.001, Wilcoxon test, p-value < 0.05 with Bonferroni correction, effect size > 0.7); intermediate condition (Table 4.8, Kruskal-Wallis p-value <0.001, Wilcoxon test, p-value < 0.05 with Bonferroni correction; effect size > 0.7).Interestingly, therefore, the value of NT that leads to the best performance is relatively small in the varying and intermediate experimental conditions. This implies that synthesizing solutions capable of operating effectively in widely varied environmental conditions does not necessarily cause an explosion of the computational cost required evaluating candidate solutions.

The best values for the mutation rate is 1% in all conditions (Table 2.6-2.8). The best value of the stochasticity parameter is = 30%, 0%, and 20% in the fixed, intermediate, and variable experimental conditions, respectively (Table2.6-2.8).

The comparison of the results obtained in the three conditions by using the best parameters for each condition shows that the agents evolved in the moderate condition outperform the agents evolved in the other two conditions both after 50 and 100 million evaluations (Figure 2.14, Kruskal-Wallis p-value <0.001, Wilcoxon test, p-value < 0.05 with Bonferroni correction, effect size > 0.7).

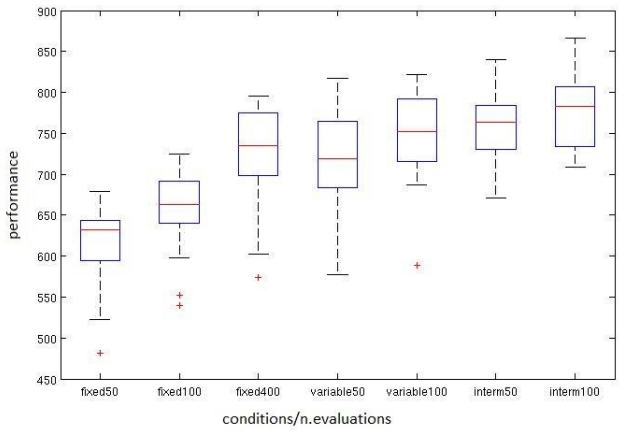


Figure 4.14. Performance of agents evolved in the fixed, variable and intermediate experimental conditions in which the environmental conditions remain stable, change every generation, and change every 100 generations, respectively. Data obtained after 50 and 100 million evaluations (400 million evaluations in the case of fixed400). Results obtained with the parameters that produced the best results in each condition: NT= 200 in the fixed condition and NT=25 in the variable and intermediate conditions; mutation rate = 1% in all conditions; stochasticity = 30%, 0%, and 20% in the fixed, intermediate, and variable experimental conditions, respectively. Boxes represent the inter-quartile range of the data

and horizontal lines inside the boxes mark the median values. The whiskers extend to the most extreme data points within 1.5 times the inter-quartile range from the box. "+" indicate the outliers.

The agents evolved in the moderate environmental condition outperform the agents evolved in the fixed environmental conditions independently from the duration of the evolutionary process. Indeed, the agents evolved in the intermediate condition with the best parameters for 50 million evaluations outperform the agents evolved in the fixed environmental condition with the best parameters for 400 million evaluations (Figure 2.14 right, Kruskal-Wallis p-value <0.001, Wilcoxon test, p-value < 0.05 with Bonferroni correction, effect size = 0.850).

4.3.4 Discussion

Previous researches reviewed in the introduction section demonstrated how robust solutions can be obtained by evaluating evolving agents multiple times in variable environmental conditions. The results demonstrate how the occurrence of variations in the environmental conditions across generations leads to better solutions. Moreover, the results demonstrate that performance are maximized when the environment vary at a moderate rate across generations, i.e. when the environment does not vary every generation but every N generations.

These results have been obtained on a standard problem commonly used as a benchmark in artificial evolution. In principle, I can expect that these results will generalize to all problems domains that require solutions robust to environmental variations. The generality of the effect, however, should be verified in future studies conducted on different problem domains.

Lineages of agents evolved in environments that vary across generations change more, at the behavioral level, than lineage of agents evolved in non-varying environment. Moreover, lineages of agents evolved in environments that vary across generations at a moderate rate change more, at the behavioral level, than lineage of agents evolved in non-varying or always-varying environments. Consequently, the advantage provided by moderate environmental variation is due, at least in part, to the fact that it promotes evolutionary change. The fact that the amount of evolutionary change is maximized at an intermediate rate of environmental variation is because it represents an optimal tradeoff between the contrastive needs of variation and stability. These observations confirm the importance of environmentally induced changes in natural evolution stressed by (West-Eberhard 2003). Moreover, these observations confirm the hypothesis that a key reason that explain the importance of environmentally induced change is their persistence over time (West-Eberhard 2003). There are no evidence about the fact that some particular favorable environment can improve the generalization performance, but the environmental variation itself promotes the evolution of more general controller, they are not insensitive to particular environment but

generalization is emergent from evolution in varyng condition not only from favorable environment. Finally, I demonstrate that moderate environmental variations are advantageous also from an evolutionary computation perspective, i.e. from the perspective of maximizing the performance that can be achieved within a limited computational budget. Indeed, the experiments carried out by systematically varying the number of trials during which evolving agents are evaluated (NT) and the rate at which the environment change across generations while maintaining the total number of evaluations constant indicate that performance is maximized when the environment vary at a moderate rate and the evolving agents are evaluated for a relatively small number of trials. Experiencing variable environmental conditions across generations permits to select solutions that are robust with respect to environmental variations while minimizing the computational cost required to evaluate each candidate solution and maximizing the number of generations.

Chapter 5. On the relation between phenotypic complexity and evolvability

5.1 Introduction

In this chapter I study the relation between phenotypic complexity and evolvability.

In Section 5.2 I show how $(1 + \lambda)$ evolutionary strategies largely outperform $(\mu + \mu)$ evolutionary strategies in the context of the evolution of digital circuits --- a domain characterized by a high level of neutrality. Moreover, I demonstrate how the inferiority of the latter method is explained by the fact that the competition among the offspring of multiple parents lead to the selection of phenotypically simple but low evolvable circuits. These circuits achieve robustness by minimizing the number of functional genes rather than by relying on redundancy or degeneracy to buffer the effects of mutations. Finally, in this section I introduce a new algorithm, named Parallel Stochastic Hill Climber (PSHC), which outperforms the other two methods considered.

In Section 5.3 I demonstrate how the efficiency of Cartesian Genetic Programming methods can be improved by preferentially selecting phenotypically larger solutions among equally good candidates. This hypothesis is demonstrated on different problems: the eight-bit parity problems, and the Paige regression problem. In all cases, the preferential selection of larger solutions provides an advantage in term of the performance of the evolved solutions and in term of number of evaluations required to evolve optimal or high-quality solutions. I show how the advantage provided by the preferential selection of larger solutions can be further extended by self-adapting the mutation rate through the one-fifth success rule. Finally, I show how for problems like the Paige regression, in which neutrality plays a smaller role, the advantage of the preferential selection of larger solutions can be obtained by preferring larger solutions also among quasi-neutral alternative candidate solutions, i.e. solutions achieving slightly different performance.

5.2 The impact of the evolutionary algorithm on the size of the evolving solution

5.2.1 Introduction

As I state in the introduction, the relationship between robustness and evolvability can be influenced also by the way in which robustness is achieved. Namely by whether robustness is achieved through the utilization of phenotypically simple solutions that minimize the number of

genes playing a functional role (de Visser et. al. 2003) or through phenotypically more complex solutions capable of buffering the effect of mutations through redundancy (i.e. through the utilization of multiple redundant components playing the same function) or degeneracy (i.e. through the utilization of multiple components playing multiple functions, see Tononi, Sporns, and Edelman, 1999; Edelman and Gally, 2001). And the way in which robustness is achieved can be influenced by the characteristics of the evolutionary process, in particular, by the level of competition among evolving individuals.

To investigate this hypothesis, I measured robustness and evolvability in digital circuits evolved with different evolutionary methods.Moreover, I chose this domain since it permits to define in an operational and measurable way the key properties that I want to analyze, namely robustness, phenotypic complexity, evolvability, and phenotypic variability (i.e. the propensity of a system to produce adaptive heritable phenotypic variations because of mutations, independently of whether or not variations are adaptive). A more detailed definition of these terms and the description of the way in which these properties are measured in the context of digital circuits are reported in Section 5.2.

The obtained results indicate that the level of competition between individuals subjected to differential reproduction influences the phenotypic complexity and the evolvability of the individuals. The strength of the effect is remarkable as demonstrated by the fact that circuits evolved with the $(1 + \lambda)$ evolutionary strategy (Rechenberg, 1973) solve the problem relatively quickly in all replications while circuits evolved with the $(\mu + \mu)$ evolutionary strategy solve the problem in only 57% of the replications. This difference is because the competition for robustness to mutations among the circuits evolved with the (μ+ μ) evolutionary strategy leads to the selection of phenotypically simple but low evolvable circuits. These circuits achieve robustness by minimizing the number of functional genes rather than by relying on redundancy or degeneracy to buffer the effects of mutations. In the $(1 + \lambda)$ evolutionary strategy there is the same tendency to have many unexpressed genes but it appears rather small as showed in the results section. The analysis of the evolutionary dynamics also provided the basis for the development of a new original algorithm, called parallel stochastic hill climber (PSHC), which achieves better results. Then, using more complex task, I show how significantly improve the performances modifying the $(1 + \lambda)$ ES to preferentially chose larger solutions among alternative solutions having the same fitness. The results collected on the eight-bit parity problem and on a continuous regression problem indicate that the new algorithm largely outperforms the standard algorithm in term of speed and quality of the evolved solutions. I show that the advantage of the preferential selection of larger solutions can

be extended by self-adapting the mutation rate through the one-fifth success rule. Finally, I show that for problems in which neutrality plays a minor role, the advantage of the preferential selection of larger solutions can be extended by preferring larger solutions among quasi-neutral alternative candidate solutions, i.e. also among solutions achieving slightly lower performance

5.2.2 Method

In the section, I describe the evolutionary algorithms and the measures used to analyze the evolutionary dynamics.

5.2.2.1 The digital circuits and fitness function

I used digital circuits with four inputs, 400 logic gates divided into 20 layers of 20 gates (a configuration enough complex to allow to solve the task), and one output for the ability to compute a 6-bit even parity function (i.e. to produce as output 1 when there is an even numbers of 1 in the input pattern and 0 otherwise).

In the first series of experiments, programs were evolved for the ability to solve a 6 bit even parity problem and were provided with 6 inputs and 1 output. Programs were evaluated on 2^6patterns, i.e. on all possible input patterns. Then to improve the difficulty and test my hypothesis on more challenging task the even bit parity problem will take 8 bit as input and the evolved programs will be evaluated on all the 2^8 possible input patterns.

Circuits are evaluated for the ability to map the 2^n possible input patterns into the corresponding desired outputs (i.e. 1 for input patterns with an even number of 1 and 0 otherwise). The fitness of the circuits is calculated based on the following equation:

$$F = 1 - \frac{1}{2^n} \sum_{j=1}^{2^n} |O_j - E_j| \qquad (10)$$

Where *n* is the number of inputs of the circuit, *j* is the number of the input pattern varying in the range [1, 2^n], O_j is the output of the circuit for pattern *j*, and E_j is the desired output for pattern *j*.

5.2.2.2 Evolutionary algorithms

To investigate the relation between the characteristics of the algorithm and the evolvability of the circuits, I compared three different algorithms.

The first is a $(1 + \lambda)$ evolutionary strategy (ES, Rechenberg, 1973; Beyer and Schwefel, 2002) that operates on the basis of a single parent, produces λ offspring, and selects the best between the parent and the offspring as a new parent (see the pseudo-code below). I choose this algorithm since it is commonly used for the evolution of digital circuits (Miller, Job and Vassiley, 2000) and, more generally, for the evolution of graph structures (Miller and Thomson, 2000; Miller, 2011).

To regulate the effect of the selection pressure, I added to the fitness of the evolving individuals a value randomly selected in the range [-*Stochasticity*, *Stochasticity*] with a uniform distribution (see Jin and Branke, 2005). I decided to use this technique in combination with a selection operator that always selects the best between parents and offspring, rather than probabilistic selection operators such as roulette wheels or tournament selection, since: (i) it allows to regulate the selective pressure quantitatively by varying a single parameter (see Back, 1994), (ii) it permits to regulate the selective pressure from the maximum value, in which only the best candidate solutions between parents and offspring are selected, to a minimal value, in which the differential reproductive probability of better and worse individuals is minimal (see Back, 1994), (iii) it is qualitatively similar to the stochastic variation of fitness caused by uncontrolled variations occurring in uncertain environmental conditions. An example of uncertain conditions is constituted by the experiments involving robots, in which the behavior displayed by the agent and, consequently, the fitness measure tend to vary during repeated evaluations even in apparently identical conditions as a consequence of small differences in the initial position and orientation of the robot, lighting conditions, etc. Previous demonstrations of the fact that the introduction of noise in the fitness measure promotes the evolution of better solutions are reported in (Bäck and Hammel, 1994; Levitan and Kauffman, 1994; Rana, Whitlev, and Cogswell, 1996).

Algorithm 1.$(1 + \lambda)$ ES

1. Generate randomly the starting genotype (G_0)
2. Evalute G_0 and get its fitness (F(G_0))
3. Generate λ mutated offspring of G_0, (M_λ) and evaluate them F(M_λ)
4. Sort G_0 and M_λ by fitness, the best is the new G_0
5. Until the max evaluations number is not reached go to point 3

The second algorithm is a $(\mu + \mu)$ ES that operates with μ parents, enables each parent to produce a single offspring, and select the μ best individuals between the parents and the offspring as new parents (see the pseudo-code below). This type of algorithm is widely used for the evolution of neural networks (e.g. Pagliuca, Milano and Nolfi, 2018). In addition, in this case I use the *Stochasticity* parameter to regulate the selective pressure, as showed in the previous chapter this kind of algorithm tends to converge over robust population but low evolvable, stochasticity prevents this type of behavior.

Notice that in this method the offspring of individuals that are more robust to mutations have more chances to be selected than the offspring of individuals that are less robust. To illustrate this point, consider the case in which two parents have the same fitness and in which the former parent is more robust to mutations than the second parent is. Moreover, imagine that the mutations received by offspring are neutral or counter-adaptive (none of the mutations produces an improvement of the fitness). The offspring of the former parent have a greater probability to preserve the fitness of the parent than the offspring of the latter parent. Consequently, the offspring of the parent that is more robust has a greater probability to be selected. Overall, this implies that the competition between the evolving individuals is regulated primarily by their fitness and secondly (in the absence of adaptive mutations) by their robustness.

The competition between the individuals evolved with the $(1 + \lambda)\,\mathrm{method}$, instead, is not influenced by robustness to mutations. This is because offspring originate from the same parent. Offspring might differ among themselves in terms of robustness because of the mutations that they received. However, mutations that alter the level of robustness without altering the fitness do not provide an adaptive advantage in the $(1 + \lambda)\,\mathrm{method}$ since two offspring with the same fitness have exactly the same probability to be selected, irrespectively of whether the former is more robust of the latter or vice versa.

Algorithm 2.$(\mu + \mu)$ ES

1. Generate random population of 20 individuals (Parents P_i)
2. Evaluate Parents and retrieve fitness($F(P_i)$)
3. For each member of the population create an offspring by mutation(M_i) and evaluate them ($F(M_i)$)
4. Order Parents and Offrsping by fitness, the best 20 individuals are the new parents
5. Until generation limit reached go to step 2

The third algorithm designed is called Parallel Stochastic Hill Climber (PSHC) consists of a combination of a $(\mu + \mu)$ and a (1 + 1) ES (see pseudo-code below). In this algorithm each parent is adapted through an (1 + 1) ES for a certain number of *Variations* during which the parent or a varied version of the parent generates a single mutated candidate solution that is discarded or used to replace the previous candidate solution depending on whether or not it is outperformed by the original solution. The best candidate solution obtained during this variation phase is then used to replace its parent and, with a certain low probability (*Interbreeding*), the worst individual of the population. The combined usage of the two types of evolutionary strategies allows the combination of the advantages of operating on a population of parents with the advantages that can be gained by reducing the level of competition between the members of the population.

Also in the case of this algorithm I regulate the selection pressure during the variation phase by adding to the fitness of the candidate solutions a value randomly selected in the range [-*Stochasticity*, *Stochasticity*] with a uniform distribution. However, the selection of the best candidate solution obtained during the variation phase is based on the actual fitness (i.e. the fitness without noise). The rationale behind this choice is that it enables to select sub-optimal solutions temporarily, i.e. during the (1 + 1) variation phase, but not during the $(\mu + \mu)$ selection phase. Selecting sub-optimal solutions reduces the risk to remain trapped in local minima but causes a reduction of performance that can be temporal or permanent. Retaining sub-optimal solutions temporarily only during the variation phase enables to reduce the risk to remain trapped in local minima while eliminating the risk to retain variation that produce a permanent reduction of performance.

Algorithm 3. Parallel Stochastic Hill Climber (PSHC)

1. Generate random population of 20 individuals (Parents P_i)
2. Evaluate all the parents and get them fitness F(P_i)
3. For each parent generate a mutated offspring (M_i) and evaluate them F(M_i)
4. For n times do the (1+1) annealing phase:
5. Generate 1 mutated copy of each M_i, (G_i) and evaluate them F(G_i)
6. If F(G_i) ≥ F(M_i), G_i is the new M_i
7. Update the parents populations replacing each P_i with its the best M_i
8. Do an interbreeding phase where the worst parent can be replaced by one of the best
9. Until max evaluations number is reached go to point 3

The PSHC is a form of island evolutionary algorithm (Whitley, Rana, and Heckendorn, 1998) in which the population is divided into a number of sub-populations and the individuals of each population are allowed to compete with the individuals of the other sub-populations only occasionally. In the case of the PSHC, however, the islands are constituted by single individuals that adapt based on an (1 + λ) evolutionary strategy instead of sub-populations in which individuals compete with the other members of the sub-population. This difference is important since, as pointed out above, the competition between evolving individuals influences the robustness and the evolvability of the agents. Indeed, experiments carried out by using a modified version of the algorithm in which the islands were formed by 2, 5, or 10 individuals led to significantly worse results (see below).

The evolutionary process is continued until a maximum number of total evaluations are performed, evaluations are counted for every time one circuit is activated, i.e. 1+1-ES only evaluates one genotype at each generations as the parent is the best from the previous generation. This number was set to $6 * 10^6$. The rationale behind this is that the evaluation of candidate solutions constitutes the major computational cost. Consequently, fixing the total number of candidate solution evaluations permits to maintain the computational cost of experiments carried with different algorithms approximately constant. Each experiment is replicated 30 times with randomly different initial populations of 20 individuals.

To verify that the differences in performance are not influenced by the parameters setting, I replicated the experiments by systematically varying the parameters, i.e. I varied the stochasticy and the mutation rate, and I compared the results achieved by different algorithms in the experiments performed with the optimal parameters.

3.2.2.3 Measures

Robustness to mutations, defined as the capability of circuits to preserve their functionality after mutations (Abdelhalim et al. 2011) is measured in two ways: (i) by calculating the fraction of offspring that have a fitness equal or greater than the fitness of the parent (fraction of fitness preserving offspring), and (ii) by calculating the fraction of offspring generated by performing a single random mutation that have a fitness equal or greater than the fitness of the parent. To estimate robustness with good precision, the measures are calculated over 10^4 variations of the parent's circuit. The first measure is influenced by the mutation rate that might vary in different experiments, while the second measure is independent of it.

Phenotypic complexity, defined as the complexity of the functional elements that constitute the system and the complexity of the way in which the elements are organized, is measured by counting the number of gates that contribute to generate the output of the circuit (functional gates). Functional gates correspond to the gate that constitute the output of the entire circuit and the gates that project connections to the output directly or indirectly, through other gates. Given that in the experimental setup genes encode the logic function performed by each gate and the way in which the gate is wired to the other gates, this measure also indicates the number of genes encoding functional properties of the system. The term phenotypic complexity does not have a unique definition (see for example see Adami, 2002; Carlson and Doyle, 2002; Crutchfield and Görnerup, 2006; Hazen, et al., 2007; Whitacre, 2010). Moreover, phenotypic complexity is hard to define formally and to measure in the case of systems composed by heterogeneous elements and/or displaying multi-level organizations. The possibility to use a relatively straightforward measure in the case of my experiments, therefore, is due to the utilization of simple feed-forward digital circuits composed by homogeneous elements.

Phenotypic variability, defined as the propensity of circuits to vary phenotypically because of mutations, is measured (following Raman and Wagner [2011]) by calculating the number of unique functions computed by circuits located in the genetic neighborhood of the original circuit. This number is estimated by performing 10 repetitions of a 1000 steps function-preserving random walk from the original circuit. The random walk is realized by: (i) generating a mutated circuit, (ii) incrementing a counter if the varied circuit computes a function that differs from the functions computed by the original circuit and by previous varied circuits, (iii) preserving or removing the mutation depending on whether or not the varied circuit has the same fitness of the original circuit, (iv) repeating the previous three operations for 1000 steps.

5.2.3 Results

In this section, I describe the results obtained and the analysis performed.

5.2.3.1 Digital circuits evolved with the $(\mu + \mu)$ ES

With the best combination of parameters (MutRate=0.02, Stochasticity=0.05, see Table 9), evolving circuits find optimal solutions in 56.66% of the replications.

	MutRate 0.01	MutRate 0.02	MutRate 0.03	MutRate 0.04
Stochasticity 0.00	46.66 (37.80)	43.33 (33.91)	33.33 (30.73)	40.00 (32.65)
Stochasticity 0.01	46.66 (39.11)	53.33 (35.83)	36.66 (34.01)	33.00 (33.41)
Stochasticity 0.02	30.00 (37.25)	46.66 (37.53)	43.33 (35.46)	30.00 (32.48)
Stochasticity 0.03	43.33 (39.89)	33.33 (38.25)	53.33 (38.40)	40.00 (35.62)
Stochasticity 0.04	30.00 (39.34)	50.00 (36.24)	30.00 (37.78)	40.00 (38.74)
Stochasticity 0.05	50.00 (41.81)	**56.66 (42.87)**	53.33 (38.59)	46.66 (37.95)
Stochasticity 0.06	43.33 (40.91)	43.33 (36.69)	50.00 (33.61)	33.33 (33.11)
Stochasticity 0.07	30.00 (35.94)	33.33 (34.34)	30.00 (38.63)	23.33 (37.23)

Table 5.1. Performance of circuits evolved with the $(\mu + \mu)$ ES in experiments carried out by using different mutation rate and stochasticity levels. μ was set to 20. The first number of each cell indicates the fraction of replications that achieved optimal performance. The numbers between parentheses indicate the average size of the evolved circuits.

The analysis of evolved circuits indicates that, overall, they are very robust with respect to mutations. Indeed, 35.12% of the offspring of the circuits evolved with the optimal parameters have a fitness equal or greater than their parent and 95.15% of the mutations are neutral (i.e., do not alter the fitness). This high robustness, however, is largely because the size of the functional part of the evolving circuits is small, i.e. only about 10% of the 400 gates actively contribute to the generation of the outputs of the circuits (see Table 5.1). Indeed, by restricting the analysis to the mutations that affect the functional gates, the percentage of neutral mutations drops from 95.15% to 0.52%.

As shown in Figure 5.1, the phenotypical complexity of the circuits, defined as the number of gates that contribute to generate the outputs of the circuits, increases during the initial phase of the evolutionary process (during which the fitness of the evolving circuits increases) and decreases during the successive phase (in which the fitness of the evolving circuits remains stable). This is the result of combined effect of the complexification of the individuals' phenotype driven by adaptive variations and the simplification of the individuals' phenotype driven by neutral variations.

The complexification originates because of the strong correlation between the fitness and the functional size of the circuits (Figure 5.2, Spearman Test, rho 0.6359, phi < 10^{-5}, n = 600). The selection of circuits fitter than their ancestors, in fact, leads to the selection of circuits that are

phenotypically more complex than their ancestors, on average.

The simplification of the individuals' phenotype, instead, is caused by two factors. First, the probability to generate offspring with less functional gates is significantly higher than the probability to generate offspring with more functional gates (Wilcoxon ranksum test, p-value $<10^{-9}$). Data obtained by analyzing the offspring of the 30 best circuits evolved in 30 replications performed with the $(\mu + \mu)$ ES algorithm. For the purpose of this analysis, each circuit was allowed to generate 10^4 offspring. Secondly, as stated above, the competition between reproducing individuals leads to the selection of individuals that are robust with respect to genetic variations. Given the strong negative correlation between the robustness and the phenotypic complexity of individuals (Figure 5.3, Spearman Test, rho -0.9017, phi < 10^{-5}, n = 600), in the majority of the cases the selection of robust individuals leads to the selection of phenotypically simpler individuals.

The fact that the phenotypic complexity increases during the initial phase of the evolutionary process in which the fitness of the evolving individuals also increases and then decreases during the successive phase is thus due to the fact that the complexification factor operates during the initial evolutionary phase only, while the simplification factor operates during the entire evolutionary process, especially during neutral evolutionary phases.

The presence of a strong positive correlation between the size and the phenotypic variability of the circuits (Figure 5.4, Spearman Test, rho 0.8897, phi < 10^{-4}, n = 600) implies that the tendency to select phenotypically simple circuits leads to the selection of low evolvable individuals. In other words, the competition between reproducing individuals drives the evolving population toward a highly robust but low evolvable area of the search space. This area corresponds to the central and most connected region of the neutral network in which the fraction of mutations that does not alter the fitness of the circuits is maximum (Wilke, 2001), but in which the fraction of mutations that gives rise to different unique phenotypes is minimum.

A correlation between the size and the phenotypic variability of circuits has already been reported by Raman and Wagner (2011). In their case, the correlation was observed by comparing randomly generated circuits of different size that computed the same logic function. The correlation, therefore, seems to characterize all circuits, irrespectively of whether or not they were evolved and irrespectively of the function computed.

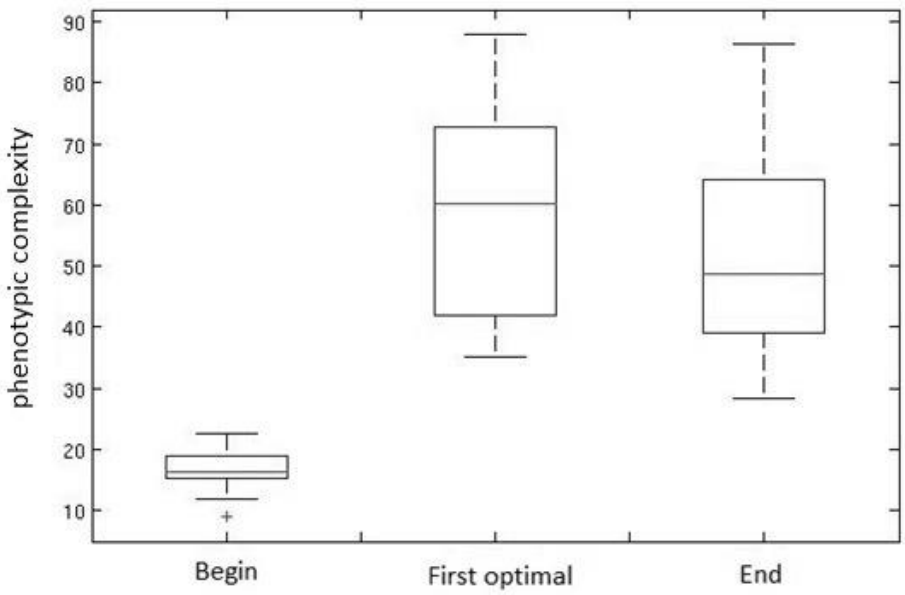


Figure 5.1. Phenotypic complexity of circuits evolved with the (μ+ μ) ES. The box-plots indicate the phenotypic complexity of best circuits of the first generation ("Begin"), of the first circuits that achieved optimal performance ("First optimal"), and of the circuits of the last generation ("End"). Data obtained by analyzing the circuits evolved in the 17 replications carried out with the best parameters that achieved optimal performance.

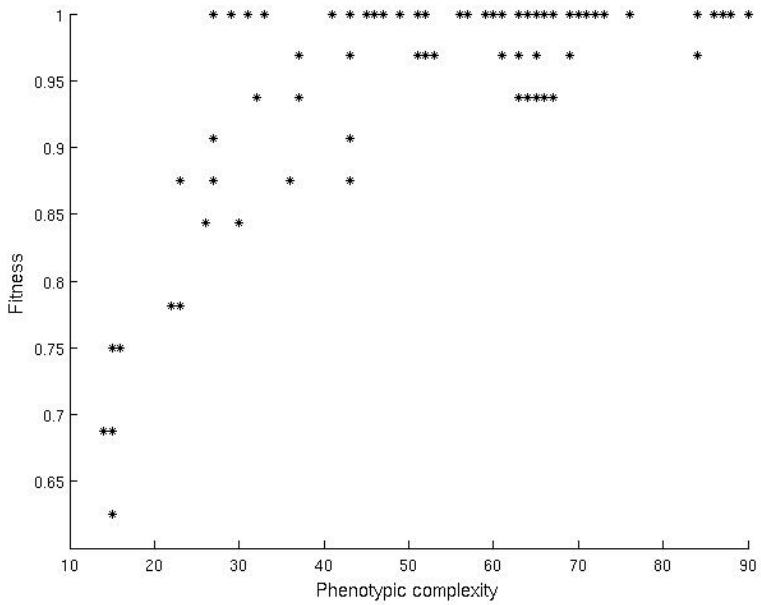


Figure 5.2. Scatter plot of fitness against phenotypic complexity in circuits evolved with the (μ+ μ) ES in 30 replications carried with the best parameters.

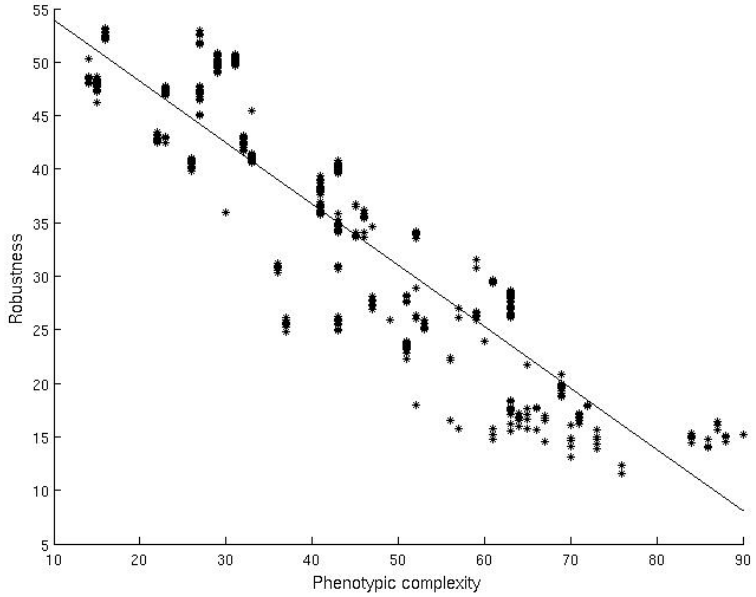


Figure 5.3. Scatter plot of robustness (i.e. fraction of offspring that maintain the same fitness of their parents) and phenotypic complexity in circuits evolved with the (μ+ μ) ES in 30 replications carried with the best parameters.

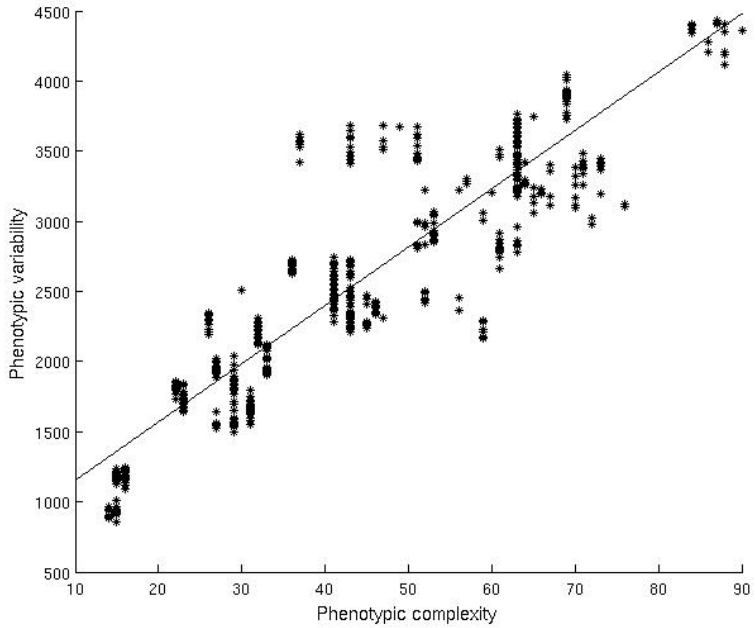


Figure 5.4. Scatter plot of phenotypic variability against phenotypic complexity in circuits evolved with the (μ+1) ES in 30 replications carried with the best parameters.

5.2.3.2 Digital circuits evolved with the $(1+\lambda)$ES

With the best combination of parameters (MutRate=0.02, Stochasticity=0.05, see Table 5.2 and 5.3), evolving circuits find optimal solutions in all replications after 301'400 evaluations, on average. This performance is significantly better than the performance of the circuits evolved with the $(\mu+\mu)$ES (Wilcoxon ranksum test, p-value $<10^{-12}$).

These results are in line with those reported in previous studies (Miller, Job, and Vassiley, 2000) and confirm the efficacy of the $(1+\lambda)$ ES in this domain. Notice also, how, in the case of this method, the optimal results are achieved without Stochasticity, see Table 5.2. The experiments reported in Miller, Job, and Vassiley (2000) were carried out without Stochasticity (the method used did not include the possibility to reduce the selection pressure through the addition of noise to the fitness).

	MutRate 0.01	MutRate 0.02	MutRate 0.03	MutRate 0.04
Stochasticity 0.00	96.66 (43.84)	*[384600] (42.38)	***[301400] (45.50)**	*[395400] (35.58)
Stochasticity 0.02	*[856070] (48.91)	*[544900] (42.76)	*[549700] (42.49)	*[463300] (40.54)
Stochasticity 0.04	86.66 (49.53)	*[541800] (43.11)	96.66 (39.96)	96.66 (40.71)
Stochasticity 0.06	43.33 (34.21)	30 (26.37)	3.33 (26.52)	0 (25.77)

Table 5.2. Performance of circuits evolved with the $(1+\lambda)$ ES in experiments carried out by using different mutation rate and stochasticity levels. λ was set to 10. The first number of each cell indicates the fraction of replications that achieved optimal performance. The experiments in which all replications found optimal solutions are indicated with an asterisk followed by the average number of evaluations that were necessary to find optimal solutions indicated with square brackets. The numbers between parentheses indicate the average size of the evolved circuits.

N. Offspring (λ)	Performance and size
5	*[408400] (44.54)
10	***[301400] (45.50)**
20	*[524900] (43.85)

Table 5.3. Performance of circuits evolved with the $(1+\lambda)$ ES in experiments carried out by using different number of offspring (λ). The mutation rate and stochasticity parameters were set to 0.03 and 0.0, respectively. All experiments found optimal solutions. The numbers indicated with square brackets represent the average number of evaluations that

were necessary to find optimal solutions. The numbers between parentheses indicate the average size of the evolved circuits.

At this point I should try to explain why the $(1 + \lambda)$ ES largely outperforms the $(\mu + \mu)$ ES.
The variability of the population is null in the case of $(1 + \lambda)$ ES. Consequently, the superiority of this method cannot be due to the diversity of the population. The advantage is rather explained by the fact that this method leads to the selection of circuits that are less robust, but that have a greater phenotypic variability than the circuits evolved with the $(\mu + \mu)$ ES (Table 5.4).

The lower performance of the circuits evolved with the $(\mu + \mu)$ ES, on the other hand, is because it drives the population toward a very robust but low evolvable region of the genetic space. As I pointed out above, the competition between evolving circuits, in the experiments carried out with the $(\mu + \mu)$ ES, drives the population toward solutions that are very robust to genetic variations. Since the easiest way to achieve robustness consists in selecting solutions that are phenotypically simple and have a low phenotypic variability, the selection of solutions that are robust to mutations produces a reduction of phenotypic variability that, in turn, causes evolutionary stagnation.
The fact that the $(1 + \lambda)$ ES is able to find optimal solutions in all replications without stochasticity (see Table 5.2) indicates that the stagnation phases affecting the circuits evolved with the $(\mu + \mu)$ ES is not caused only by the characteristics of the fitness surface (i.e., by the presence of local minima), but rather by the combined effect of the fitness surface and the tendency of the population to move toward areas of the genetic space containing highly robust but low-evolvable solutions. We can schematize this process by considering all the candidate solutions with the same fitness as a series of nodes and the genetic variations transforming parents into offspring with the same fitness as links between the nodes. The nodes and the links form one or more neutral networks (Shuster et al., 1994; Van Nimwegen, Crutchfield and Huynen, 1999) constituted by connected candidate solutions. Robustness is higher in the central part of the network in which the number of connections between nodes is maximum and lower in peripheral parts of the network. On the other hand, phenotypic variability is lower in the central part of the network and higher in the periphery (see also Hu et al., 2012). Stagnation thus originates from the tendency of circuits evolved with the $(\mu + \mu)$ ES to move toward the central portion of the neutral network, which has a low phenotypic variability.

	$(\mu + \mu)$	$(1 + \lambda)$	Wilcoxon ranksum p-value
Phenotypic complexity (number of functional gates)	61.0	61.8	>0.05
Phenotypic variability	3826.3	4826.0	$< 10^{-1}$
Robustness (% offspring with a fitness equal or greater than the parent)	23.7565	14.3	$< 10^{-0}$

Table 5.4. Comparison of the characteristics of the first circuits evolved with the $(\mu + \mu)$ and the $(1 + \lambda)$ ES that achieved optimal performance (i.e. the 17 out of 30 replications that achieved optimal performance in the case of the $(\mu + \mu)$ experiments and 30 out of 30 replications in the case of the $(1 + \lambda)$ experiments).

5.2.3.3 Digital circuits evolved with the PSHC algorithm

Table 5.5 shows the results of the experiment carried out by evolving circuits with the Parallel Stochastic Hill Climber (PSHC) algorithm. To analyze the role of the critical parameters, I carried out 20 series of experiments in which I systematically varied the mutation rate and the stochasticity. The Variations parameter was set to 100 and the size of the population was set to 20. As can be see the best results are obtained without stochasticity and with the mutation rate parameter set to 0.02. Table 3.6 shows the results obtained in 7 additional series of experiments in which the Variations parameter was varied and the stochasticity and mutation rate parameters were set to 0.0 and 0.02, respectively (i.e., to the optimal values).

As can be seen, the PSHC algorithm allows the evolving circuits to find optimal solutions in all replications and, in the case of the best parameters (MutRate=0.02 and Stochasticity=0.0, Variations=100), after only 275,300 evaluations. The results obtained with the PSHC algorithm are significantly better than the results obtained with $(\mu + \mu)$ and $(1 + \lambda)$algorithms reported above (Kruskal Wallis test p-value $< 10^{-11}$).

As reported above, the experiments I carried out with a variation of the PSHC algorithm in which the population was divided in into 10, 5, or 2 sub-populations formed by 2, 4 or 10 individuals led to much worse results. Indeed, these experiments achieved optimal performance in only 84%, 62%, and 54% of the replications, respectively. This since the competition among individuals of sub-populations favors the selection of phenotypically simpler solutions, which have a lower evolvability.

	MutRate 0.01	MutRate 0.02	MutRate 0.03	MutRate 0.04	MutRate 0.05
Stochasticity 0.0	*[413,270]	***[275,300]**	*[361,730]	*[692,800]	83.33 (42.88)
Stochasticity 0.01	*[516,270]	*[568,800]	*[815,800]	*[853,730]	80 (46.36)
Stochasticity 0.03	*[537,600]	*[798,630]	*[809,450]	96.66 (45.20)	83.33 (43.67)
Stochasticity 0.05	*[1,024,530]	*[1,065,420]	*[1,160,500]	93.33 (51.29)	73.33 (48.42)

Table 5.5. Performance of circuits evolved with the PSHC method in experiments carried out by varying mutation rate and stochasticity. The number of parents (λ) was set to 20. The number of variations was set to 100. The first number of each cell indicates the fraction of replications that achieved optimal performance. The experiments in which all replications found optimal solutions are indicated with an asterisk followed by the average number of evaluations that were necessary to find optimal solutions indicated with square brackets. The numbers between parentheses indicate the average size of the evolved circuits.

Variations	
1	*[710’300]
10	93.33
50	*[764’200]
100	***[275’300]**
150	*[454’200]
200	*[398’410]
500	*[584’650]

Table 5.6. Performance of circuits evolved with the PSHC method in experiments carried out by with different number of Variations. The mutation rate and stochasticity parameters were set to 0.2 and 0.0, respectively. The number of parents (λ) was set to 20.The first number of each cell indicates the fraction of replications that achieved optimal performance. The experiments in which all replications found optimal solutions are indicated with an asterisk followed by the average number of evaluations that were necessary to find optimal solutions indicated with square brackets. The numbers between parentheses indicate the average size of the evolved circuits.

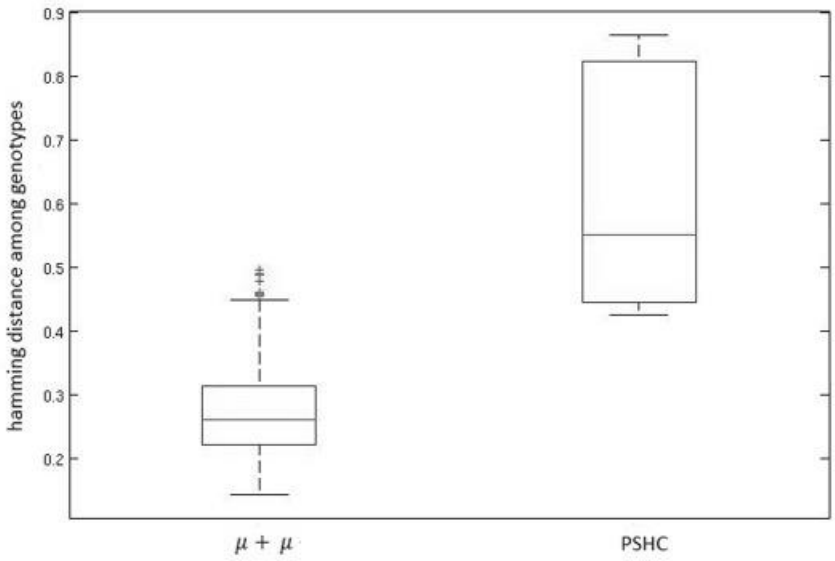


Figure 5.5. Genetic diversity of the populations evolved with the $(\mu + \mu)$ and PSHC algorithms calculated by measuring the average hamming distance, the number of positions at which the corresponding genes are different, between the genotypes of the last generation. Boxes represent the inter-quartile range of the data and horizontal lines inside the boxes mark the median values. The whiskers extend to the most extreme data points within 1.5 times the inter-quartile range from the box. Points marked with the plus sign indicate the outliers.

The comparison of the characteristics of the first circuits that achieved optimal performance evolved with the three different algorithms reveals that circuits evolved with the PSHC have: (i) a greater phenotypic complexity than the circuits evolved with the other two algorithms (Table 5.7, Kruskal Wallis test, p-value <0.0081), (ii) a greater population diversity than the circuits evolved with the (μ+μ)and $(1 + \lambda)$ algorithms (Figure 5.5, Wilcoxon ranksum test, p-value $<10^{-8}$), and (iii) a greater phenotypic variability than the circuits evolved with the $(\mu + 1)$algorithm(Table 5.7, Wilcoxon ranksum test, p-value $<10^{-5}$) but a lower phenotypic variability with respect to the circuits evolved with the $(\mu + 1)$algorithm (Table 5.7, Wilcoxon ranksum test, p-value $<10^{-7}$)

	(μ+μ)	$(1 + \lambda)$	PSHC	Kruskal Wallis p-value
Phenotypic complexity (number of functional gates)	61.0	61.8	71.6333	0.0081
Phenotypic variability	3826.3	4826.0	4198.5	0.0063

Robustness (% offspring with a fitness equal or greater than the parent)	23.75	14.3	18.79	0.0027

Table 5.7. Comparison of the characteristics of the first circuits evolved with different algorithms that achieved optimal performance (i.e. the 17 out of 30 replications that achieved optimal performance in the case of the (μ+μ) experiments and the 30 out of 30 replications that achieved optimal performance in the case of the $(1+\lambda)$ and PSHC experiments).

The comparison between the first circuits that achieved optimal performance (Table 5.7) and the optimal circuits obtained at the end of the evolutionary process (Table 5.8) shows how the evolving circuits tend to become progressively more robust with respect to genetic variations during phases in which the fitness remain stable. Indeed, the percentage of offspring that preserve the same fitness of their parents increases significantly in all cases (Wilcoxon ranksum test, p-value $< 10^{-15}$). The increased robustness is achieved by reducing the phenotypic complexity and, consequently, by reducing the phenotypic variability.

	(μ+μ)	$(1+\lambda)$	PSHC	Kruskal Wallis p-value
Phenotypic complexity (number of functional gates)	48.94	44.83	43.86	>0.05
Phenotypic variability (individual)	2617.7	2211.1	2418.3	0.0375
Robustness (% offspring with a fitness equal or greater than the parent)	33.7	27.34	39.80	0.0058

Table 5.8. Comparison of the characteristics of the circuits evolved with different algorithms at the end of the evolutionary process. Data calculated by using the first optimal individual circuit of the last generation. In the case of the (μ+μ) experiments, I included only the replications that achieved optimal performance.

Overall, these results indicate that the advantage of the PSHC in this problem domain is due to its ability to limit the effects of the competition between reproducing individuals that favours the

selection of phenotypically simple individuals characterized by low phenotypic variability, and to its ability to maximize the variability of the population.

5.2.4 Discussion

In this section, I investigated how the characteristics of the evolutionary algorithm influence the evolvability of candidate solutions, i.e. the propensity of evolving individuals to generate better solutions as a result of random genetic variations. This objective has been pursued by evolving digital circuits which represent a classic domain of application for evolutionary algorithms and which share important properties with natural systems such as proteins, RNA, regulatory circuits and metabolic networks.

The results indicate that in this domain, which is characterized by a high level of neutrality, $(1 + \lambda)$ ES largely outperform (μ+μ) ES. The analysis of the evolutionary dynamics indicates that this difference is due to competition for robustness to mutations among evolving individuals. When individuals compete for robustness and when robustness is achieved simply by minimizing the number of genes playing a functional role, as in the case of the circuits evolved with the (μ+μ) ES, evolution tends to select individuals located in high robust regions of the genetic space which are characterized by low phenotypic variability and, consequently, low evolvability. When the evolving individuals do not compete for robustness to mutations, as in the case of the circuits evolved with the $(1 + \lambda)$ ES, evolution selects individuals that are less robust but have a higher phenotypic variability and evolvability. Overall, these results show how neutrality and robustness to mutations do not necessarily enhance evolvability. They can also reduce evolvability and cause evolutionary stagnation, as also observed by Ancel and Fontana (2000).

Whether robustness to mutations enhances or diminishes phenotypic variability and evolvability depends on whether robustness is achieved through the development of parsimonious (phenotypically simple) solutions minimizing the number of genes playing functional roles, or through phenotypically more complex solutions capable of buffering the effect of mutations. Robustness to mutations of the latter type can evolve as a correlated side effect of the evolution of robustness to environmental variations (De Visser et al., 2003), which cannot be improved through phenotypic simplification. Therefore, whether robustness to mutation enhance or reduce evolvability might depend on whether or not robustness to mutation is combined with robustness to environmental variations. For evidences supporting this hypothesis in the context of evolving digital circuits, see Milano and Nolfi (2016).

The comprehension of the effects that the competition between evolving individuals has on

robustness and evolvability allowed us to design a new evolutionary algorithm, named Parallel Stochastic Hill Climber (PSHC), which outperforms the other two methods considered. This is achieved by limiting the negative effects that the competition among evolving individuals can have on evolvability while preserving the advantage provided by the utilization of a diversified population.

5.3 The preferential selection of larger phenotypes promotes the evolution of better solutions

5.3.1 Introduction

Evolutionary algorithms are known for their ability to discover simple solutions, i.e. the tendency to select the smallest available solutions. This is not surprising since evolution typically proceeds by selecting minimal sub-optimal solutions first and by then progressively extending those solutions until, possibly, an optimal solution is found. From an application point of view, the tendency to select compact solution is also desirable since it permits to generate solutions that are cheaper, lighter, and/or faster than non-compact solutions.

Compact solutions, however, can be less evolvable than more elaborated solutions. Consequently, the tendency to select minimal solutions might reduce the chance to produce behavioral variation because of genetic variations --- a property that constitutes a crucial prerequisite for evolutionary progress.

To produce adaptive heritable variations, evolving programs should possess both robustness to genetic variation and phenotypic variability. Robustness is required to enable a reliable transfer of capacities from parents to offspring in the presence of mutations. Phenotypic variability, namely the ability to produce phenotypic variations because of genetic variation, is required to generate adaptations. One simple way to achieve robustness consists in minimizing the size of the current candidate solutions since this permits to minimize the chance that mutations alter functional components of the solution and consequently the fitness of the solution. Improving robustness in this way however only provides an advantage on the short term since it causes a reduction of phenotypic variability that, in turn, causes a reduction of the chance to produce adaptive variations.

In this chapter I verify whether the preferential selection of phenotypically large solutions permit to preserve and/or enhance the evolvability of candidate solutions and consequently the probability

to evolve high performing solutions in the context of Cartesian Genetic Programing. The results collected on two qualitatively different problems: the eight-bit parity problem and the Paige regression problem indicate that the preferential selection of phenotypically large solutions speed-up the evolutionary process and permit the discover of better solutions. I show that the advantage of the preferential selection of larger solutions can be extended by self-adapting the mutation rate through the one-fifth success rule. Finally, I show that for problems in which neutrality plays a minor role, the advantage of the preferential selection of larger solutions can be extended by preferring larger solutions among quasi-neutral alternative candidate solutions, i.e. also among solutions displaying similar performance.

Recent works have demonstrated the applicability of Cartesian Genetic Programming also to problems that are more complex with respect to those used in this section (see Miller [2019] for a review). I restricted my analysis to two relatively complex problems to permits an easier comparison with results obtained with $(1 + \lambda)$ ES and to evaluate the efficacy of the method proposed on two qualitatively different domains.

Cartesian Genetic Programming (CGP, Miller and Thomson, 2000; Miller, 2011), as described in chapter 3, is a form of Genetic Programming (Koza, 1992; Langdon and Poli, 2002). It is usually used to evolve acyclic computational structures of nodes (graph) indexed by their Cartesian coordinates but can also be extended to evolve recurrent (cyclic) structures (Turner and Miller, 2014). The method has been successfully applied to evolve digital circuits (Miller, Job and Vassilev, 2000 a-b), robots' controllers (Harding and Miller, 2005), Atari games players (Wilson, Cussat-Blanc, Luga and Miller, 2018), neural networks (Khan, Ahmad, Khan and Miller, 2013), image classifier (Harding, Graziano, Leitner and Schmidhuber, 2012), molecular docking (Garmendia-Doval, Morley and Juhos, 2003) and regression programs (Harding, Miller and Banzhaf, 2009).

CGP programs are usually evolved through a $(\mu + \lambda)$ Evolutionary Strategy (Rechenberg, 1973; Beyer and Schwefel, 2002), where μ is set to 1 and λ is usually set to 4 (Miller and Thomson, 2000; Miller, 2011). $(1 + \lambda)$ Evolutionary Strategies can also be referred as hill climbers. This means that during each generation a single parent produces four offspring with mutations, as described above. The mutation probability is usually uniform for all genes. When none of the offspring is better than the parent and at least an offspring equals the parent, the offspring is selected. The comparative study reported in Milano, Pagliuca and Nolfi (2017) indicates that the utilization of a single parent might play an important role in CGP. Indeed, CPG programs evolved on the five-bits parity task with μ = 1 largely outperformed programs evolved with μ> 1.

5.3.1.1 On the relation between size and phenotypic variability

We now discuss the relation between functional size and phenotypic variability, i.e. the propensity of candidate solutions to vary phenotypically because of mutations. The phenotypic variability of a candidate solution can be estimated by measuring the number of programs located in the genetic neighborhood of the original candidate solution displaying different unique behaviors. In the context of digital circuits, programs displaying different unique behaviors correspond to candidate solutions that produce output matrixes that differ among themselves for at least one output value. Phenotypic variability constitutes a crucial prerequisite for adaptive progress. Clearly, in the majority of cases offspring producing outputs that differ from those produced by their parent will have a lower fitness than the parent. However, the larger the number of different unique solutions located around the candidate solution, the greater the probability that the genetic neighborhood includes at least one adapted solution.

A positive correlation between size and phenotypic variability was reported by Raman and Wagner (2011). They analyzed digital circuits with randomly generated genotypes that included four inputs, four outputs, and a variable number of nodes. The function list of the nodes included the OR, AND, XOR, NAND and NOR functions. Phenotypic variability was estimated by performing 2,000 steps function-preserving random walk from the original circuit. The random walk was realized by: (i) generating a varied circuit by mutating one gene, (ii) incrementing a counter if the varied circuit produce a different unique behavior (i.e. computes a function that differ from the functions computed by the original circuit and by the previous varied circuits), (iii) preserving or removing the mutation depending on whether the varied circuit has the same fitness of the original circuit or not, (iv) repeating the previous three operations for 2,000 steps. Raman and Wagner (2011) refer to this measure with the term evolvability rather than with the term phenotypic variability. I prefer to distinguish between the propensity of a program to vary phenotypically independently of whether the variations are adaptive or nor (phenotypic variability) and the propensity of a program to generate adaptive variations (evolvability).

The presence of a positive correlation was confirmed by the analysis of digital circuits evolved for the ability to solve a five-bit even parity problem (Milano, Pagliuca and Nolfi, 2018). Indeed, the fitness and the functional size of evolved circuits were strongly correlated. Moreover, the circuits evolved with the $(1 + \lambda)$ ES were significantly larger than the circuits evolved with the $(\mu + \mu)$ ES. This combined with the observation that $(1 + \lambda)$ ES method largely outperform the $(\mu + \mu)$ ES method, with respect to speed and fraction of experiments achieving optimal performance, indicates that the advantage of the $(1 + \lambda)$ ES method is because it tends to evolve larger programs

with respect to the $(\mu + \mu)$ ES method (Milano, Pagliuca and Nolfi, 2018).

The relation between robustness to mutation and phenotypic variability can be illustrated by relying on the notion of neutral network (Shuster et al., 1994; Van Nimwegen, Crutchfield and Huynen, 1999). A neutral network is constituted by a series of nodes connected through bidirectional links in which nodes correspond to candidate solutions with identical fitness and links correspond to single mutations that enable to transform a candidate solution in genetically different candidate solution achieving the same fitness. The number of links per nodes and consequently the robustness to mutation is higher in the central part of the neutral network (Shuster et al., 1994; Van Nimwegen, Crutchfield and Huynen, 1999). Under neutral evolution, evolving individuals do not move in an entirely random way over the neutral network but move preferentially in highly connected parts ofthe network, resulting in phenotypes that are relatively robust against mutations (Van Nimwegen, Crutchfield and Huynen, 1999). Such robust regions of the neutral networks include minimal solutions that achieve robustness by minimizing the number of functional genes but which have a low phenotypic variability (Wagner 2008, 2011; Hu et al., 2012). The preferential selection of larger solutions drives the evolutionary search toward the peripheral areas of the neutral network and/or toward solutions that achieve robustness through redundancy or degeneracy (Tononi, Sporns and Edelman, 1999; Edelman and Gally, 2001, Milano and Nolfi, 2016). These areas are characterized by a greater phenotypical variability than areas including minimal solutions. In other words, the preferential selection of larger solutions drives the evolutionary search toward solutions that are more evolvable.

Whether these properties hold also in the case of continuous problem is unknown due to the difficulty of measuring phenotypic variability in systems of this kind.

Several works investigated the possibility to minimize the size of the program once an optimal solution is found (Kalganova and Miller, 1999; Vassilev, Job and Miller, 2000); Gajda and Sekanina, 2007 and 2010, found that extending the number of generations of CGP without rewarding smaller perfect solutions actually gave smaller circuits than when you reward small circuits. This is suggestive that there is a tendency for CGP to make the phenotypes small even when they have a perfect fitness. This objective is typically realized by preferentially selecting phenotypically smaller solutions once an optimal solution is found. This is particularly interesting in the case of digital circuits since it enables to minimize the number of electronic components required to manufacture the circuit in hardware. The objective of this work is different since I study the factors that can promote evolution of high-quality solutions independently from the size of the solution.

5.3.2 Method

As mentioned above, I compared the performance of two algorithms: a standard $(1+\lambda)$ ES and a $(1+\lambda)$ ES-PL algorithm that preferentially select program with functionally larger phenotypes. In both cases, during each generation a single parent produces λ offspring with mutations. In the case of the $(1+\lambda)$ ES method, the $1+\lambda$ candidate solutions are sorted primarily on the basis of their fitness (in descending order), and secondarily based on whether they correspond to an offspring or to the parent (in this order). The first individual is then used to replace the parent. In the case of the $(1+\lambda)$ ES-PL method, instead, the $(1+\lambda)$ individuals are sorted primarily based on their fitness (in descending order), secondarily based on the size of their functional circuit (in descending order), and then based on whether they correspond to an offspring or to the parent (in this order). The first individual is then used to replace the parent.

Algorithm $(1+\lambda)$ ES-PL

1. Generate randomly the starting parent genotype (G_0)
2. Evalute G_0 and get its fitness (F(G_0))
3. Generate λ mutated offspring of G_0, (M_λ) and evaluate them F(M_λ)
4. Sort G_0 and M_λ by fitness, the best is the new G_0
5. If parent parent and offspring share the same fitness order them by size and the greater is the new G_0

The chromosomes of candidate solutions include N three-digit tuples that encode the function and the input of N corresponding nodes and M one-digit tuples that encode the index of M corresponding output nodes. The value of the genes encoding the input of a node are generated randomly with a uniform distribution in the range [0, G-1], where G is the index of the corresponding node. Offspring are generated by creating a mutated copy of the parent chromosome. The genes of the initial chromosome are created randomly with a uniform distribution in the range described above. Mutations are realized by replacing a fraction of randomly selected genes with new integers selected randomly with a uniform distribution in the ranges described above. Nodes are not arranged in fixed layers (i.e. I used a 1D representation, where the numbers of row is set to one and eache nodes is connected to one of the previous).

In the first series of experiments, programs were evolved for the ability to solve an 8 bit even parity problem and were provided with 8 inputs and 1 output. As in related studies (e.g. Thomson and

Miller, 2000), the function list of the nodes included the following four logic operators [AND, NAND, OR, and NOR]. Programs were evaluated on 2^8 patterns, i.e. on all possible input patterns. The λ and the mutrate parameters was set to 4 and to 2%, respectively. The evolutionary process was terminated when a candidate solution achieved optimal performance or after a total of 1 million candidate solutions were evaluated. The parameter N was varied systematically in the range [100-1000] within multiple experiments. The fitness of a candidate solution was computed based on the inverse of the offset between the outputs produced by the program and the desired outputs. More specifically, the fitness is calculated based on the following equation:

$$F = 1 - \frac{1}{2^n} \sum_{j=1}^{2^n} |O_j - E_j| \qquad (11)$$

Where *n* is the number of inputs of the circuit, *j* is the number of the input patterns varying in the range [1,2^n], O_j is the output of the circuit for pattern *j*, E_j is the desired output for pattern *j*.

In the case of the regression problem, I used the Pagie function (12) that constitutes a challenging benchmark (Pagie and Hogeweg, 1997; Turner and Miller, 2015). Following Turner and Miller (2015), programs are provided with two inputs, 1 output and N nodes. The parameter N was varied systematically in the range [50-200] in multiple experiments. The input patters consist of 676 samples taken randomly from the range x_1 in [-5.0, 5.0] and x_2 in [-5.0, 5.0] with an uniform random distribution, and the function set contains the following operators: [:, +, −, ∗, and /]. The fitness encodes the regression loss, i.e. the sum of the absolute errors produced by the program. To measure the fraction of times in which the evolutionary process manages to solve the problem I consider the problem solved when the total error is lower than 0.1. The continuous nature of this problem implies that the probability that offspring have identical fitness is much smaller in this problem than in the parity problem.

$$Pagie\ Function \qquad f(x_1, x_2) = \frac{1}{1 + x_1^{-4}} + \frac{1}{1 + x_2^{-4}} \qquad (12)$$

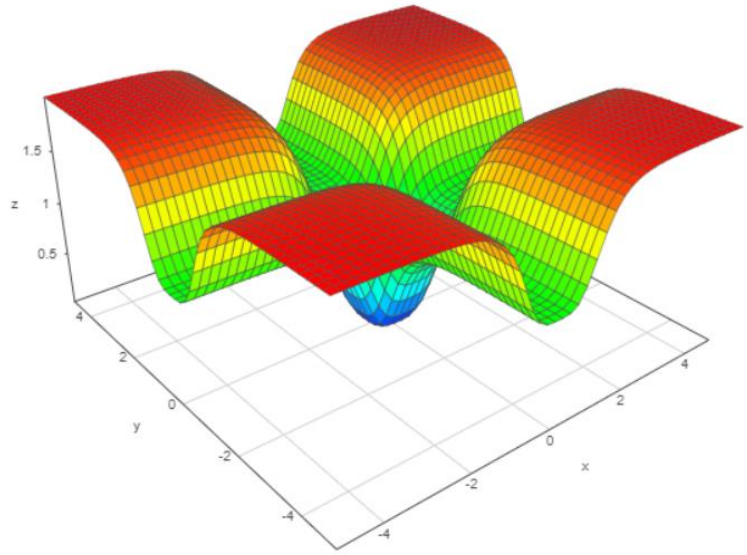


Figure 5.7. The Pagie function.

To automatically adapt the mutation rate to the characteristics of the problem and to the specific evolutionary phase I used the one-fifth success rule that varies the mutation rate so to maintain the ratio of offspring that are equal or better than the parent to 1/5 (Rechenberg, 1973; Schumer and Steiglitz, 1968). This is realized by varying the mutation rate after the evaluation of each offspring on the basis of the following equation:

$$Mrate = \begin{cases} Mrate * 1.4 & if\ f(offsp.) \geq f(parent) \\ Mrate * 1.4^{-\frac{1}{4}} & otherwise \end{cases} \qquad (13)$$

We will refer to the variants of the two algorithms described above that include the automatic adaptation of the mutation rate with the name (1 + 4) ES-AM and (1 + 4) ES-PL-AM.

The ratio between offspring that have a fitness equal or greater than their parent and offspring that have a lower fitness depends on the mutation rate, i.e. higher the mutation rate is higher the probability that the offspring receive maladaptive mutations is. Moreover it depends on the following factors: (i) the local fitness surface (the greater the derivative of the local fitness surface is, the greater the chance that mutations cause maladaptive effects is), (ii) the number of functional genes (the greater the number of functional genes, the greater the probability that mutations have maladaptive effect is), and (iii) the robustness to mutations (the greater the robustness of the current parent to mutations is, the lower the probability that mutations cause a fitness loss is). The

one-fifth success rule permits to appropriately tune the mutation rate to the variations of these factors across generations.

5.3.3 Results

5.3.3.1 The parity problem

Table 5.9 reports the results of 20 series of experiments performed with the $(1 + 4)$ ES in which I systematically varied the size of the genotype (i.e. the number of nodes) and the mutation rate. As can be seen, the value of these two hyper-parameters have a strong impact on performance.

This can be explained by considering that that length of the genotype influences the size of the functional circuit of evolving candidate solutions (Table 5.9, data indicated within parenthesis) that in turns influences the evolvability of candidate solutions. Moreover, it can be explained by considering that the optimal mutation probability is influenced by the size of the functional circuit. This since, the higher the size of the functional circuit, the higher the probability that random variations alter the functional circuit. The problem is further complicated by the fact that the size of the functional circuit can change significantly over generations (see Figure 5.8).

	MutRate 1%	**MutRate 2%**	**MutRate 3%**	**MutRate 4%**
Nodes 100	0% (32.1)	35% (35.3)	10%(33.7)	5%(31.6)
Nodes 200	55% (42.4)	35% (41.6)	20%(38.7)	10%(36.9)
Nodes 400	70% (49.3)	50%(47.9)	25%(46.8)	10%(45.5)
Nodes 600	70% (56.3)	40%(55.9)	10%(54.2)	0%(53.6)
Nodes 1000	70% (65.4)	20% (66.8)	5% (64.3)	0% (62.3)

Table 5.9. Performance of the best circuits evolved with the (1+4**)** ES obtained by varying the number of nodes and the mutation rate. The number in each cell indicates the percentage of replications that found an optimal solution over 30 replications after one million of evaluations, which correspond to 200.000 generations. The number in parentheses indicate the average size of the functional circuits at the end of the evolutionary process.

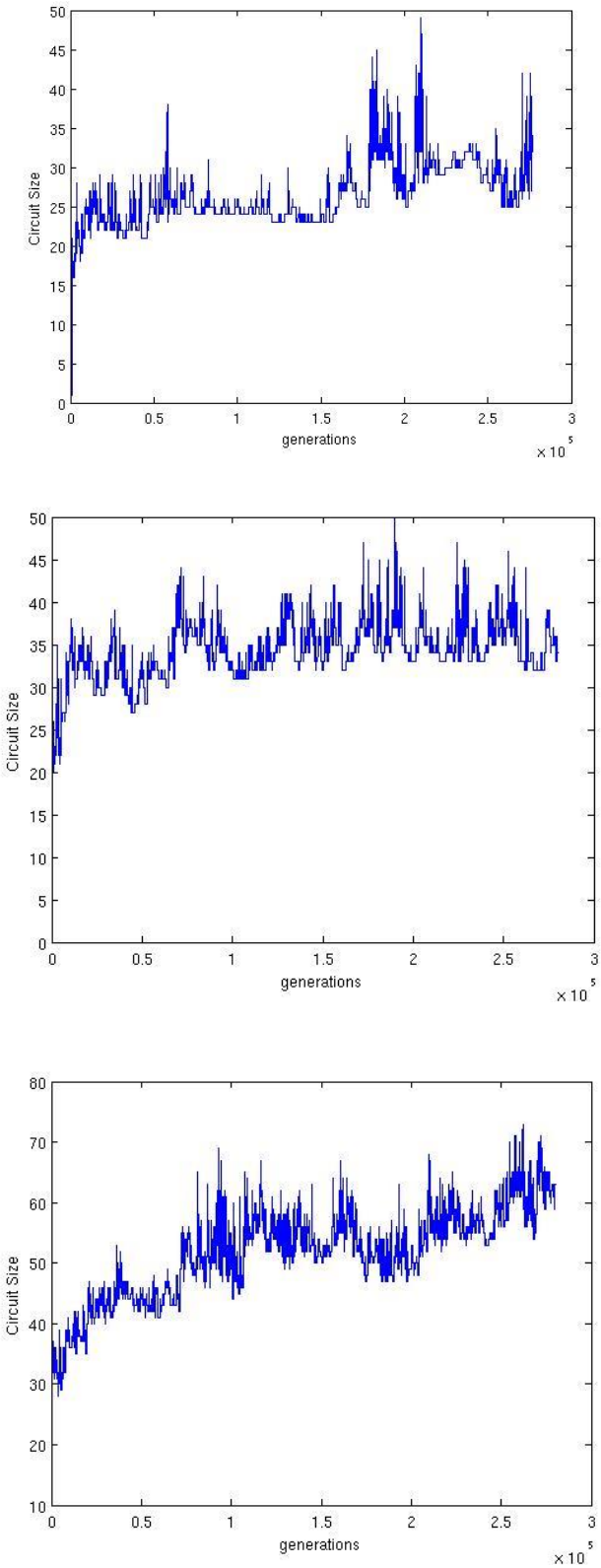


Figure 5.8. Functional size of evolving circuits across generations in a typical replication. Result obtained in the experiment carried with 100, 400, and 1000 gates, respectively with the optimal mutation rate (i.e. 2%, 1%, and 3%).

The $(1 + 4)$ ES-AM method represents a good solution to these problems since it releases the experimenter from the burden of setting the mutation rate hyperparameter, permits the adaptation of the mutation rate dynamically during the course of the evolutionary process, and permits to

achieves performance that are better or at least equally good with respect to the performance obtained by the $(1 + 4)$ ES method with the mutation rate optimized for each condition (Table 3.10). Indeed, the $(1 + 4)$ ES-AM method permits to achievement of good performance without manually optimizing the mutation rate providing that the genotype includes at least 400 gates. The performance of the ES-AM method does not statistically differ from the performance of the ES method with optimal mutation rate in all cases (Mann-Whitney U-test p-value > 0.05).

The fact that the $(1 + 4)$ ES-AM-PL method outperforms the $(1 + 4)$ ES-AM when the length of the genotype is 100 or 200(Mann-Whitney U-test p-value < 0.001) and produce equally good performance in the other cases (Mann-Whitney U-test p-value > 0.05) demonstrates that the preferential selection of larger phenotype is advantageous when the size of the genotype is sub-optimal.

CGP Nodes	**50**	**100**	**200**	**400**	**600**	**800**	**1000**
$(1 + 4)$ ES	0% [1%]	35% [2%]	55% [1%]	70% [1%]	70% [1%]	70 [1%]	70 [1%]
$(1 + 4)$ ES-AM	0%	30%	30%	75%	70%	70%	63%
$(1 + 4)$ ES-AM-PL	0%	75%	80%	75%	70%	70%	70%

Table 5.10. Performance obtained with the (1+λ) ES and with the (1+λ) ES-AM algorithms in experiments carried with genotype of different length. The results with the former method refer to those obtained by setting the mutation rate to the value that resulted optimal in each condition (indicated in square brackets). The first raw reports the same data described in Table 3.1 for experiments with 100-1000 Nodes. In the case of the experiment carried with the other methods the initial mutation rate was set to 2%.

5.3.3.2 The Paige regression problem

In the case of the Paige regression problem, the preferential selection of larger solutions permits to achieve better performance (Table 5.11). Indeed, the performance achieved by the $(1 + 4)$ ES-AM-PL method are significantly better than the performance achieved by the $(1 + 4)$ ES-AM method (Mann-Whitney U-test p-value < 0.001) in all cases.

The fraction of replication that manage to successfully solve the problem, however, is rather small. To verify whether a less greedy selection strategy could improve performance, I designed a $(1 + 4)$ ES-AM-QN methods in which the parent is replaced with one of the offspring even when the

performance of the offspring is lower than that of the parent up to a limit of 10%. This new method permits to retain not only offspring that received adaptive or neural mutations but also offspring that received quasi-neutral counter-adaptive mutations. Moreover, I designed the $(1 + 4)$ ES-AM-QN-PL method that selects the individual with the larger functional solution among quasi-neutral multiple offspring.

Algorithm	**CGP Nodes 50**	**CGP Nodes 100**	**CGP Nodes 200**
$(1 + 4)$ ES-AM	0%[75.65]	6%[65.23]	6%[66.25]
$(1 + 4)$ ES-AM-PL	3%[62.45]	10%[56.43]	10%[59.63]
$(1 + 4)$ ES-AM-QN	23%[42.21]	16%[44.76]	16%[42.22]
$(1 + 4)$ ES-AM-QN-PL	46%[26.57]	46%[25.53]	26%[31.04]

Table 5.11. Performance of the best circuits evolved with the (1+4) ES-AM, (1+4) ES-AM-PL, (1+4) ES-AM-QN and (1+4) ES-AM-QN-PL in three series of experiments in with the number of nodes is set to 50, 100, and 200. The number in each cell indicates the percentage of replications that successfully solved the problem over 30 replications after 500.000 evaluations that corresponds to 100.000 generations. The number in square bracket indicate the average loss of the best-evolved candidate solutions averaged over 30 replications. The initial mutation rate was set to 2%.

As shown in Table 5.11, the utilization of a quasi-neutral selection scheme and the utilization of the preferential selection of larger solutions lead to significantly better results. Indeed, the performance obtained with the $(1 + 4)$ ES-AM-QN method are significantly better than the performance obtained with the $(1 + 4)$ ES-AM-PL and $(1 + 4)$ ES-AM methods (Mann-Whitney U-test p-value < 0.001). Moreover, the performance obtained with the $(1 + 4)$ ES-AM-QN-PL method are significantly better than the performance obtained with all other methods (Mann-Whitney U-test p-value < 0.001).

The utilization of quasi-neutral selection scheme did not provide an advantage in the case of the parity problems (results not shown). This can be explained by considering that the probability to produce neutral offspring, i.e. offspring achieving the same fitness of their parent, is much higher in the parity problem than in the Paige regression problem.

5.3.4 Discussion

In this section, I demonstrated how the efficiency of Cartesian Genetic Programming methods could be enhanced through the usage of adaptive mutation and through the preferential selection of phenotypically larger solutions.

The rationale behind the method proposed is that evolution tends to select solutions that are robust with respect to mutations and that the simplest way to achieve robustness consists in selecting programs that use a minimal number of nodes. This since smaller the number of functional nodes is, smaller the probability than these nodes are altered by mutations is. As pointed out in previous research (Raman and Wagner, 2011; Milano, Pagliuca and Nolfi, 2017) the chance to produce phenotypically varied programs because of genetic variations is positively correlated with the size of the solutions. Consequently, the tendency to select minimal solutions that are very robust to mutations reduces the probability to generate better solutions because of mutations that, in turn, reduces the efficacy of the evolutionary process. The preferential selection of larger solutions introduced in this section permits to neutralize this negative effect. It permits to promote the selection solutions that have a greater chance to generate better solutions because of genetic variation.

The advantage of the preferential selection of larger solutions has been validated on the eight-bit parity classification and on the “Paige” regression problem. In the former case, the preferential selection of larger solutions among equally good offspring permit to improve the performance in experiments in which the length of the genotype is sub-optimal. In the latter case, the preferential selection of larger solutions led to better performance in all cases. I also showed how the utilization of a quasi-neutral selection operator permit to further improve performance in a continuous regression problem characterized by a lower level of neutrality than the parity problem. The advantage provided by the preferential selection of larger solutions can be further extended by self-adapting the mutation rate through the one-fifth success rule (Rechenberg, 1973; Schumer and Steiglitz, 1968). The advantage of the combined effect of the preferential selection of larger solutions and of self-adaptation of the mutation rate is because the robustness of evolving solutions is strongly influenced by the size of their functional circuit. Consequently, the dynamic adaptation of the mutation rate becomes especially important when the functional size of the evolving solutions varies significantly among generations.

Chapter 6. General conclusion

In this work, I investigated the relation between environmental variations and evolvability evolving digital circuits and neuro-agents. The results of both studies demonstrated how environmental variations promote the evolution of better solutions, i.e. increase the evolvability of the agents.

In the case of the experiments with neuro-agents situated in an external environment I demonstrated how the advange is maximum when the environment varies at a moderate rate. The analysis of the course of the evolutionary process indicates that this advantage is due to the fact that environmental variations promote evolutionary change expecially when the rate of environmental variation is moderate.

In the case of the experiment performed by evolving digital circuits, I demonstrated how circuits evolved in variable environmental conditions (i.e. circuits subjected to faults) outperform circuits evolved in stable environmental conditions, independently from whether they are evaluated in faults and no-faults conditions. Also in this case, therefore, environmental variations enhance evolvability.

The nature of digital circuits permits to carry on detailed analyses that are difficult to perform on neuro-controlled agent. In particular, permits to measure the phenotypic complexity of the evolving solutions. This analysis revelead that evolvability is strongly influenced by the way in which the evolving circuit achieve robustness with respect to genetic variations. More specifically by whether robustness with respect to genetic variations is achieved by minimizing the number of functional genes or through redundancy and degeneracy mechanisms, capable of buffering the effect of genetic variations. The former and the latter case lead to phenotypically smaller and larger solutions, respectively. Given that evolvability is positively correlated with the size of the phenotype, the way in which robustness is achieved influences the evolvability of the systems. As for the case of robustness with respect to genetic variations obtained through redundancy or degeneracy, robustness to environmental variations constituted by faults lead to more complex and more evolvable phenotypes.

These analyses enabled me to explain why $(1 + \lambda)$ ES largely outperform (μ+μ) ES in the context of the evolution of digital circuits and more generally in the context of Cartesian Genetic Programming. This difference arises from the fact that the competition among the offspring of multiple parents lead to the selection of circuits that optimize robustness with respect to genetic variations by

minimizing the size of the functional circuits which produce as a side effect a reduction of the system evolvability.

Finally, these analyses enabled me to design two new algorithms that outperform standard algorithm in the context of Cartesian Genetic Programming. The first algorithm, named Parallel Stochastic Hill Climber (PSHC), combines the $(1 + \lambda)$ ES and the (μ+μ) ES to exploit the advantage of using a diversified population while reducing the tendency to select phenotypically simple but low evolvable solutions. The second algorithm, instead, improve performance by directly selecting phenotypically larger solutions among solution with identical or similar performance.

4.1 Contributions to knowledge

In summary, the major contributions of this thesis are the following:

- the thesis provides the first systematic analysis of the role of environmental variations in artificial evolution and demonstrate how evolving agents benefit from the exposure to moderate environmental variations in the problems considered.
- the thesis demonstrates the presence of a positive correlation between phenotypic complexity and evolvability.
- the thesis introduces two new algorithms that outperform standard methods used in Cartesian Genetic Programming, at least on the problem considered.
- the analyses reported in the thesis permit to elucidate why certain algorithms outperform alternative algorithms in the context of Cartesian Genetic Programming.

Chapter 7. Future Works

Recent works have demonstrated the applicability of Cartesian genetic programming also to problems that are more complex with respect to those used in this thesis (see Miller 2020 for a review). I restricted my analysis to two relatively complex problems to enable an easier comparison with results obtained with standard methods and to evaluate the efficacy of the method proposed on two qualitatively different domains. Future works might investigate the applicability of our methods to more complex problems and the potential advantages of more elaborated mutation methods such as the single active mutation (Golman and Punch, 2017) or insertion (Kalkreuth et al 2015). Futhermore, continuous problem like double pole balancing or Paige regression could be discretized in the fitness landscape (Koza ,1992), doing this neutrality is allowed, while is not present in continuous fitness task. In this way is possible to try to extend the principal findings of this thesis also in continuous domain.

References

Abdelhalim L., Blachon S. Selbig J and Nikolosky Z. (2011). Robustness of metabolic networks: A review of existing definitions. Biosystems, (106) 1: 1-8.

Adami C. (2002). Sequence complexity in Darwinian evolution. Complexity, 8:49-57.

Ancel L.W. and Fontana W. (2000).Plasticity, evolvability, and modularity in RNA. Journal of Experimental Zoology part B Molecular Developmental Evolution, 288: 242–283.

Anderson C.J.R. (2013). The role of standing genetic variation in adaptation of digital organisms to a new environment. Artifical Life 13:3-10.

Back T. (1994). Selective pressure in evolutionary algorithms: A characterization of selection mechanisms. in Proc. 1st IEEE Conf. Evol. Comput.,Jun. 27–29, 1994, pp. 57–62

Bäck T., and Hammel U. (1994). Evolution strategies applied to perturbed objective functions. In Proceedings of the International Conference on Evolutionary Computation.pp. 40–45.

Balch M. (2003).Complete digital design. New York, NY: McGraw-Hill.

Bedau M.A. and Packard N.H. (2003) Evolution of evolvability via adaptation of mutation rates. Biosystems, 69: 143–162.

Beyer H. G. and Schwefel H. P. (2002). Evolution strategies—A comprehensive introduction. Natural Computation, (1) 1: 3–52.

Blythe P.W. (1998). Evolving robust strategies for autonomous flight: A challenge to optimal control theory. In I. Parmee (Ed.), Adaptive Computing in Design and Manufacture, 269-283. London: Springer Verlag.

Bongard J. (2011). Morphological change in machines accelerates the evolution of robust behavior. Proceedings of the National Academy of Sciences (108) 4: 1234-1239.

Branke J. (2002). Evolutionary Optimization in Dynamic Environments. New York: Springer Science and Business Media Press.

Carlson J M, Doyle J (2002). Complexity and robustness. PNAS 99:2538-2545.

Carvalho J.T. and Nolfi S. (2016). Behavioral plasticity in evolving robots. Theory in Biosciences, 135: 201-216.

Crutchfield J.P., Görnerup O. (2006). Objects that make objects: the population dynamics of structural complexity. Journal of The Royal Society Interface, 3:345-349.

Cruz C., Gonzalez J.R. and Pelta D.A. (2011). Optimization in dynamic environments: a survey on problems, methods and measures. Soft Computing, 15: 1427-1488.

De Visser J. A. et al. (2003). Perspective: Evolution and detection of genetic robustness. Evolution, 57 (9): 1959-1972.

Di Paolo E. (2002). Spike-time dependent plasticity for evolved robots. Adaptive Behavior (10) 3-4: 243-263.

Earl D.J. and Deem M.W. (2004). Evolvability is a selectable trait. PNAS, 101: 11531–11536.

Edelman G.M., and Gally J.A. (2001). Degeneracy and complexity in biological systems. Procedings of the National Academy of Science USA, 98 (13): 763–768.

Florian R. V. (2007). Correct equations for the dynamics of the cart-pole system. Technical Report. Center for Cognitive and Neural Studies. Coneural, Romania.

Frei, R., & Whitacre, J. (2012). Degeneracy and networked buffering: principles for supporting emergent evolvability in agile manufacturing systems. *Natural Computing*, *11*(3), 417-430.

Garmendia-Doval A.B., Morley D.S., and Juhos S. (2003). Post docking filtering using cartesian genetic programming. In P. Liartdet (Ed.), Proceedings of the 6th International Conference on Artificial Evolution, pp. 435–446.

Harding S. and Miller J.F. (2005). Evolution of robot controller using cartesian genetic programming. In Proceeding of the 8th European Conference on Genetic Programming, vol. 3447. Lausanne, Switzerland, pp. 62–73.

Harding S., Graziano V., Leitner J. and Schmidhuber J. (2012). MT-CGP: Mixed type Cartesian genetic programming. In Proceedings of the 14th Genetic and Evolutionary Computation Conference (GECCO). Philadelphia, PA, USA, pp. 751–758.

Harding S., Miller J., and Banzhaf W. (2009). Self modifying Cartesian genetic programming: Fibonacci, squares, regression and summing. In Proceedings of the 12th European Conference on Genetic Programming (EuroGP), vol. 5481. Tuebingen, Germany, pp. 133–144.

Hare M. and Elman J.L. (1995). Learning and morphological change. Cognitiom (56) 1: 61-98.

Hartmann M., and Haddow P. (2004). Evolution of fault tolerant and noise-robust digital designs. Computers and Digital Techniques, IEE Proceedings 151: 287-294.

Hazen RM, Griffin PL, Carothers JM, Szostak JW (2007). Functional information and the emergence of biocomplexity. Proceedings of the National Academy of Sciences, 104:8574-8581.

Houle, D. (1992). Comparing evolvability and variability of quantitative traits. Genetics 130, 195-204.

Hu T., Payne J.L., Banzhaf W, and Moore J.H. (2012). Evolutionary dynamics on multiple scales: a quantitative analysis of the interplay between genotype, phenotype, and fitness in linear genetic programming. Genetic Programming and Evolvable Machines (13) 3: 305–337.

Igel C. (2003). Neuroevolution for reinforcement learning using evolution strategies. In R.Sarker et al. (Eds) Congress on evolutionary computation, vol 4. IEEE Press, New York, pp 2588-2595.

Jacoby N. (1997). Evolutionary robotics and the radical envelope-of-noise hypothesis. Adaptive Behavior, 6 (2): 325-368.

Jin Y. (2005). Evolutionary optimization in uncertain environments: A survey. IEEE Transactions on Evolutionary Computation, 9 (3): 303-317.

Kashtan N., Noor E., and Alon U. (2007). Varying environments can speed up evolution. PNAS, (104) 34: 13711-13716.

Kawecki T.J. (1994). Accumulation of deleterious mutations and the evolutionary cost of being a generalist. American Naturalist 144: 833–838.

Kalkreuth, Rudolph, Krone. (2015). Improving convergence in Cartesian genetic programming using adaptive crossover, mutation and selection. IEEE Symposium Series on Computational Intelligence, pp. 1415–1422

Khan M.M., Ahmad A.M., Khan G.M., and Miller J. (2013). Fast learning neural networks using cartesian genetic programming. Neurocomputing, 121: 274-289.

Kirschner M. and Gerhart J. (1998). Evolvability. PNAS, 95: 8420–8427.

Koza J.R. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA: MIT Press.

Krischner M.W. and Gerhart J.C. (2005). The Plausibility of Life: Resolving Darwin's Dilemma. USA: Yale University Press.

Langdon W.B. and Poli R. (2002). Foundations of Genetic Programming. Berlin: Springer Verlag.

Lenski RE, Barrick JE, Ofria C. (2006). Balancing robustness and evolvability. PLoS Biology, 4: 2190-2192.

Levitan B. and Kauffman S. (1994). Adaptive walks with noisy fitness measurements. Molecular Diversity, (1) 1:53–68.

Macia J., and Solé R.V. (2009). Distributed robustness in cellular networks: Insights from synthetic evolved circuits." Journal of The Royal Society Interface 6 (33): 393–400.

Masel J. and Trotter M.V. (2010). Robustness and evolvability. Trends in Genetics, 26 (9): 406-414.

Massera G., Ferrauto T., Gigliotta O. and Nolfi S. (2014). Designing adaptive humanoid robots through the FARSA open-source framework. Adaptive Behavior (22) 4: 255-265.

Mikkulainen R. and Gomez F. (1997). Incremental evolution of complex general behavior. Adaptive Behavior (5) 3-4: 317-342.

Milano N., Carvalho J. and Nolfi S. (in press). Environmental variation promotes adaptation in artificial evolution. Proceedings of the EEE Symposium Series on Computational Intelligence (IEEE SSCI). IEEE Press.

Milano N., Nolfi S. (2016). Robustness to faults promotes evolvability: Insights from evolving digital circuits, PLoS ONE. 11(7): e0158627

Milano N., Pagliuca P. and Nolfi S. (2017). Robustness, evolvability and phenotypic complexity: Insights from evolving digital circuits. arXiv preprint 1712.04254.

Miller J. and Hartmann M. (2001). Evolving messy gate for fault tolerance: some preliminary findings. In Proceedings 3rd NASA Workshop on Evolvable Hardware, 116-123.

Miller J.F. (2011). Cartesian Genetic Programming. Berlin: Springer Verlag.

Miller J.F. and Smith S.L. (2006). Redundancy and computational efficiency in Cartesian genetic programming. IEEE Transactions on Evolutionary Computations, 10 (2): 167-174.

Miller J.F. and Thomson P. (2000). Cartesian genetic programming. In Proceedings of the Third European Conference on Genetic Programming (EuroGP), vol 1820. Berlin: Springer Verlag, pp. 121–132.

Miller J.F., Job D., and Vassiley V.K. (2000). Principles in the evolutionary design of digital circuits. Journal of Genetic Programming and Evolvable Machines 1 (1): 8-35.

Miller J.F., Thompson A., Thompson P. and Fogarty T. (Eds.) (2000). Proceedings of the 3rd International Conference on Evolvable Systems: From Biology to Hardware. Lecture Notes on Computer Science, no. 1801. Berlin, Germany: Springer Verlag.

Miller, J.F. Cartesian genetic programming: its status and future. Genet Program Evolvable Mach 21, 129–168 (2020).

Newman M. (2010). Networks: An Introduction. (Oxford University Press, Oxford, 2010)

Nguyen T.T., Yang S. and Branke J. (2012). Evolutionary dynamic optimization: A survey of the state of the art. Swarm and Evolutionary Computation, 6:1-24.

Nimwegen van E., Crutchfield J., and Huynen M. (1999). Neutral evolution of mutational robustness. Proc. Natl. Acad. Sci. 96, 9716–9720.

Nolfi S. and Floreano D. (1999) Learning and evolution. Autonomous Robots, 7: 89–113

Nolfi S. , and Floreano D. (2000). Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. Cambridge, MA: MIT Press/Bradford Books

Nolfi S., Bongard J., Husband P, and Floreano D (2016). Evolutionary Robotics, in Siciliano B., Oussama Khatib (eds.), Handbook of Robotics II Edition, Berlin: Springer Verlag.

Nolfi S., Floreano D., Miglino O. & Mondada F. (1994). How to evolve autonomous robots: different approaches in evolutionary robotics. In R. Brooks and P. Maes (Eds.), Proceedings of the International Conference Artificial Life IV. Cambridge Mass: MIT Press,

O'Donnell D. R., Parigi A., Fish J.A., Dworkin I, and Wagner A.P. (2014). The roles of standing genetic variation and evolutionary history in determining the evolvability of anti-predator strategies. PloS ONE, 9 (6): e100163.

Pagie L. and Hogeweg P. (1997). Evolutionary consequences of coevolving targets. Evolutionary Computation 5(4), 401–418.

Pagliuca P., Milano N. and Nolfi S. (2018). Maximizing adaptive power in neuroevolution. PLoS ONE 13(7): e0198788.

Palmer M.E. and Feldman M.W. (2011). Spatial environmental variation can select for evolvability. Evolution, 65 (8): 2345-2356.

Pigliucci M. (2008). Is evolvability evolvable? Nat. Rev. Genet., 9:75–82.

Poli R., Langdon W.B., Phee McN.F., A field guide to genetic programming (2008). Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk. Accessed Apr 2019

Raman K. and Wagner A. (2011). The evolvability of programmable hardware. Journal of The Royal Society Interface 8 (55): 269–81.

Rana S., Whitlev L.D., and Cogswell R. (1996). Searching in the presence of noise. In H. M. Voigt (Ed.), Parallel Problem Solving from Nature. Lecture Notes in Computer Sciences, 1141:198–207. Berlin: Springer-Verlag.

Rechenberg I. (1973). Evolutionstrategie—Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Stuggart: Frommann-Holzboog.

Schumer M. and Steiglitz K. (1968). Adaptive step size random search. IEEE Transactions on Automatic Control, 13:270–276.

Schuster P., Fontana W., Stadler P.F. and Hofacker I.L. (1994) From sequences to shapes and back: a case study in RNA secondary structures. Proceedings Royal Society London B 255: 279–284.

Sebald A.V. and Fogel D.E. (1992). Design of fault tolerant neural networks for pattern classification. In D.B. Fogel and W. Atmar (Eds.) Firstst Annual Conference on Evolutionary Programming. San Diego: Evolutionary Programming Society.

Sekanina, L. (2004). Evolvable computing by means of evolvable components. *Natural Computing*, *3*(3), 253-292.

Sgro C.M., and Hoffmann A.A. (2004). Genetic correlations, tradeoffs and environmental variation. Heredity 93: 241-248.

Silva, S., & Costa, E. (2009). Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. Genetic Programming and Evolvable Machines, 10(2), 141-179.

Sniegowski P.D. and Murphy H.A. (2006). Evolvability. Current Biology, 16: 831–834.

Szafraniec K., Borts R., Korona R. (2001). Environmental stress and mutational load in diploid strains of the yeast Saccharomyces cerevisiae. Proceedings of the National Academy of Science USA 98:1107–1112.

Thompson A. and Layzell P. (2002). Evolution of robustness in an electronic design. In: Miller J.,

Thompson A., Thomson P., Fogarty T.C. (eds) Evolvable Systems: From Biology to Hardware. ICES 2000. Lecture Notes in Computer Science, vol 1801. Springer, Berlin, Heidelberg.

Thompson A., Layzell P. and Zebulum R. (1999). Explorations in design space: Unconventional electronics design through artificial evolution. IEEE Transactions on Evolutionary Computation 3 (3): 167–96.

Tjornfelt-Jensen M. and Hansen T.K. (1999). Robust solutions to job shop problems. In Proceeding of the Congress on Evolutionary Computation, (2): 1138-1144. IEEE Press.

Tononi G., Sporns O., and Edelman G.M. (1999). Measures of degeneracy and redundancy in biological networks. Proceedings of the National Academy of Science USA, 96: 3257–3262.

Turner, A. J., & Miller, J. F. (2014, April). Cartesian Genetic Programming: Why No Bloat?. In European Conference on Genetic Programming (pp. 222-233). Springer, Berlin, Heidelberg.

Tuner J.T. and Miller J.F. (2015). Neutral genetic drift: an investigation using Cartesian Genetic Programming. Genetic Programming and Evolvable Machines, (16) 4: 531-558.

Turner A.J. and Miller J.F. (2014). Recurrent cartesian genetic programming. In 13th International Conference on Parallel Problem Solving from Nature (PPSN 2014), LNCS, vol. 8672. Berlin: Springer Velag, pp. 476–486

Van Nimwegen E., Crutchfield J. P., Huynen M. (1999). Neutral evolution of mutational robustness. PNAS 96:9716–9720.

Vieira C., Pasyukova E.G., Zeng Z.B., Hackett J.B., Lyman R.F., and Mackay T.F.C. (2000). Genotype–environment interaction for quantitative trait loci affecting life span in Drosophila melanogaster. Genetics 154: 213–227.

Wagner A. (2008). Neutralism and selectionism: a network-based reconciliation. Nature Review Genetics: 9: 965-974.

Wagner A. (2008). Robustness and evolvability: a paradox resolved. Proceeding of the Royal Society B, 275: 91-100.

Wagner A. (2011). The Origins of Evolutionary Innovations: A Theory of Transformative Change in Living Systems. Oxford, U.K.: Oxford University Press.

Wagner A. (2011). The Origins of Evolutionary Innovations: A Theory of Transformative Change in Living Systems. Oxford, U.K.: Oxford University Press.

Wagner G.P., Altenberg L. (1996). Perspective: Complex adaptations and the evolution of evolvability. Evolution, 50:967–976.

Wagner G.P., Altenberg L. (1996). Perspective: Complex adaptations and the evolution of evolvability. Evolution, 50: 967–976.

West-Eberhard M. J. (2003). Developmental Plasticity and Evolution. New York, U.S.A: Oxford University Press.

Whitacre J.M. (2010). Degeneracy: a link between evolvability, robustness and complexity in biological systems. Theoretical Biology and Medical Modelling, 7:6

Whitley D. (2001). An overview of evolutionary algorithms: practical issues and common pitfalls. Information and Software Technology, 43: 817-831.

Whitley D., Rana S., and Heckendorn R.B. (1998). The island model genetic algorithm: On separability, population size and convergence. Journal of Computing and Information Technology, 7:33–47.

Wieland, A. (1991). Evolving controls for unstable systems. In Proceedings of the International Joint Conference on Neural Networks, Volume II, Seattle, WA, USA, pp. 667–673. IEEE Press.

Wierstra D., Schaul, T., Glasmachers T, Sun Y. and Schmidhuber, J. (2014). Natural evolution strategies. Journal of Machine Learning Research 15: 949-980

Wilke C.O. (2001). Adaptive evolution on neutral networks. Bullettin of Mathematical Biology, 63: 715-730.

Wilson D.G., Cussat-Blanc S., Luga H. and Miller J.F. (2018). Evolving simple programs for playing Atari games. arXiv preprint arXiv:1806.05695

Zhou Z-H. and Liu X-Y. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Transactions on Knowledge and Data Engineering, (18) 1: 63-77.