Alma Mater Studiorum – Università di Bologna

Campus di Cesena

Ph.D. in Computer Science and Engineering

38th cycle

Online adaptation of autonomous robots

Supervisor:

Prof. Andrea Roli

Presended by:

Paolo Baldini

Academic year 2024–2025

Abstract

Robots diffusion is constantly increasing, yet their applications remain largely confined to industries and other highly controlled environments; the cause is their inability to address unexpected situations that can occur in the open world. We argue that endowing robots with the ability to adjust their own behavior can improve their operational efficacy in uncontrolled environments. In this dissertation, we investigate factors enabling the runtime modification of robot behaviors. The proposed operative approach consists in the systematic investigation and resolution of increasingly complex conditions. During the exploration, we employ robots with minimal computational capabilities, so as to investigate the factors enabling adaptation without them being obscured by the complexity of the control system employed; this also permits providing adaptive capabilities to simple robots, which might not be able to operate complex control software.

The principal contributions of this dissertation include the identification of factors essential for adapting systems, among which stand out the need for self-evaluation, re-evaluation, and time- and intensity-modulated adaptation. Additionally, we highlight the importance of well-devised evaluation criteria and of their inherent characteristics, mainly the evaluation time and whether they tend to induce the emergence of a specific behavior or permit a broad spectrum of outcomes. We propose adaptive mechanisms capable of operating on various robot control architectures, including one designed offline by an automatic system with the explicit intent of optimizing its runtime adaptive capabilities. Finally, we present, define, and propose solutions to the problem of "situationality", a condition in which external factors determine the performance of the robot regardless of its capabilities.

Overall, this dissertation aims to contribute to the literature on online adaptation in robotics, shedding light on some frequently underestimated issues. We expect that the proposed solutions and considerations will support the future advancement of this discipline.

Keywords: robotics, online adaptation, autonomous robots, minimal cognitive systems.

Acknowledgements

Writing this dissertation has been a long and difficult process that I could not have completed without the assistance and encouragement of many individuals in my life. Consequently, I would like to devote this section to expressing my gratitude to those who provided unwavering support.

First and foremost, I would like to thank Andrea Roli, my Ph.D. supervisor, for his guidance over the years. His mentorship and support have been essential to the progress of this dissertation as well as to my own personal and professional growth; I owe him a great deal.

The second person I would like to thank is Michele Braccini, who escorted me during my journey and helped me to approach it pragmatically. He has played a crucial role in concretizing acquired theoretical knowledge into publishable research contributions, all while making the process enjoyable with his sense of humor.

Following, I would like to express my gratitude to Mauro Birattari, who supervised me during my stay in Belgium. He introduced me to an exceptional research environment, showing what a large and cohesive group aligned toward common objectives can achieve.

I would also like to thank all the colleagues I met during this journey, whose presence greatly enriched the experience. Special mentions go to Alberto Franzin, Giuseppe Antonio Patarino, and Guillermo Legarda Herranz for their friendship during my time abroad: you made that period fly! A thank also goes to Gianluca Aguzzi and Samuele Burattini for their friendliness and the frequent help in navigating the complex bureaucracy of the university.

Beyond professional connections, I would like to thank all my family for their support over the years. In particular, I would like to thank my wife, Büşra Oktar, for her encouragement and

for ensuring I maintain a healthy balance between work and life. I also express gratitude to my mother, Nicoletta Pincio, and my aunt, Sabrina Pincio, for their dedicated efforts in facilitating my daily life. In closing, and in the hope that this will not cause any offence, I would like to express my gratitude to my dog, Sole, who is always capable of making me smile, no matter how bad the situation.

Finally, I would like to thank my friends for helping me find the necessary respite from the seemingly endless workload. Especially, I would like to thank Filippo Suzzi, Matteo Graffieti, Mattia Sassano, and Pawel Striz. Special mention goes to Alessandra Bonoli for the friendship and the company kept while in Bruxelles.

Contents

Introduction

Complex machines able to operate in the world with minimal human supervision: this is what most people think about after hearing the word "robot". This picture originates from science fiction novels of the previous century, which depicted (semi-)humanoid artifacts coexisting alongside humans in everyday life. Nonetheless, current robotics falls short of this definition: robots are generally complex machines engineered for highly specific and constrained operational conditions. The primary example of their successful application is indeed in the industrial sector, where the conditions can be precisely controlled and the missions arranged in a clear sequence of tasks; in all the remaining physical operational domains, humans still remain virtually necessary. Nevertheless, a change may be in progress, driven by the artificial intelligence promise to create systems capable of operating through intrinsic knowledge acquisition and learning. The long-term goal is to finally produce entities that perceive, understand, and act autonomously, that is, without human supervision. Achieving this necessitates robots that exhibit flexibility in the face of unexpected changes occurring in the world around them.

Despite technological advancements, however, most robotic solutions still consider only static design-time conditions, meaning that the system is optimized for performance under the circumstances present during its creation. This approach has several limitations. First, the conditions considered at design time must be representative of those encountered during operation. Second, the operational conditions must remain overall constant. Failing to satisfy these requisites introduces significant risks of underperformance during operation. For instance, a robot might be trained to perform a mission in an environment containing specific perceptual cues; if said cues are absent or differ significantly in the operational setting, the ability of the robot

to complete the mission may be compromised. Similarly, if the cues exist but disappear during operation, the system would face the same type of risk. One could then argue that robots are intended to work solely in the environment for which they are explicitly designed. Nevertheless, this assertion contains two crucial fallacies: (i) we employ robots in specific known contexts because of necessity, not because it is necessarily the optimal course of action, and (ii) unexpected changes are not limited to the environment, but also extend to the robot itself. The degradation of perception and action due to consumption and age clearly exemplifies this latter scenario. In such cases, the ideal outcome is for the robot to sustain its operational functionality and to prevent further damage by adjusting its control strategy; nevertheless, the typical result is a progressive decrease in performance culminating in the early failure of the assigned task.

To address the variety of events that may occur during operation, the robotics community has recently begun investigating mechanisms to enable the runtime modification of the control strategy. The goal is to dynamically alter the robot behavior to better fit the operational conditions. With reference to the earlier example of missing environmental cues, the robot could find an alternative strategy to complete the mission despite the missing signals. Similarly, with reference to the second example of degraded hardware, the robot could simply act more carefully or perceive for a longer period so as to mitigate noise. Furthermore, the ability to adapt can even foster the creation of novel and more effective operative strategies, for instance, exploiting tools to complete tasks more efficiently. These examples clearly highlight the potential effectiveness of robots capable of situational adaptation, a capability that would greatly facilitate their proliferation in everyday life. Nevertheless, despite their potential, these mechanisms are still in their infancy and thus necessitate further research and development to achieve the robustness required for real-world implementation.

To conclude, while current robotics fulfills many human requirements, its widespread integration into everyday life still requires an essential addition: the ability to adapt. This lies at the end of a path strewn with challenges, which we nevertheless believe must be addressed in order to advance the field. In this dissertation, we will proceed towards this goal by beginning an exploration into online adaptation, a domain that has been initiated but not yet fully elucidated. Specifically, we will investigate the factors enabling embodied adaptation and its limitations. Throughout, we will continue to propose new strategies to address them, focusing

on the generalizability of the approaches. To these ends, the investigations will employ robots with minimal computational capabilities. This serves a dual goal: on the one hand, it prevents these enabling factors from being obscured by the complexity and power of sophisticated computational systems; on the other hand, it enables the use of the proposed methodologies even in computationally limited robots.

Research questions

Having discussed the context and the general goals of this work, we will now proceed to detail the corresponding research questions; these should help the reader navigate this dissertation and rationalize the logical sequence of empirical investigations performed.

RQ1 *What operational conditions and properties are essential to embodied online adaptation? What hinders its efficacy?*

Identifying the operational conditions and the properties of the adaptation mechanism that enable the robot to adapt is of primary importance and a key goal of this dissertation. Conversely, identifying the factors that complicate and limit its applicability permits to avoid them whenever possible and to focus research endeavors towards their solution.

RQ2 *How to assess online adaptation mechanisms?*

Robots adapting online may have additional constraints during the adaptation process, and, therefore, they may require different evaluation approaches than those commonly used in robotics. It is thus important to discuss how to properly evaluate such mechanisms.

RQ3 *What are the abstract, general characteristics that enable the adaptation of diverse control architectures?*

In this dissertation, we are interested in identifying general characteristics of adaptation mechanisms that enable adaptation regardless of the control system employed. The goal is to provide a body of knowledge that can be easily generalized to different control systems.

RQ4 *Can robots adapt to changes in internal and external conditions?*

The ability to adapt to changes is the primary characteristic supporting the development of adaptive mechanisms in robots; therefore, understanding which types of situations it permits to overcome is a fundamental aspect of this investigation. Specifically, we consider the influence of internal and external changes, where the former include damages and the latter environmental variations.

RQ5 *Is it possible to co-adapt different robots to collaborate? What does it require?*

The adapting robots of the future will likely have to operate in groups; this requires comprehending the difficulties that this adaptation context presents and how we can overcome them. Therefore, this dissertation will consider the problem of co-adaptation as a representative investigation scenario.

RQ6 *How can we combine online adaptation with currently existing design techniques?*

Online adaptation is not intended as a standalone approach but rather as an additional mechanism to improve the robots capabilities. Therefore, herein, we investigate its combination with an offline design approach as a representative investigation scenario.

Structure of the dissertation

This dissertation investigates the use of online adaptation methodologies in robotics. It commences by establishing the fundamental principles of adaptation in Chapter 1. This part covers the biological facets of natural adaptation alongside their underlying enabling mechanisms. Subsequently, the focus shifts to a thorough review of (semi-)automatic design approaches in robotics, encompassing both classical design methodologies operating before deployment and novel mechanisms enabling adaptation.

After laying the scientific groundwork in the background chapter, the dissertation proceeds to the experimental part, detailing the investigations conducted to move towards the desired goal. Specifically, Chapter 2 investigates the efficacy of an adaptive mechanism devised for minimal robots operating in static environmental conditions. The aim is twofold: (i) determine whether the proposed mechanism possesses the capacity to enable online adaptation by testing its ability to discover effective behaviors during runtime, and (ii) utilize the experiment as a

test-ground for selecting the most appropriate evaluation metrics. Following, Chapter 3 begins investigating online adaptation to address a problem truly necessitating the runtime variation of the control strategy: internal system faults. This application context entails increased complexity, necessitating particular operational characteristics for successful execution. Chapter 4 investigates, instead, external changes by assessing adaptation within a dynamic environment; the goal is to understand its capability to manage also this type of situation and to identify the required characteristics for success in such scenarios. Chapter 5 presents further investigations on this topic through the proxy of co-adaptation, which can be regarded as the adaptation to a specific instance of rapid external variations: the behavior of other robots. We investigate whether this requires techniques and approaches different from the ones used in the previous chapters. We conclude the experimental part of the dissertation in Chapter 6 by observing the effects of combining offline design and online adaptation. The goal is to investigate whether this combination supports overcoming the individual limitations inherent in each approach.

After the core experimental part of this dissertation, we devote Chapter 7 to summarize the characteristics of the created adaptive controllers and the important parameters and strategies for their functioning. Following, we reserve Chapter 8 to discuss ancillary works by this dissertation author that are related to, but not directly within the scope of, online adaptation and robotics. Subsequently, Chapter 9 discusses ongoing and future research directions that the author of this dissertation intends to pursue, all of which are pertinent to the topic of online adaptation in robotics. Finally, Chapter Conclusion provides a comprehensive synthesis and articulates the final conclusions of this dissertation.

Part I

On the nature of adaptation

1 Background

This chapter discusses the background theory that inspired the investigations presented in this dissertation. It begins by describing the biological concept of adaptation in all its facets, diverting later on to biological-inspired adaptive systems; however, due to the vast amount of literature on the topic, we will focus only on adaptation in robotics. This should help to understand some choices and decisions presented in the next chapters and to justify the logical sequence of empirical investigations detailed therein.

1.1 Biological inspiration

Before adventuring into the realm of robotic adaptation, we devote this section to the broad topic of changes in biological organisms. The goal is to briefly describe the concepts fostering and inspiring our investigations. The main idea is that life is not static, but relies on continuous changes. This permits it to persist despite the chaos revolving in the world around. Indeed, changes are the motor for adaptation, that is, the ability to vary so as to better fit a continuously changing environment. Differently, if organisms or populations remained static, they would likely not survive in the world for long. In order to face different types of unpredictable changes, adaptation operates at different timescales (Lasker, 1969; Kuzawa et al., 2011) affecting organisms in multiple ways (see Figure 1.1). The most famous is probably *evolutionary adaptation*, that affects the characteristics of organisms across generations. In a shorter time, *developmen-*

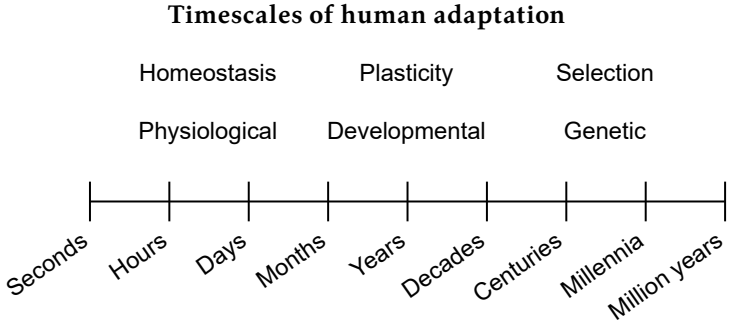


Figure 1.1: Timescales of human adaptation. Of the lines above the scale, the top indicates the adaptation process, while the bottom indicates the type of change.

tal plasticity affects the characteristics of an individual mostly during development. At an even shorter interval, *learning* and *short-term adaptation* enable changing the way organisms operate in different moments of their lifetime. For instance, human populations living on the Andes mountains, Peru, adapted during generations so as to develop a larger thorax and greater lung capacity so as to mitigate the reduced amount of oxygen (Lasker, 1969). Humans growing up there but coming from a different environment were more subject to issues compared to natives, but their bodies adapted during growth to mitigate them. Finally, humans visiting the area presented noticeable difficulties in the first days, requiring a higher heart-beat rate to compensate for the lack of oxygen, and then slightly adapting by increasing hemoglobin levels.

The occurrence of changes at different timescales thus enables increasing the survival rate of individuals and better fitting the environment. Nevertheless, not all changes are necessarily advantageous to the organism: some might be deleterious. Life implicitly manages the retention of changes according to how they affect the fitness of the individual in its environment (Katju et al., 2018). If a change decreases the fit of the organism, it will likely be reverted or eliminated in future generations. If it improves the synergy of the organism with its environment, then the change tends to be fostered and replicated. Finally, if it does not affect the fitness, its persistence will depend on other factors, such as its frequency in a population. Overall, the main factor determining the retention of changes is their utility to the organism. In the following sections, we will discuss adaptation at different timescales and the role of fitness in the process.

1.1.1 Evolutionary adaptation

Discussing long-term adaptation requires introducing *evolutionary adaptation*, which is based on *selection* (G. Williams, 2019). This acts over many generations, and it is the main source of changes in most species. The core idea is that, over multiple generations, organisms begin to differ so as to fit their environment better. What drives this change is the ability to reproduce and procreate, preserving advantageous variations (Darwin, 1859). This means that, in the general framework of evolutionary adaptation, the metric used to understand how useful a change is depends on the reproductive capability of its host. The idea is that disadvantageous variations lead to a sexually unappealing individual or to its premature death. For instance, the thickness of the fur on animals living in cold environments could make the difference between life and death in extreme situations (Blix, 2016). Individuals with thicker fur would thus have higher chances of surviving and reproducing, increasing their ratio among the population and leading to the diffusion of their trait at the expense of the other.

In most organisms, we can identify two processes causing variations with respect to previous generations: *recombination* and *mutation* (Fisher, 1930); both occur during the organism's reproduction. The former requires more than one individual to mate (e.g., sexual reproduction); the latter could theoretically involve just one individual (i.e., asexual reproduction). During *recombination*, the offspring inherits a combination of the parents' genomes. This collage of genetic material permits expressing different combinations of traits, potentially affecting the fitness of the progeny. During *mutation*, the genetic material undergoes random changes that possibly affect the expression of traits and the fitness of the individual. This type of variation is not frequent (Sanjuán et al., 2010), but it can lead to novel characteristics that neither parent had. Most of the time, mutations are deleterious, but sometimes they can also be neutral or advantageous (Nei, 2013). In the former case, the organism presenting them experiences a decrease in fitness that, if too drastic, hinders mating and thus the propagation of the mutation. In the latter cases, the increase in fitness facilitates the propagation of the mutation to offspring.

The described process of evolutionary adaptation is the most common; nevertheless, some organisms use different strategies. For instance, in asexual replicating organisms, changes happen mainly thanks to mutation (Meeûs et al., 2007). This is because the lack of mating pre-

vents the occurrence of recombination. Interestingly, asexual bacteria replace this mechanism by exchanging genetic material with other individuals in a process known as *horizontal transmission* (Bell, 2007). The result is that a relatively large amount of their genome comes from different evolutionary lines, increasing the diversity within the species.

1.1.2 Exaptation

Until now, we assumed that a trait is selected if it performs a function that is advantageous in the specific environment in which the organism operates. This is overall true, but it is not the only way it can happen. A trait can evolve for a specific purpose but end up being useful for another function later. In this case, talking about selection would be misleading, as it would suggest that the trait has been selected for that specific role. To solve this semantic issue, Gould et al. (1982) propose the term *exaptation*. The aim is to more clearly refer to a trait that was adapted for a specific purpose and then co-opted for a new, different one. An example is that of birds' feathers, that were initially selected for thermal control and only later became a tool enabling flight. This idea breaks the traditional view of evolutionary adaptation as a linear process, showing that changes depend on a long evolutionary history that could select traits for unimagined reasons.

1.1.3 Developmental plasticity

Genetic selection operates over many generations. This implies that it cannot directly respond to overly rapid changes in the environment or in external conditions such as those occurring during the individual's lifetime. Therefore, life developed other strategies to attain fast adaptation. For long-term changes, *developmental plasticity* is the reference process (Kuzawa et al., 2011). This affects the characteristics of an individual (i.e., its phenotype) so that they do not only depend on the genotype, but also on the specific conditions encountered during growth. For instance, plastic phenotypes have been shown to depend on the level of stress of mothers before birth (Crespi et al., 2005; Kuzawa et al., 2011). Meanwhile, colony insects specialize in their role according to nutrition (Hartfelder et al., 1998). This process guarantees some flexibility to the individual, permitting to fit to a variety of conditions and increasing its (and its

population) chances of survival. Nevertheless, this flexibility comes at a cost; indeed, plasticity requires additional complexity and energy to operate (DeWitt et al., 1998). The genome must encompass the possibility for the trait to exhibit plasticity for it to actually occur (Bradshaw, 1965; Lema, 2008). This is supposedly made possible by switch genes, that sense the environment and then control the expression of alternative phenotypes (Sommer, 2020). For this motivation, individuals or species usually display a limited number of plastic traits that develop when several conditions are met (Ghalambor et al., 2010). Specifically, the populations must be exposed to variable environments easily identifiable by environmental cues, and each environment should favor a specific phenotype, with no phenotype exhibiting a good fit globally.

Developmental plasticity is thus a powerful tool permitting organisms to adapt to conditions present during their lifetime. Nevertheless, as for evolutionary traits, it is not guaranteed that all these environmentally-driven changes are advantageous: sometimes a change can also be disadvantageous (Ghalambor et al., 2007). To discriminate between these two cases, biologists often refer to *adaptive* and *maladaptive* traits. The former refers to positive changes, while the latter refers to negative changes that can also have been induced by factors such as stress[1]. The risk of maladaptation is the result of the way plastic processes operate. As previously said, the genome itself enables plasticity to a varying degree; however, if the adaptive process is flawed to begin with, the result might be disadvantageous. In these cases, we expect evolutionary adaptation to take over, filtering out genes causing maladaptation in subsequent generations. The result is that, usually, only adaptive plasticity is present with relatively high frequency.

The phenotype on which plasticity operates can be morphological and physiological, with some studies even including behavioral (Sommer, 2020). Plasticity operating on *morphology* modifies the visible aspects of the organism. For instance, Levis et al. (2018) report dietary-induced morphological changes in tadpoles. When fed with shrimps, the subjects of the experiment developed larger jaw muscles, more notched mouthparts, and shorter guts. Instead, when fed with detritus, they developed more denticle rows. Another example is the ability of some aquatic invertebrates to alter their body shape based on the flow of water, developing a more streamlined shape in fast-moving currents to reduce drag (Langerhans, 2008). This latter example, however, is only partially ascribable to phenotypic plasticity, with part of the vari-

[1] In this case, we refer to this kind of change as *nonadaptive* (Sommer, 2020).

ation due to genetic differentiation (i.e., the hereditary differences between populations of a species). Meanwhile, Lema (2008) reports that changes in temperature and nutrients induce changes in the morphology (particularly in the size) and in the development time of the individuals involved in the study (i.e., Amargosa River pupfish). Phenotypic plasticity can also modify the *physiology* of individuals, that is, the way the organism works. For instance, Carroll (2003) reports that prolonged pre- and post-natal exposure to elements such as drugs or toxins can affect the long-term structure or function of the respiratory control neural network. Additionally, nutrition and stress during growth influence the adult risk of developing cardiovascular diseases (Barker, 1995). Finally, developmental plasticity is also sometimes associated with changes in the *behavior*. For instance, it has been reported that drug consumption during development can affect brain plasticity and behavior in adulthood (Kolb et al., 2011). Alternatively, Lema (2008) observes that temperature and salinity in the environment influence the aggressiveness of pupfish. Nevertheless, it is also common to ascribe changes in the behavior of humans and animals to learning, which is more dependent on experience than developmental plasticity operating on the behavior (Galván, 2010). The different terminology for referring to the underlying process of behavioral adaptation stems from an ongoing debate about the use of "phenotypic plasticity", that is sometimes regarded as a process causing only permanent (i.e., non-reversible) changes (Ghalambor et al., 2010). For this reason, later we will also discuss behavioral changes from the point of view of learning, which can operate even on shorter timescales.

1.1.4 Homeostasis

Developmental plasticity enables organisms to adapt to variable conditions after birth; nevertheless, it is not useful in the case of fast changes occurring during life. For this motivation, organisms use a different process to maintain their physiologic parameters in a satisfactory range: *homeostasis* (Cannon, 1929). This process regulates the physiological reaction of the organism so as to tackle perturbations. An example is that of the core-body temperature, that can be maintained in a survival range with processes such as vasoconstriction and vasodilation. The former reduces thermal loss by reducing the blood-flow to peripheral areas of the body;

the latter reduces internal temperature by increasing the blood-flow to skin, thus increasing its thermal dispersion (Benzinger, 1969). Obviously, the thermal homeostatic process is much more complex, including metabolic changes, sweating, hormonal regulation, etc.; still, it is a clear example of biological self-regulation.

An important characteristic of homeostasis is that it is reversible, meaning that the changes it induces are not permanent (Kuzawa et al., 2011). This permits the flexible response to perturbations by temporarily changing the organism's functioning. Nevertheless, this comes at a cost: the process requires energy and can put the body under stress. For instance, the modulation of the heartbeat is part of many homeostatic processes. However, peculiar conditions could concurrently require low and high heartbeats. This could be impossible to achieve, requiring the modification of other parameters. In the same way, the frequency at which the heart can operate is limited, meaning that regulation in extreme situations might not be possible. Kuzawa et al. (2011) propose the example of increasing the heartbeat to overcome the lack of oxygen at high elevation; in this situation, further increasing the heartbeat to increase the performance while escaping from a predator might not be possible. The body is thus limited in its self-regulatory processes, risking reaching its limit when multiple challenges are at stake, and possibly not being able to sustain them for long.

Finally, as for developmental plasticity, homeostasis is also enabled by the genome, which must codify the structures enabling this adaptive process (Rothman et al., 2021).

1.1.5 Learning

Until now, we have discussed adaptive changes operating on limited boundaries and occurring during the lifetime of individuals. For instance, developmental plasticity usually has a limited number of possible physical outcomes that depend on the genome. Similarly, homeostasis regulates physiological parameters in a limited, suitable range of values. However, these types of changes are not able to produce a suitable response to every possible variation; actually, they tackle a limited set of scenarios. To enable a greater level of survivability, intelligent organisms can vary their behavior so as to overcome a broader range of issues. They do so by changing their mental representations and associations according to experience. This process is usually

called *behavioral plasticity* or *learning*.

Intelligent living beings are able to learn thanks to their complex and plastic brain (Jäncke, 2009). The idea is that a huge number of neurons interact so as to produce a response to specific situations. This interaction is made possible by means of synapses, that interconnect different neurons. Learning is believed to be mainly the formation, strengthening, or weakening of these connections (Hebb, 1949). Neurons produce axons that grow to connect to other neurons across the brain. When an axon encounters another neuron, it forms a connection. This connection can then strengthen if the synapses are stimulated at high frequency for a brief time, or due to the pairing of presynaptic activity with postsynaptic depolarization (Martin et al., 2000). Differently, the connection can weaken if it is stimulated at low frequency for long periods (Purves et al., 2012). Eventually, weak synapses can be pruned so as to improve the computational efficiency of the brain without altering its functioning (Chechik et al., 1999). This main learning process is not, however, the only possible one. Recently, researchers discovered that *neurogenesis* (i.e., the creation of new neurons), which was thought to terminate during development, continues instead in some part of the brain, supporting synaptic plasticity in learning (Ormrod, 2018).

1.2 Robotics

Most technology in the current world is heavily inspired by natural solutions; indeed, eons of evolution have permitted to identify effective strategies to face a broad spectrum of problems and difficulties that we encounter also during the design of artificial systems. This is especially true for robotics, that operates in the same environment as other biological beings, and could thus benefit from inspiration when considering morphology and control. Nevertheless, in this section, we will focus on biologically inspired *approaches* to the automatic design of robots rather than discussing biologically inspired solutions to problems. Specifically, we focus on control software design and adaptation techniques, which somehow reproduce evolutionary and learning processes in silico. Mainly, we can distinguish two macro-categories of approaches (Eiben et al., 2010a): (i) *offline*, acting before the robot deployment, and (ii) *online*, acting during its operation. In the literature, this distinction is often nuanced; nevertheless, we

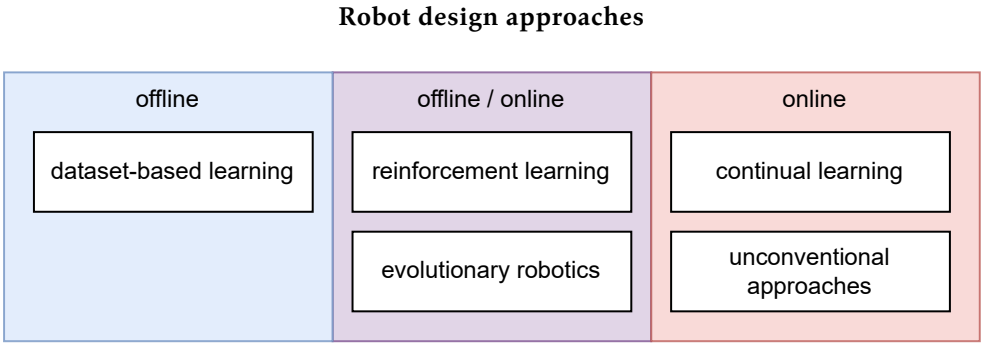


Figure 1.2: Automatic approaches to the design of robot morphology and control.

can notice that the former generally relies on external systems that are then severed away from the working robot once deployed (or at least, they are not supposed to be used further). Instead, in the latter, the adaptive process is a core part of the system and continues operating at need during the lifetime of the robot (or at least, it theoretically could). In the following sections, we will discuss prominent examples of both approaches (see Figure 1.2).

During our dive into design approaches for robotics, we will often distinguish between single- and multi-robot systems. Those are two different engineering fields that, although sharing some common characteristics, exhibit significant differences. Operationally, the latter leverages distributed computation and capabilities to address a broader set of missions with respect to the former (Parker et al., 2016); specifically, it permits addressing distributed tasks that are inherently impossible for single robots. This advantage comes, however, at a cost: designing multi-robot systems requires considering the complex dynamics that emerge from the interaction of many agents, which notably complicates the process. Engineering single-robot systems is thus typically considered easier than engineering multi-robot ones (Parker et al., 2016), leading to their faster adoption in production environments. Nonetheless, also some types of multi-robot systems are employed in real-world scenarios, generally, those characterized by a centralized architecture, which simplifies deployment and control (decentralized systems such as swarms of autonomous robots are instead still rarely used in production; Schranz et al. (2020)). In the following sections, we will discuss designing approaches for single- and multi-robot systems.

1.2.1 Offline approaches

Offline approaches to robot design have been applied both to control software and body morphology. Mainly, we can distinguish two different strategies: (i) *evolutionary-*, and (ii) *learning*-based. The former draws inspiration from biological evolution, testing and combining populations of solutions to discover one that properly fits the environment (i.e., produces the desired outcome). The latter produces a desired output by learning input-output mappings[2]. In the following sections, we explore the uses of these two approaches, presenting notable experimental works employing them. Additionally, we discuss a few more strategies that do not fall under these two classes.

Evolutionary robotics

Evolutionary robotics is a prominent approach for the design of robot control software and, as we will see, morphology. This leverages the evaluation of multiple solutions so as to identify the ones possessing the characteristics useful for the robot to succeed in the selected task. Then, it proceeds by combining (i.e., crossover) and varying (i.e., mutation) all these selected solutions with the aim to find a more effective one. The process is clearly inspired by evolutionary adaptation, in which more fit individuals reproduce and less fit die. In the long term, the expected result is offspring behaving better than their parents and ancestors. The process stops when a desired threshold of performance is reached or when a search budget (e.g., time or iterations) is consumed. In the macro-context of offline design and learning, evolutionary robotics has been applied both to the design of single- or multi-robot systems. In this section, we present works approaching both these different research tracks.

When it comes to individual robots, evolutionary robotics has mostly been used as an unbiased search process[3] potentially producing solutions that perform better than those generated by humans (Doncieux et al., 2015). Among the first works on the topic, there is that of Dorigo et al. (1991), in which the authors propose genetic programming (Holland, 1992) to automatically generate robot control strategies by combining low-level behaviors. Nevertheless, the authors do

[2]Input-output mappings can be passed explicitly to the system or learned implicitly.

[3]Technically, human biases can influence even evolutionary robotics, for instance, when devising evaluation functions; nevertheless, the bias is surely reduced during the design itself.

not test the said process, but rather discuss its potential applicability. In its discussion, Brooks (1992) proposes something similar: a system able to organize multiple low-level behaviors by means of genetic programming applied to custom code; nevertheless, he also does not provide experimental results in his work. Meanwhile, Lewis et al. (1992) work on using genetic programming to create a neural network based controller able to make a legged robot walk. The generated motion is quite simple, consisting only of forward and backward movements, but it is the first example of a functioning controller created by means of genetic programming that the author of this dissertation could find. After this preliminary work, also Cliff et al. (1993) proposed to evolve a neural network to control a robot; specifically, one with recurrent architecture. Nevertheless, also in this case, the task that they select to test the system is rather simple: moving to the center of an arena and staying there. Just one year later, however, Miglino et al. (1994) propose a more advanced wandering behavior by using evolved neural networks. From there on, many works started to investigate evolutionary robotics in more complex tasks. For instance, in Nolfi et al. (1995) and Nolfi (1997), the authors propose the evolution of a feedforward neural network controller for the localization, identification, and grasping of a target. Meanwhile, Jakobi et al. (1995) start to notice that controllers evolved in simulation present issues when used in the real world. The cause is the difference between the simulated and real world, with the controller overfitting the former; since the result is a performance drop when deployed to the real robot, the authors call this problem *reality gap*. From that moment on, tackling the reality gap will be the target of a significant research effort from the robotics community.

After these foundational investigations, evolutionary robotics continued to develop, with many works trying to investigate its application in different areas. As the literature in the field is too broad to be fully described, following, we just discuss those investigations providing novel perspectives or approaches. For instance, in Nolfi et al. (2002), the authors describe the evolution of feedforward and recurrent neural networks to visually discriminate and reach the largest of two targets. Interestingly, the authors report the emergence of different strategies in static and variable environments, with the latter being characterized by targets of different sizes and positions. This highlights a level of specialization that is typically attributed to evolutionary robotics, but at the time of the study, this implication had not yet been explic-

itly described. Slightly later, Bianco et al. (2004a) move away from the use of movable robots, investigating the use of evolutionary robotics to develop the controller of a robotic arm that pushes objects. This work proposes an alternative to the classical inverse-kinematic approach commonly used to control robotic arms, showing that evolutionary robotics could be effective even in such cases. In 2005, Harding et al. propose an interesting application of evolutionary strategies in robotics: modifying the computational properties of a hardware component used to control a robot according to some inputs (Harding et al., 2005); the authors call this approach *evolution in materio*. In Lipson et al. (2006), the authors make a comprehensive discussion about the difficulties of crossing the reality gap, proposing several alternatives. Mainly, they propose using (i) noisy simulator, (ii) simulators evolved to mimic the real-robot peculiarities, (iii) plastic controllers. Among those, the former two are offline approaches, while the latter operates online. Taking a further inspiration from biology, some works started to consider the use of evolved gene regulatory network models to control robots (Quick et al., 2003; Roli et al., 2011). Those simulate the structures that determine gene expression and thus the functioning of cells (Karlebach et al., 2008). These works demonstrate that these biologically inspired models can be successfully used to control robots, with the further advantage of being computationally simple to implement and operate. More recently, Nygaard et al. (2018) use multi-objective evolutionary optimization to optimize morphology and control of a four-legged robot in different conditions. However, the considered changes in morphology are limited to minor adjustments in component lengths. Husbands et al. (2021) report the evolution of hardware circuits based on FP*A (i.e., Field-Programmable Gate / Transistor / Analog Array) for controlling robots, enabling leveraging even the potential of analog processing. Furthermore, in the same article, the authors report recent investigations on the use of chaotic systems[4] for robot control. Nadizar et al. (2021) and Nadizar et al. (2022) imitate biological pruning of neural connections with the aim of reducing the complexity of evolved artificial neural networks used to control soft-robots. They find that the effectiveness of pruned individuals does not decrease, while the complexity of their networks obviously does. Some of these authors also investigate the difference in performance while using different types of biologically inspired neural networks: perceptron, recurrent, and spiking (Nadizar et al., 2023). They find that recurrent neural networks usually

[4] A chaotic system is characterized by a dynamic that is unpredictable in the long term.

attain higher performance, but spiking neural networks sometimes generalize more. Recently, the field of evolutionary robotics has enlarged to encompass even aquatic and aerial robots. For instance, Corucci et al. (2018) investigate the evolution of rigid and soft robots for aquatic environments and the possibility of evolving a morphology and control enabling operations in both. Meanwhile, Bergonti et al. (2024) propose evolving morphology and control of a drone, resulting in an optimized number of active joints and energy consumption.

After the first single-robot investigations, the research community of evolutionary robotics started to consider also the automatic creation of control software for groups of robots. This specific field of investigation is usually regarded as particularly complex, having to organize and coordinate multiple robots to complete a common mission. Therefore, the use of automatic design approaches has been seen as a potentially powerful way to explore the emergence of more effective coordination than that achievable by human programmers. Among the first works investigating automatic robot controllers design by means of evolutionary approaches are those of Quinn (2000) and Quinn (2001), in which the authors compare two different approaches for the evolution of control software for a co-operative task: (i) shared controller among robots (i.e., homogeneous), and (ii) unique controller per robot (i.e., heterogeneous). While they found that the latter performs better, it must be noted that their task favored the emergence of two different roles among the robots. Later, different works started considering the evolution of neural network controllers for a variety of different tasks: aggregation (Trianni et al., 2003), co-operative transportation (Groß et al., 2004), coordinated movement (Quinn et al., 2003; Baldassarre et al., 2003a; Baldassarre et al., 2003b), etc. Notable is the work of Tuci et al. (2008), that coordinated movement in a group of physically heterogeneous robots communicating via an evolved acoustic signals protocol. Nevertheless, all these early works considered only small groups of robots: generally fewer than five. The question arises, then, as to whether these techniques also work for larger groups of robots. Dorigo et al. (2004) try to answer this question by investigating the scalability of controllers designed by means of evolutionary robotics so as to perform coordinated motion. They observe that the evolved controllers work even with larger groups with respect to the ones considered during design, confirming the potential of these techniques. Recently, evolutionary robotics applied to swarms and groups of robots has advanced rapidly. Many works considered the problem of collective decision-making (Morlino et al., 2012; Almansoori et al.,

2024). Duarte et al. (2016) propose evolved aquatic swarms for homing, dispersion, clustering, and area monitoring. Others considered the evolution of controllers different from neural networks, such as behavior trees (Jones et al., 2018). In this line of work, Ligot et al. (2020) propose the evolution of neural networks to be used as modules in finite state machine controllers. Meanwhile, Gomes et al. (2018) evolve a repertoire of task-agnostic controllers using quality diversity evolutionary algorithms, showing that, among them, lie behaviors with performance comparable to that of behaviors optimized for specific missions. Even heterogeneous swarms have been proposed; for instance, Gomes et al. (2019) evolve co-operation between an aerial and a ground robot to complete a simulated foraging task.

Evolutionary approaches have also been used to evolve the morphology of agents. One of the first works on this field has been that of Sims (1994), that co-evolved morphology and control of simulated agents. However, in order to see the first physical implementations of similar entities, we have to wait until 2000, when Lipson et al. (2000) present a system able to design the morphology and the controller of a robot in simulation so as to then implement it in the real world. Later, in Hiller et al. (2012), the authors expand the investigations so as to produce soft-robots, discussing the complexity of this particular type of co-evolution (Cheney et al., 2016). Haller et al. (2005) co-evolve morphology and control of an underwater robot in simulation so as to increase its effectiveness. More recently, it is notable the work of Kriegman et al. (2020), in which the authors take a step further, recreating the evolved morphologies with biological cells. Other interesting works on the topic are J.C. Bongard et al. (2003) and Auerbach et al. (2014). In the former, the authors evolve virtual structures with the idea of physical implementation; in the latter, the authors propose a tool for the co-evolution of morphology and control, also re-implementing one resulting creature in hardware.

Learning

Machine learning has traditionally been a prominent approach for the creation of robot controllers. This can usually be divided into three main categories: supervised, unsupervised, and reinforcement learning. The former two use sets of collected data (i.e., datasets) to train the robot offline; the latter uses experience to do it. In this section, we introduce briefly all these methods.

Supervised learning uses training data associated with a desired solution; this makes it clear to the robot what outcome it has to produce. Usually, supervised learning in robotics is applied to regression and classification problems (Rybczak et al., 2024). The former involves learning mappings between data, such as motor responses to inputs or control commands, and also permits predicting events such as collision and slippage (Uriot et al., 2022; Gonzalez et al., 2018). The latter is used in navigation, localization, and terrain classification (Gonzalez et al., 2018; H. Li et al., 2023; Zheng et al., 2024). For example, regression has been used to predict future soft-robot states according to actuation (Z. Chen et al., 2025). Furthermore, supervised learning has also been used in what is commonly called *reservoir computing*. Reservoir computing is a framework in which the inputs of the computing system pass through a "reservoir" that projects them into a high-dimensional space, permitting the training of a simple linear readout layer (Nakajima et al., 2021); this approach has also been used to control robots (Baldini, 2022). For instance, Antonelo et al. (2008) use reservoir computing in a robot navigating a maze so as to detect locations and complex events occurring in time. In that work, the authors employed a dedicated recurrent neural network as a reservoir; nevertheless, later works demonstrated that even the robot body itself can be involved in the computation. This is the case, for instance, of Nakajima et al. (2013), in which the authors use a robot soft-body as a reservoir. Later, also Bhovad et al. (2021) propose using the body of an origami-based robot as a reservoir to perform pattern generation for movement.

Unsupervised learning attempts instead to extract knowledge from large amounts of data without any indication of relationships or expected output. It is typically employed to solve classification tasks by identifying groups of similar samples, such as in localization and anomaly detection (T. Chen et al., 2020). For instance, Balaska et al. (2020) use unsupervised learning in the former case to achieve autonomous localization during navigation. Instead, Häussermann et al. (2015) use it in the latter case for their anomaly detection framework for robot behaviors considering spatial-temporal data. Meanwhile, Samuel et al. (2021) train autoencoders to identify anomalous situations in robotic-assisted surgery. Finally, Khatib et al. (2024) implement leader-following in a group of simulated robots equipped with a (unsupervised trained) system for filtering out anomalous sensory data.

Supervised and unsupervised learning techniques have their place in robotics; nevertheless,

reinforcement learning is significantly more commonly used to train robot behaviors. Reinforcement learning is a specific class of machine learning that uses direct experience to learn how to operate in an environment (Sutton et al., 2018). The process revolves around the use of rewards, which are bestowed upon the agent each time it achieves a goal or reaches a desired state. When this happens, a model records which action was successfully used in each state, so as to increase its chance of being used again in the same conditions. Sometimes, in order to discourage the system from performing specific actions, designers also use negative rewards, reducing the probability of using specific actions in specific states. Usually, the system starts by *exploring* random actions so as to start collecting data; then, once it has reached a sufficient level of knowledge, it begins *exploiting* the gained knowledge.

Reinforcement learning is now a well-established approach that benefits from ongoing efforts aimed at increasing its effectiveness. For instance, the first reinforcement learning strategies, such as Q-learning, depended on discrete states and actions to operate (Watkins et al., 1992), making them impractical when the state-action space increases or becomes continuous (Jang et al., 2019). The current state-of-the-art in reinforcement learning employs (deep) neural networks to approximate expected rewards in a given state and to select appropriate actions (Silver et al., 2016), permitting operations even in the continuous domain.

The literature contains large evidence of the use of reinforcement learning in robotics. Singh et al. (2022) report that it has been used for learning (i) navigation, (ii) grasping, (iii) vision, (iv) human-robot interactions, and (v) multi-robot coordination tasks. For instance, D. Li et al. (2024) propose a framework for overcoming issues while moving from simulation to real-world operations in the context of autonomous driving (i.e., navigation). Similarly, Cheng et al. (2018) propose collision avoidance in an underwater environment subject to disturbances such as currents. Joshi et al. (2020) perform objects grasping with an actuated arm; and later, Liang et al. (2022) perform a similar task using a multi-finger robotic arm with tactile fingerprint sensing, which increases the perception and thus the difficulty of the control. Kalashnikov et al. (2018) enhance grasping tasks by introducing control based on vision. Practically, they continuously update the grasping plan and strategy based on the most recent observations. In the scope of human-robot interaction, Ghadirzadeh et al. (2021) propose a framework for deep reinforcement learning to assist humans in packaging operations: a motion capture suite captures

the operator's position and passes it to the robot so as to produce a coherent behavior. Meanwhile, C. Chen et al. (2019) train a system capable of interpreting human motion in a crowd, so as to enable the seamless navigation of robots. Finally, in the scope of multi-robot coordination, D. Wang et al. (2020) propose a system trained to overcome resource competition and perform obstacle avoidance between robots in real time. Gharbi et al. (2023) and Szpirer et al. (2024) propose inverse reinforcement learning to train a swarm of robots to complete a mission starting from examples of desired behavior.

Since its introduction, reinforcement learning brought significant advancements in robotics, enabling the learning of complex tasks. Nevertheless, it remains a surprisingly difficult topic to work on (Schaal, 2002). Despite the proposal of successful approaches and solutions, tackling high-dimensional spaces, the reality gap, and data collection still hinder its adoption in many contexts.

Other approaches

Evolutionary algorithms are commonly used for the exploration of search spaces. Nevertheless, they are not the only way to find effective solutions. An alternative approach is sampling different solutions from reasonable distributions and discarding poor ones in a process called *racing* (Birattari et al., 2002). The remaining solutions can then be used to select the section of the space in which to search (i.e., the sampling distributions) so as to iteratively converge to the area of the solution space containing the best solutions (Balaprakash et al., 2007). Therefore, racing and evolutionary strategies share the idea that suitable solutions have common characteristics and that testing highlights the best-performing ones. The difference resides rather in the way they select candidate solutions to evaluate: the former by sampling, the latter by crossover and mutation.

In robotics, racing approaches have been successfully used in AutoMoDe (Birattari et al., 2021), a tool that generates controllers suitable for completing a target mission with swarms of robots. The generated controllers are usually finite state machines (Francesca et al., 2015), but different versions of the tool also considered behavior trees (Kuckling et al., 2018) and artificial neural networks (Ligot et al., 2020). Racing algorithms have demonstrated to be very reliable and effective for designing robot controllers that successfully complete selected missions. For

instance, they have produced solutions for aggregation, foraging (Francesca et al., 2015), decision (Hasselmann et al., 2018a), sheltering, and coverage (Francesca et al., 2014) tasks. Furthermore, racing algorithms operating by composing low-level behaviors have been shown to outperform both human programmers and evolutionary generated neural networks (Francesca et al., 2014; Hasselmann et al., 2021).

1.2.2 Online approaches

Most artificial systems in our world are primarily static and lack adaptive capabilities. Therefore, it is intuitive that researchers in the field of adaptive robotics take inspiration from biological systems to propose novel solutions. Among the most investigated is *neuroplasticity*, which enables adaptation through the variation of weights, nodes, or architecture of neural networks. Alternative approaches used *learning* paradigms to continuously train the system during operation. Practically, these solutions are implemented by means of different approaches: (i) *evolutionary*-based, (ii) *learning*-based, and (iii) other unconventional approaches. The evolutionary approach is classical evolutionary robotics operating at runtime on the robot itself (and thus with reduced resources). This changes the architecture and operative strategy of the controller iteratively. Alternatively, evolutionary algorithms can also be used offline to obtain plastic systems that adapt online. Learning-based approaches modify, instead, the behavior by apprehending suitable input-output mappings or strengthening successful actions at runtime, with obviously increased difficulties with respect to offline approaches. Finally, the diversity of existing approaches requires a separate category consisting of everything not belonging to the previous two. In the following subsections, we briefly introduce and discuss the main use of these approaches (some of which might even include a bootstrap phase offline), presenting notable experimental works employing them.

Evolutionary robotics

As introduced in Section 1.2.1, evolutionary-inspired approaches are largely used in robotics. Nevertheless, in that section, we focused on the design of a static controller able to operate in the conditions considered during the offline learning phase. In this section, instead, we consider

two different applications of evolutionary robotics: (i) to design adaptive control systems, and (ii) to operate at runtime on the robot itself.

Until now, a widespread solution for the design of adaptive control systems has been the creation of plastic neural networks. Those enable behavioral variability thanks to the dynamic modulation of network weights according to activation (Floreano et al., 2000a) or interferences from other neurons (Soltoggio et al., 2008). For instance, in Floreano et al. (1999) and Floreano et al. (2000b), the authors evolve a set of local weight modification rules that control the plasticity of a neural controller so as to adapt online to solve a task. The task requires the robot to stay under a light source positioned in the environment and to turn it on whenever it (automatically) turns off. The same authors also propose a similar approach where the architecture of the neural network is evolved as well (Floreano et al., 2001a; Floreano et al., 2001b). In these works, the authors report a much better performance for the adaptive controller when compared against a controller with fixed weights. Interestingly, they also notice that the increase in performance is due to the plastic controller memorizing the two different behaviors required to complete the task, dynamically switching between them (Urzelai et al., 2001). Niv et al. (2002) also evolve rules modulating the synaptic weight of a neural network, modelling bees' foraging behaviors. While the previous works considered the evolution of the network architecture and the synapses modification rules, Soltoggio et al. (2008) and Dürr et al. (2008) introduce a new concept: the modulatory neuron, which influences the synaptic plasticity of neurons it connects to. They find that this approach performs better than controllers with local adaptation rules, attaining higher performance in a T-maze task.

In addition to creating plastic controllers, evolutionary robotics has also been used for the optimization of control software runtime, on the robot itself (Eiben et al., 2012). Among the first to propose this approach, we report Floreano et al. (1994) and Floreano et al. (1996), in which the authors train a robot to perform collision avoidance and homing, respectively. Their experiment is just explorative, without aiming at any real-world application scenario. Mahdavi et al. (2006) use an evolutionary algorithm for a worm-like robot that undergoes actuation faults; when the damage occurs, the process finds new movement strategies that work with the current state of the robot. Bredeche et al. (2010) discuss some possible problems of online evolution of control software, such as noisy evaluations and limited computational power, proposing

some solutions (e.g., re-evaluation) and assessing them in a simulated collision avoidance task. Later, Eiben et al. (2010b) investigate the factors that most influence the outcome of the adaptive process, identifying the mutation as the main driver of change. Specifically, they highlight that automatically deciding the mutation intensity leads to better performance with respect to using a fixed mutation intensity. In Haasdijk et al. (2010), the authors perform similar experiments proposing even to re-evaluate the solutions previously tested; the goal is to face noise, which can affect the evaluation of the controllers. Finally, in Haasdijk et al. (2011) they propose discarding clearly inferior models early without re-evaluating them. Silva et al. (2012a) and Silva et al. (2014b) investigate the combination of neuroevolution and neuromodulation in a foraging task, reporting that this improves the performance with respect to neuroevolution only. Some of these authors later propose incorporating behavioral building blocks and early discarding poor solutions to decrease the adaptation time of the system (Silva et al., 2014a; Silva et al., 2016). Meanwhile, Silva et al. (2017) propose an embedded evolutionary algorithm, odNEAT (Silva et al., 2012b; Silva et al., 2015b), so as to simultaneously optimize topology and weights of a neural network controller with the aim to perform three different tasks: (i) navigation with collision avoidance, (ii) homing, and (iii) aggregation in less than one hour. More recent works include Braccini et al. (2020), Braccini et al. (2022c), Braccini et al. (2022b), and Braccini et al. (2023), in which the authors use a mechanism maintaining the best controller found (i.e., individual, in the context of evolutionary robotics) and performing small changes so as to produce the desired behavior. Later, they explore attaining active homeostasis in a virtual agent, whereby it modifies the environment so as to make it more comfortable for itself (Braccini et al., 2024b). The novelty of these works resides in the use of an unconventional computing medium, the Boolean network, and in the clear biological inspiration, constituting a fundamental part of the investigation.

In the context of groups of robots, evolutionary strategies have been used in various ways. Floreano et al. (1997), Nolfi et al. (1998b), and Nolfi et al. (1998a) co-evolve a prey and predator in simulation; therefore, the emerging behaviors are adversarial. Despite being in simulation, we present these works in this section because the adaptation is not intended to end, working in a continuous loop. Later, to complete the investigation, the authors try the task also on real robots (Floreano et al., 1998). Bianco et al. (2004b) propose, among other goals, to adapt

the control software in groups of robots by sharing their genotype with mutations; their goal is to move towards open-ended, in hardware, robotic evolution. Baele et al. (2009) propose a similar approach, aiming to create swarms of robots able to self-assemble and evolve their behavior at runtime; they conduct a demonstrative experiment in which robots endowed with two different evolution strategies interact, investigating which strategy is more efficient. An interesting use of evolutionary strategies in multi-robot systems is proposed in Pini et al. (2008), where the authors make a learner robot apprehend a behavior from a demonstrator. Ferrante et al. (2015) evolve a controller selecting different behaviors at runtime in a swarm of foraging robots; then, they proceed by assessing the automatic emergence of such behaviors starting from lower-level ones. They find that the resulting strategy is optimized for the environmental characteristics present during adaptation. In Silva et al. (2015a), the authors show that distributed online evolution and subsequent sharing of robot controllers speed up learning selected tasks. Heinerman et al. (2016) follow a similar approach, achieving a foraging behavior in a group of robots by sharing their genome. E. Hansen et al. (2018) attempt to use neuroevolution in a task requiring some robots to spread across two targets so as to create a virtual connectivity line. Their results show that, as the group size increases, neuroevolution attains better performance than pre-programmed robots. Jones et al. (2019) evolve foraging behavior for a swarm; their approach is quite interesting, as each robot is a node of a distributed evolutionary process performing onboard simulations and selecting the most promising outcome. Obviously, this approach requires a great amount of computational power (the authors report that it reaches 1 teraflop when considering all the robots), but it permits learning the task in around 15 minutes. Finally, recently Ferigo et al. (2025) propose co-designing body and brain of robots that can change behavior online by means of plastic synapses. Their approach is to use evolutionary robotics to select the organization of composable soft-robots, the *voxels*, and their controlling artificial neural network.

Learning

In recent years, the need to tackle dynamic contexts has started to arise, especially from the field of machine learning. One of the proposals to face changing situations has been *online reinforcement learning*. Unlike the approach presented in Section 1.2.1, online reinforcement learning

considers the possibility of never-ending the model update, or, at least, of performing some sort of fine-tuning on the robot itself. Nair et al. (2021) explores, for instance, this latter case, training a robotic arm offline and fine-tuning it online, improving the performance noticeably with respect to the non-tuned system. Even Wen et al. (2020) propose to fine-tune a control system online by means of reinforcement learning, but they do so on a robotic knee prosthesis to enhance walking capabilities. In Zhang et al. (2020), the authors take one step more, proposing to continuously train the robot online so as to improve its navigation capabilities in a dynamic environment. Bakker et al. (2006) employ a robot that builds a model of the environment on the fly and concurrently trains the action policy on it so as to perform a localization and approaching task. Despite interesting results, the literature on online reinforcement learning is not particularly developed, with research effort having focused on different approaches.

Many significant results on adaptation come from the research area known as *continual learning* (L. Wang et al., 2024). Continual learning is a branch of machine learning that considers the possibility of performing additional training so as to enhance the capability of the system after deployment. This usually considers three possible fundamental types of learning (Ven et al., 2022): (i) learning new, different tasks, (ii) learning the same task in different conditions, and (iii) learning to distinguish different classes. Currently, we can identify five approaches that have been used to reach these goals (L. Wang et al., 2024): (i) regularization, penalizing the change of important parameters during subsequent training phases, (ii) replay, including old representative samples during new training phases, (iii) optimization, finding parameter updates that prevent interference between old and new tasks, (iv) representation, trying to learn the common patterns among tasks, and (v) architectural, using task-specific parameters or network parts. Nevertheless, what holds back continual learning is the risk of forgetting previously acquired knowledge during subsequent training iterations, a process called *catastrophic forgetting*. This is indeed the main target of the research effort on the topic.

As for what concerns robotics, continual learning has been extensively used recently. Auddy et al. (2023) use it to learn motion skills from demonstrations. Liu et al. (2024) use it for a similar goal: learning grasping from a teacher. Their system includes, indeed, a module selecting informative knowledge to feed to the control system for further training. Churamani et al. (2022) use it to "fine-tune" the robot behavior on human preferences so as to improve interac-

tion. Piqué et al. (2022) use continual learning so as to face changes in soft-robots undergoing degradation. It is also common to use continual learning to enable robots to recognize more and more classes of objects. For instance, Ayub et al. (2022) propose a system for few-shot class learning, in which the incoming data are unlabeled. Nie et al. (2024) also propose continual learning to identify new object classes for recognition tasks, but they additionally permit the robot to query the name of the new classes. Hajizada et al. (2024) propose continuous learning to distinguish different classes in the world with a few collected samples, labeled or unlabeled. Finally, Ye et al. (2024) propose continual learning to better discriminate the target during a human following task.

What differentiates continual learning from the approaches we will propose in this dissertation is that, in its classical setting, it requires the system to learn contents in a sequence as if they were presented simultaneously, thus without forgetting previous ones. For this motivation, this approach ends up being computationally demanding (Shaheen et al., 2022). In this work, we do not impose this constraint, permitting the robot to forget previous information that is no longer relevant or useful. The idea is that this will permit even low-end robots to adapt in a much less computationally intensive way. Furthermore, continual learning requires collecting data, often annotated, to be used to train the system (Lesort et al., 2020; Shaheen et al., 2022), while our approach considers data only implicitly through the proxy of performance.

Other approaches

In addition to the previously described approaches, we identify further strategies used to provide adaptive capabilities to robots; since we are unable to include them in the previous sections, we describe them here. One of them is proposed in Ram et al. (1997), in which the authors fine-tune low-level behaviors online by randomly varying their current parameters. Another is proposed in Urzelai et al. (1998), in which the authors propose a compound system capable of self-adapting some low-level behavioral modules by means of reinforcement learning and of combining them by means of evolutionary strategies. They test the system in a task requiring moving in an environment while avoiding obstacles and periodically recharging the batteries. Later, they also require the robot to move objects outside the arena while maintaining the previously evolved abilities. In Urzelai et al. (1999), the authors use an algorithm inspired by evo-

lutionary strategies, but computationally simpler, identifying the probability of a parameter to belong to a successful solution. Notably, they propose the use of a "decay" factor resetting the aforementioned probability so as to tackle changing environments. Lipson et al. (2006) propose that the robot estimate its own structure so as to use it to synthesize new opportune behaviors at runtime. The idea is to leverage the perception resulting from an action so as to identify morphological changes. The authors validate the idea on both simulated (J.C. Bongard et al., 2004) and real robots (J. Bongard et al., 2006), succeeding in creating a resilient machine capable of overcoming faults. Cully et al. (2015) use, instead, a different approach, proposing to pre-process the expected performance of different behaviors, so as to select the most promising for a specific condition online in a few trials. The system then updates the prediction of behaviors similar to the one tried on the real robot. The authors validate their approach on a real robot, confirming its efficacy.

While holding all the previous works in high regard, we devote a separate discussion to the so-called *free energy principle* (Friston et al., 2006), which is recently arousing interest in the research community. The free energy principle aims to model learning as the minimization of the surprise an agent experiences in the world; this can be done in two ways: (i) adjusting its model of the environment, and (ii) adjusting its actions so as to minimize surprise about the effects they have on the environment. We refer to the former as *perceptual inference* and to the latter as *active inference* (Friston, 2009). Despite the hype surrounding this framework, we find the literature involving its application in robotics quite lacking. Hamann (2014) starts from swarm robotic considerations to propose an investigation about the emergence of different behaviors in virtual agents when trying to minimize surprise in future-state predictions. Annabi et al. (2020) use active inference to induce a system to learn motor primitives for a robotic arm. Hu et al. (2020) try to learn the optimal control policy for a robotic arm with uncertain parameters and noise by minimizing the free energy. Oliver et al. (2022) do something similar, using active inference to correct the motion of a robot in space according to its expected trajectory in conditions of noise. Meanwhile, multiple works attempted to use active inference to achieve active vision, which is the ability to manipulate viewpoints so as to create an internal model of the world (Van de Maele et al., 2021; Haddon-Hill et al., 2024); in these experiments, this is done with a robotic arm holding a camera. Bos et al. (2022) propose a system to estimate the

robot state and inputs, updating the free energy principle prior expectation about the world in different conditions. Finally, despite involving virtual agents, we find interesting the work of Berseth et al. (2021), in which the authors effectively update both the internal model of the world and the actions to execute so as to minimize surprise.

While the free energy principle is currently a hot topic, we believe that some work is still required to clearly demonstrate its advantage in adaptive robotics. Therefore, we have decided not to investigate this aspect in this dissertation.

Chapter bibliography

Almansoori, A., M. Alkilabi, and E. Tuci (2024). "On the evolution of adaptable and scalable mechanisms for collective decision-making in a swarm of robots." In: *Swarm Intelligence* 18.1, pp. 79–99.

Annabi, L., A. Pitti, and M. Quoy (2020). "Autonomous learning and chaining of motor primitives using the Free Energy Principle." In: *2020 International Joint Conference on Neural Networks (IJCNN)*. Institute of Electrical and Electronics Engineers, pp. 1–8.

Antonelo, E.A., B. Schrauwen, and D. Stroobandt (2008). "Event detection and localization for small mobile robots using reservoir computing." In: *Neural Networks* 21.6, pp. 862–871.

Auddy, S. et al. (2023). "Continual learning from demonstration of robotics skills." In: *Robotics and Autonomous Systems* 165, p. 104427.

Auerbach, J. et al. (2014). "RoboGen: Robot Generation through Artificial Evolution." In: *ALIFE 14. The Fourteenth International Conference on the Synthesis and Simulation of Living Systems*. The MIT Press, pp. 136–137.

Ayub, A. and C. Fendley (2022). "Few-Shot Continual Active Learning by a Robot." In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., pp. 30612–30624.

Baele, G. et al. (2009). "Open-ended on-board Evolutionary Robotics for robot swarms." In: *2009 IEEE Congress on Evolutionary Computation*. Institute of Electrical and Electronics Engineers, pp. 1123–1130.

Bakker, B. et al. (2006). "Quasi-online reinforcement learning for robots." In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* Institute of Electrical and Electronics Engineers, pp. 2997–3002.

Balaprakash, P., M. Birattari, and T. Stützle (2007). "Improvement Strategies for the F-Race Algorithm: Sampling Design and Iterative Refinement." In: *Hybrid Metaheuristics*. Vol. 4771. Springer Berlin Heidelberg, pp. 108–122.

Balaska, V. et al. (2020). "Unsupervised semantic clustering and localization for mobile robotics tasks." In: *Robotics and Autonomous Systems* 131, p. 103567.

Baldassarre, G., S. Nolfi, and D. Parisi (2003a). "Evolution of Collective Behavior in a Team of Physically Linked Robots." In: *Applications of Evolutionary Computing. EvoWorkshop 2003: EvoBIO, EvoCOP, EvoIASP, EvoMUSART, EvoROB, and EvoSTIM, Essex, UK, April 14-16, 2003, Proceedings*. Vol. 2611. Springer Berlin Heidelberg, pp. 581–592.

— (2003b). "Evolving Mobile Robots Able to Display Collective Behaviors." In: *Artificial Life* 9.3, pp. 255–267.

Baldini, P. (2022). *Reservoir Computing in robotics: a review*. arXiv: 2206.11222.

Barker, D.J.P. (1995). "Fetal origins of coronary heart disease." In: *British Medical Journal* 311.6998, pp. 171–174.

Bell, G. (2007). *Selection: The Mechanism of Evolution*. 2nd ed. Oxford, United Kingdom: Oxford University Press.

Benzinger, T.H. (1969). "Heat regulation: homeostasis of central temperature in man." In: *Physiological Reviews* 49.4, pp. 671–759.

Bergonti, F. et al. (2024). "Co-Design Optimisation of Morphing Topology and Control of Winged Drones." In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. Institute of Electrical and Electronics Engineers, pp. 8679–8685.

Berseth, G. et al. (2021). "SMiRL: Surprise Minimizing Reinforcement Learning in Unstable Environments." In: *International Conference on Learning Representations (ICLR 2021)*.

Bhovad, P. and S. Li (2021). "Physical reservoir computing with origami and its application to robotic crawling." In: *Scientific Reports* 11.1, p. 13002.

Bianco, R. and S. Nolfi (2004a). "Evolving the Neural Controller for a Robotic Arm Able to Grasp Objects on the Basis of Tactile Sensors." In: *Adaptive Behavior* 12.1, pp. 37–45.

— (2004b). "Toward open-ended evolutionary robotics: evolving elementary robotic units able to self-assemble and self-reproduce." In: *Connection Science* 16.4, pp. 227–248.

Birattari, M., A. Ligot, and G. Francesca (2021). "AutoMoDe: A Modular Approach to the Automatic Off-Line Design and Fine-Tuning of Control Software for Robot Swarms." In: *Automated Design of Machine Learning and Search Algorithms*. Springer International Publishing, pp. 73–90.

Birattari, M. et al. (2002). "A Racing Algorithm for Configuring Metaheuristics." In: *GECCO-2002: Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*. Vol. 2. Morgan Kaufmann Publishers, pp. 11–18.

Blix, A.S. (2016). "Adaptations to polar life in mammals and birds." In: *Journal of Experimental Biology* 219.8, pp. 1093–1105.

Bongard, J., V. Zykov, and H. Lipson (2006). "Resilient machines through continuous self-modeling." In: *Science* 314.5802, pp. 1118–1121.

Bongard, J.C. and H. Lipson (2004). "Automated damage diagnosis and recovery for remote robotics." In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*. Vol. 4. Institute of Electrical and Electronics Engineers, pp. 3545–3550.

Bongard, J.C. and R. Pfeifer (2003). "Evolving Complete Agents using Artificial Ontogeny." In: *Morpho-functional Machines: The New Species*. Tokyo: Springer Japan, pp. 237–258.

Bos, F. et al. (2022). "Free Energy Principle for State and Input Estimation of a Quadcopter Flying in Wind." In: *2022 International Conference on Robotics and Automation (ICRA)*. Institute of Electrical and Electronics Engineers, pp. 5389–5395.

Braccini, M., E. Barbieri, and A. Roli (2023). "The Role of Dynamical Regimes of Online Adaptive BN-Robots in Noisy Environments." In: *Artificial Life and Evolutionary Computation*. Vol. 1780. Springer Nature Switzerland, pp. 183–194.

Braccini, M., A. Roli, and S.A. Kauffman (2020). *Online adaptation in robots as biological development provides phenotypic plasticity*. arXiv: 2006.02367.

— (2022b). "A Novel Online Adaptation Mechanism in Artificial Systems Provides Phenotypic Plasticity." In: *Artificial Life and Evolutionary Computation*. Vol. 1722. Springer Nature Switzerland, pp. 121–132.

Braccini, M. et al. (2022c). "On the Criticality of Adaptive Boolean Network Robots." In: *Entropy* 24.10, 1368:1–1368:21.

Braccini, M. et al. (2024b). "Sensory–Motor Loop Adaptation in Boolean Network Robots." In: *Sensors* 24.11, p. 3393.

Bradshaw, A.D. (1965). "Evolutionary Significance of Phenotypic Plasticity in Plants." In: vol. 13. Academic Press, pp. 115–155.

Bredeche, N., E. Haasdijk, and A.E. Eiben (2010). "On-Line, On-Board Evolution of Robot Controllers." In: *Artifical Evolution*. Vol. 5975. Springer Berlin Heidelberg, pp. 110–121.

Brooks, R.A. (1992). "Artificial Life and Real Robots." In: *Toward a Practice of Autonomous Systems. Proceedings of the First European Conference on Artificial Life*. Cambridge, Massachusetts, USA: The MIT Press, pp. 3–10.

Cannon, W.B. (1929). "Organization for physiological homeostasis." In: *Physiological Reviews* 9.3, pp. 399–431.

Carroll, J.L. (2003). "Invited Review: Developmental plasticity in respiratory control." In: *Journal of Applied Physiology* 94.1, pp. 375–389.

Chechik, G., I. Meilijson, and E. Ruppin (1999). "Neuronal Regulation: A Mechanism for Synaptic Pruning During Brain Maturation." In: *Neural Computation* 11.8, pp. 2061–2080.

Chen, C. et al. (2019). "Crowd-Robot Interaction: Crowd-Aware Robot Navigation With Attention-Based Deep Reinforcement Learning." In: *2019 International Conference on Robotics and Automation (ICRA)*. Institute of Electrical and Electronics Engineers, pp. 6015–6022.

Chen, T. et al. (2020). "Unsupervised Anomaly Detection of Industrial Robots Using Sliding-Window Convolutional Variational Autoencoder." In: *IEEE Access* 8, pp. 47072–47081.

Chen, Z. et al. (2025). "Data-Driven Methods Applied to Soft Robot Modeling and Control: A Review." In: *IEEE Transactions on Automation Science and Engineering* 22, pp. 2241–2256.

Cheney, N. et al. (2016). "On the Difficulty of Co-Optimizing Morphology and Control in Evolved Virtual Creatures." In: *ALIFE 2016. The Fifteenth International Conference on the Synthesis and Simulation of Living Systems*. The MIT Press, pp. 226–233.

Cheng, Y. and W. Zhang (2018). "Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels." In: *Neurocomputing* 272, pp. 63–73.

Churamani, N. et al. (2022). "Continual Learning for Affective Robotics: A Proof of Concept for Wellbeing." In: *2022 10th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*. Institute of Electrical and Electronics Engineers, pp. 1–8.

Cliff, D., P. Husbands, and I. Harvey (1993). "Explorations in Evolutionary Robotics." In: *Adaptive Behavior* 2.1, pp. 73–110.

Corucci, F. et al. (2018). "Evolving Soft Locomotion in Aquatic and Terrestrial Environments: Effects of Material Properties and Environmental Transitions." In: *Soft Robotics* 5.4, pp. 475–495.

Crespi, E.J. and R.J. Denver (2005). "Ancient origins of human developmental plasticity." In: *American Journal of Human Biology* 17.1, pp. 44–54.

Cully, A. et al. (2015). "Robots that can adapt like animals." In: *Nature* 521.7553, pp. 503–507.

Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. London, United Kingdom: John Murray.

DeWitt, T.J., A. Sih, and D.S. Wilson (1998). "Costs and limits of phenotypic plasticity." In: *Trends in Ecology & Evolution* 13.2, pp. 77–81.

Doncieux, S. et al. (2015). "Evolutionary Robotics: What, Why, and Where to." In: *Frontiers in Robotics and AI* 2.

Dorigo, M. and U. Schnepf (1991). "Organisation of robot behaviour through genetic learning processes." In: *Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments*. Vol. 2. Institute of Electrical and Electronics Engineers, pp. 1456–1460.

Dorigo, M. et al. (2004). "Evolving Self-Organizing Behaviors for a Swarm-Bot." In: *Autonomous Robots* 17.2, pp. 223–245.

Duarte, M. et al. (2016). "Evolution of Collective Behaviors for a Real Swarm of Aquatic Surface Robots." In: *PLOS ONE* 11.3, pp. 1–25.

Dürr, P. et al. (2008). "Evolvability of Neuromodulated Learning for Robots." In: *2008 ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems (LAB-RS)*. Institute of Electrical and Electronics Engineers, pp. 41–46.

Eiben, A.E., E. Haasdijk, and N. Bredeche (2010a). "Embodied, On-line, On-board Evolution for Autonomous Robotics." In: *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*. Vol. 7. Springer, pp. 361–382.

Eiben, A.E., G. Karafotias, and E. Haasdijk (2010b). "Self-Adaptive Mutation in On-line, On-board Evolutionary Robotics." In: *2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop*. Institute of Electrical and Electronics Engineers, pp. 147–152.

Eiben, A.E., S. Kernbach, and E. Haasdijk (2012). "Embodied artificial evolution: artificial evolutionary systems in the 21st Century." In: *Evolutionary Intelligence* 5.4, pp. 261–272.

Ferigo, A. et al. (2025). "Totipotent neural controllers for modular soft robots: Achieving specialization in body-brain co-evolution through Hebbian learning." In: *Neurocomputing* 614, p. 128811.

Ferrante, E. et al. (2015). "Evolution of Self-Organized Task Specialization in Robot Swarms." In: *PLOS Computational Biology* 11.8, pp. 1–21.

Fisher, R.A. (1930). *The Genetical Theory of Natural Selection*. Oxford: The Clarendon Press.

Floreano, D. and F. Mondada (1994). "Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot." In: *Animals to Animats 3. Proceedings of the Third International Conference on Simulation of Adaptive Behavior*. The MIT Press, pp. 421–430.

— (1996). "Evolution of homing navigation in a real mobile robot." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.3, pp. 396–407.

Floreano, D. and S. Nolfi (1997). "God Save the Red Queen! Competition in Co-Evolutionary Robotics." In: *Genetic Programming 1997: Proceedings of the Second Annual Conference*. Morgan Kaufman Publishers.

Floreano, D., S. Nolfi, and F. Mondada (1998). "Co-Evolution and Ontogenetic Change in Competing Robots." In: *Advances in the Evolutionary Synthesis of Intelligent Agents*. The MIT Press, pp. 273–305.

Floreano, D. and J. Urzelai (1999). "Evolution of Neural Controllers with Adaptive Synapses and Compact Genetic Encoding." In: *Advances in Artificial Life*. Vol. 1674. Springer Berlin Heidelberg, pp. 183–194.

— (2000a). "Evolutionary on-line selforganisation of autonomous robots." In: *Proceedings of the 5th International Conference on Artificial Life and Robotics*.

— (2000b). "Evolutionary robots with on-line self-organization and behavioral fitness." In: *Neural Networks* 13.4, pp. 431–443.

— (2001a). "Evolution of Plastic Control Networks." In: *Autonomous Robots* 11.3, pp. 311–317.

— (2001b). "Neural morphogenesis, synaptic plasticity, and evolution." In: *Theory in Biosciences* 120.3, pp. 225–240.

Francesca, G. et al. (2014). "An Experiment in Automatic Design of Robot Swarms." In: *Swarm Intelligence*. Vol. 8667. Springer International Publishing, pp. 25–37.

Francesca, G. et al. (2015). "AutoMoDe-Chocolate: automatic design of control software for robot swarms." In: *Swarm Intelligence* 9.2, pp. 125–152.

Friston, K. (2009). "The free-energy principle: a rough guide to the brain?" In: *Trends in Cognitive Sciences* 13.7, pp. 293–301.

Friston, K., J. Kilner, and L. Harrison (2006). "A free energy principle for the brain." In: *Journal of Physiology-Paris* 100.1, pp. 70–87.

Galván, A. (2010). "Neural plasticity of development and learning." In: *Human Brain Mapping* 31.6, pp. 879–890.

Ghadirzadeh, A. et al. (2021). "Human-Centered Collaborative Robots With Deep Reinforcement Learning." In: *IEEE Robotics and Automation Letters* 6.2, pp. 566–571.

Ghalambor, C.K., L.M. Angeloni, and S.P. Carroll (2010). "Behavior as Phenotypic Plasticity." In: *Evolutionary Behavioral Ecology*. Oxford University Press, pp. 90–107.

Ghalambor, C.K. et al. (2007). "Adaptive versus non-adaptive phenotypic plasticity and the potential for contemporary adaptation in new environments." In: *Functional Ecology* 21.3, pp. 394–407.

Gharbi, I. et al. (2023). *Show me what you want: Inverse reinforcement learning to automatically design robot swarms by demonstration*. arXiv: 2301.06864.

Gomes, J. and A.L. Christensen (2018). "Task-Agnostic Evolution of Diverse Repertoires of Swarm Behaviours." In: *Swarm Intelligence*. Vol. 11172. Springer International Publishing, pp. 225–238.

Gomes, J., P. Mariano, and A.L. Christensen (2019). "Challenges in cooperative coevolution of physically heterogeneous robot teams." In: *Natural Computing* 18.1, pp. 29–46.

Gonzalez, R. and K. Iagnemma (2018). *DeepTerramechanics: Terrain Classification and Slip Estimation for Ground Robots via Deep Learning*. arXiv: 1806.07379.

Gould, S.J. and E.S. Vrba (1982). "Exaptation—a Missing Term in the Science of Form." In: *Paleobiology* 8.1, pp. 4–15.

Groß, R. and M. Dorigo (2004). "Evolving a Cooperative Transport Behavior for Two Simple Robots." In: *Artificial Evolution. 6th International Conference, Evolution Artificielle, EA 2003, Marseilles, France, October 27-30, 2003, Revised Selected Papers*. Vol. 2936. Springer Berlin Heidelberg, pp. 305–316.

Haasdijk, E., A. Atta-ul-Qayyum, and A.E. Eiben (2011). "Racing to improve on-line, on-board evolutionary robotics." In: *GECCO '11. Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*. Association for Computing Machinery, pp. 187–194.

Haasdijk, E., A.E. Eiben, and G. Karafotias (2010). "On-line evolution of robot controllers by an encapsulated evolution strategy." In: *IEEE Congress on Evolutionary Computation*. Institute of Electrical and Electronics Engineers, pp. 1–7.

Haddon-Hill, G.W. and S. Murata (2024). "Active Vision for Physical Robots Using the Free Energy Principle." In: *Artificial Neural Networks and Machine Learning – ICANN 2024. 33rd International Conference on Artificial Neural Networks, Lugano, Switzerland, September 17–20, 2024, Proceedings, Part X*. Vol. 15025. Springer Nature Switzerland, pp. 270–284.

Hajizada, E., B. Swaminathan, and Y. Sandamirskaya (2024). "Continual Learning for Autonomous Robots: A Prototype-based Approach." In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Institute of Electrical and Electronics Engineers, pp. 13988–13995.

Haller, B. von, A. Ijspeert, and D. Floreano (2005). "Co-evolution of Structures and Controllers for Neubot Underwater Modular Robots." In: *Advances in Artificial Life*. Vol. 3630. Springer Berlin Heidelberg, pp. 189–199.

Hamann, H. (2014). "Evolution of Collective Behaviors by Minimizing Surprise." In: *ALIFE 14. The Fourteenth International Conference on the Synthesis and Simulation of Living Systems*. The MIT Press, pp. 344–351.

Hansen, E.A. et al. (2018). "Achieving Connectivity Between Wide Areas Through Self-Organising Robot Swarms Using Embodied Evolution." In: *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. Institute of Electrical and Electronics Engineers, pp. 875–883.

Harding, S. and J.F. Miller (2005). "Evolution in materio: a real-time robot controller in liquid crystal." In: *2005 NASA/DoD Conference on Evolvable Hardware (EH'05)*. Institute of Electrical and Electronics Engineers, pp. 229–238.

Hartfelder, K. and W. Engels (1998). "2 Social Insect Polymorphism: Hormonal Regulation of Plasticity in Development and Reproduction in the Honeybee." In: ed. by R.A. Pedersen and G.P. Schatten. Vol. 40. Current Topics in Developmental Biology. Academic Press, pp. 45–77.

Hasselmann, K., F. Robert, and M. Birattari (2018a). "Automatic Design of Communication-Based Behaviors for Robot Swarms." In: *Swarm Intelligence*. Vol. 11172. Springer International Publishing, pp. 16–29.

Hasselmann, K. et al. (2021). "Empirical assessment and comparison of neuro-evolutionary methods for the automatic off-line design of robot swarms." In: *Nature Communications* 12.1, p. 4345.

Häussermann, K., O. Zweigle, and P. Levi (2015). "A Novel Framework for Anomaly Detection of Robot Behaviors." In: *Journal of Intelligent & Robotic Systems* 77.2, pp. 361–375.

Hebb, D.O. (1949). *The organization of behavior: a neuropsychological theory*. New York, NY: John Wiley & Sons, Inc.

Heinerman, J. et al. (2016). "On-line Evolution of Foraging Behaviour in a Population of Real Robots." In: *Applications of Evolutionary Computation*. Vol. 9598. Springer International Publishing, pp. 198–212.

Hiller, J. and H. Lipson (2012). "Automatic Design and Manufacture of Soft Robots." In: *IEEE Transactions on Robotics* 28.2, pp. 457–466.

Holland, J.H. (1992). "Genetic Algorithms." In: *Scientific American* 267.1, pp. 66–73.

Hu, Y. et al. (2020). "Learning Control for Robotic Manipulator with Free Energy." In: *2020 39th Chinese Control Conference (CCC)*. Institute of Electrical and Electronics Engineers, pp. 1903–1908.

Husbands, P. et al. (2021). "Recent advances in evolutionary and bio-inspired adaptive robotics: Exploiting embodied dynamics." In: *Applied Intelligence* 51.9, pp. 6467–6496.

Jakobi, N., P. Husbands, and I. Harvey (1995). "Noise and the reality gap: The use of simulation in evolutionary robotics." In: *Advances in Artificial Life*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 704–720.

Jäncke, L. (2009). "The plastic human brain." In: *Restorative Neurology and Neuroscience* 27.5, pp. 521–538.

Jang, B. et al. (2019). "Q-Learning Algorithms: A Comprehensive Classification and Applications." In: *IEEE Access* 7, pp. 133653–133667.

Jones, S. et al. (2018). "Evolving Behaviour Trees for Swarm Robotics." In: *Distributed Autonomous Robotic Systems: The 13th International Symposium*. Vol. 6. Cham: Springer International Publishing, pp. 487–501.

Jones, S. et al. (2019). "Onboard Evolution of Understandable Swarm Behaviors." In: *Advanced Intelligent Systems* 1, p. 1900031.

Joshi, S., S. Kumra, and F. Sahin (2020). "Robotic Grasping using Deep Reinforcement Learning." In: *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. Institute of Electrical and Electronics Engineers, pp. 1461–1466.

Kalashnikov, D. et al. (2018). "Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation." In: *Proceedings of The 2nd Conference on Robot Learning*. Vol. 87. Proceedings of Machine Learning Research (PMLR), pp. 651–673.

Karlebach, G. and R. Shamir (2008). "Modelling and analysis of gene regulatory networks." In: *Nature Reviews Molecular Cell Biology* 9.10, pp. 770–780.

Katju, V. and U. Bergthorsson (2018). "Old Trade, New Tricks: Insights into the Spontaneous Mutation Process from the Partnering of Classical Mutation Accumulation Experiments with High-Throughput Genomic Approaches." In: *Genome Biology and Evolution* 11.1, pp. 136–165.

Khatib, A. et al. (2024). "Enhancing Multi-Robot Systems Cooperation through Machine Learning-based Anomaly Detection in Target Pursuit." In: *Journal of Robotics and Control (JRC)* 5.3, pp. 893–901.

Kolb, B. and R. Gibb (2011). "Brain plasticity and behaviour in the developing brain." In: *Journal of the Canadian Academy of Child and Adolescent Psychiatry* 20.4, pp. 265–276.

Kriegman, S. et al. (2020). "A scalable pipeline for designing reconfigurable organisms." In: *Proceedings of the National Academy of Sciences* 117.4, pp. 1853–1859.

Kuckling, J. et al. (2018). "Behavior Trees as a Control Architecture in the Automatic Modular Design of Robot Swarms." In: *Swarm Intelligence*. Vol. 11172. Springer International Publishing, pp. 30–43.

Kuzawa, C.W. and Z.M. Thayer (2011). "Timescales of human adaptation: the role of epigenetic processes." In: *Epigenomics* 3.2, pp. 221–234.

Langerhans, R.B. (2008). "Predictability of phenotypic differentiation across flow regimes in fishes." In: *Integrative and Comparative Biology* 48.6, pp. 750–768.

Lasker, G.W. (1969). "Human Biological Adaptability." In: *Science* 166.3912, pp. 1480–1486.

Lema, S.C. (2008). "The Phenotypic Plasticity of Death Valley's Pupfish: Desert fish are revealing how the environment alters development to modify body shape and behavior." In: *American Scientist* 96.1, pp. 28–36.

Lesort, T. et al. (2020). "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges." In: *Information Fusion* 58, pp. 52–68.

Levis, N.A., A.J. Isdaner, and D.W. Pfennig (2018). "Morphological novelty emerges from pre-existing phenotypic plasticity." In: *Nature Ecology & Evolution* 2.8, pp. 1289–1297.

Lewis, M.A., A.H. Fagg, and A. Solidum (1992). "Genetic programming approach to the construction of a neural network for control of a walking robot." In: *Proceedings 1992 IEEE International Conference on Robotics and Automation*. Vol. 3. Institute of Electrical and Electronics Engineers, pp. 2618–2623.

Li, D. and O. Okhrin (2024). "A platform-agnostic deep reinforcement learning framework for effective Sim2Real transfer towards autonomous driving." In: *Communications Engineering* 3.1, p. 147.

Li, H., H. Jiao, and Z. Yang (2023). "Ship trajectory prediction based on machine learning and deep learning: A systematic review and methods analysis." In: *Engineering Applications of Artificial Intelligence* 126, p. 107062.

Liang, H. et al. (2022). "Multifingered Grasping Based on Multimodal Reinforcement Learning." In: *IEEE Robotics and Automation Letters* 7.2, pp. 1174–1181.

Ligot, A., K. Hasselmann, and M. Birattari (2020). "AutoMoDe-Arlequin: Neural Networks as Behavioral Modules for the Automatic Design of Probabilistic Finite-State Machines." In: *Swarm Intelligence*. Vol. 12421. Springer International Publishing, pp. 271–281.

Lipson, H. and J.B. Pollack (2000). "Automatic design and manufacture of robotic lifeforms." In: *Nature* 406.6799, pp. 974–978.

Lipson, H. et al. (2006). "Evolutionary Robotics for Legged Machines: From Simulation to Physical Reality." In: *Intelligent Autonomous Systems* 9, pp. 11–18.

Liu, J. et al. (2024). "Continual Learning for Robotic Grasping Detection With Knowledge Transferring." In: *IEEE Transactions on Industrial Electronics* 71.9, pp. 11019–11027.

Mahdavi, S.H. and P.J. Bentley (2006). "Innately adaptive robotics through embodied evolution." In: *Autonomous Robots* 20.2, pp. 149–163.

Martin, S.J., P.D. Grimwood, and R.G.M. Morris (2000). "Synaptic Plasticity and Memory: An Evaluation of the Hypothesis." In: *Annual Review of Neuroscience* 23, pp. 649–711.

Meeûs, T. de, F. Prugnolle, and P. Agnew (2007). "Asexual reproduction: Genetics and evolutionary aspects." In: *Cellular and Molecular Life Sciences* 64.11, pp. 1355–1372.

Miglino, O., K. Nafasi, and C.E. Taylor (1994). "Selection for Wandering Behavior in a Small Robot." In: *Artificial Life* 2.1, pp. 101–116.

Morlino, G., V. Trianni, and E. Tuci (2012). "Evolution of Collective Perception in a Group of Autonomous Robots." In: *Computational Intelligence*. Vol. 399. Springer Berlin Heidelberg, pp. 67–80.

Nadizar, G. et al. (2021). "On the effects of pruning on evolved neural controllers for soft robots." In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '21. Lille, France: Association for Computing Machinery, pp. 1744–1752.

Nadizar, G. et al. (2022). "Merging pruning and neuroevolution: towards robust and efficient controllers for modular soft robots." In: *The Knowledge Engineering Review* 37, e3.

Nadizar, G. et al. (2023). "An experimental comparison of evolved neural network models for controlling simulated modular soft robots." In: *Applied Soft Computing* 145, p. 110610.

Nair, A. et al. (2021). "AWAC: Accelerating Online Reinforcement Learning with Offline Datasets." In: *International Conference on Learning Representations (ICLR 2021)*.

Nakajima, K. and I. Fischer (2021). Springer Singapore.

Nakajima, K. et al. (2013). "A soft body as a reservoir: case studies in a dynamic model of octopus-inspired soft robotic arm." In: *Frontiers in Computational Neuroscience* 7.

Nei, M. (2013). *Mutation-Driven Evolution*. 1st ed. Oxford, United Kingdom: Oxford University Press.

Nie, X. et al. (2024). "Online Active Continual Learning for Robotic Lifelong Object Recognition." In: *IEEE Transactions on Neural Networks and Learning Systems* 35.12, pp. 17790–17804.

Niv, Y. et al. (2002). "Evolution of Reinforcement Learning in Uncertain Environments: A Simple Explanation for Complex Foraging Behaviors." In: *Adaptive Behavior* 10.1, pp. 5–24.

Nolfi, S. (1997). "Evolving non-trivial behaviors on real robots: A garbage collecting robot." In: *Robotics and Autonomous Systems* 22.3, pp. 187–198.

Nolfi, S. and D. Floreano (1998a). "Coevolving Predator and Prey Robots: Do "Arms Races" Arise in Artificial Evolution?" In: *Artificial Life* 4.4, pp. 311–335.

— (1998b). "How co-evolution can enhance the adaptive power of artificial evolution: Implications for evolutionary robotics." In: *Evolutionary Robotics*. Vol. 1468. Springer Berlin Heidelberg, pp. 22–38.

Nolfi, S. and D. Marocco (2002). "Evolving robots able to visually discriminate between objects with different size." In: *International Journal of Robotics and Automation* 17.4, pp. 163–170.

Nolfi, S. and D. Parisi (1995). "Evolving non-trivial behaviors on real robots: An autonomous robot that picks up objects." In: *Topics in Artificial Intelligence*. Vol. 992. Springer Berlin Heidelberg, pp. 243–254.

Nygaard, T.F. et al. (2018). "Real-world evolution adapts robot morphology and control to hardware limitations." In: *GECCO '18. Proceedings of the Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, pp. 125–132.

Oliver, G., P. Lanillos, and G. Cheng (2022). "An Empirical Study of Active Inference on a Humanoid Robot." In: *IEEE Transactions on Cognitive and Developmental Systems* 14.2, pp. 462–471.

Ormrod, J.E. (2018). *Human Learning*. 8th ed. New York, NY: Pearson plc.

Parker, L.E., D. Rus, and G.S. Sukhatme (2016). "Multiple Mobile Robot Systems." In: *Springer Handbook of Robotics*. Cham: Springer International Publishing, pp. 1335–1384.

Pini, G. and E. Tuci (2008). "On the design of neuro-controllers for individual and social learning behaviour in autonomous robots: an evolutionary approach." In: *Connection Science* 20.2–3, pp. 211–230.

Piqué, F. et al. (2022). "Controlling Soft Robotic Arms Using Continual Learning." In: *IEEE Robotics and Automation Letters* 7.2, pp. 5469–5476.

Purves, D. et al. (2012). *Neuroscience*. 5th ed. Sunderland, Massachusetts: Sinauer Associates Inc.

Quick, T. et al. (2003). "Evolving Embodied Genetic Regulatory Network-Driven Control Systems." In: *Advances in Artificial Life. 7th European Conference, ECAL 2003, Dortmund, Germany, September 14-17, 2003, Proceedings*. Vol. 2801. Springer Berlin Heidelberg, pp. 266–277.

Quinn, M. (2000). "Evolving co-operative homogeneous multi-robot teams." In: *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*. Vol. 3. Institute of Electrical and Electronics Engineers, pp. 1798–1803.

— (2001). "A comparison of approaches to the evolution of homogeneous multi-robot teams." In: *Proceedings of the 2001 Congress on Evolutionary Computation*. Vol. 1. Institute of Electrical and Electronics Engineers, pp. 128–135.

Quinn, M. et al. (2003). "Evolving controllers for a homogeneous system of physical robots: structured cooperation with minimal sensors." In: *Philosophical Transactions of the Royal Society A* 361, pp. 2321–2343.

Ram, A. et al. (1997). "Case-based reactive navigation: a method for on-line selection and adaptation of reactive robotic control parameters." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 27.3, pp. 376–394.

Roli, A. et al. (2011). "On the design of Boolean network robots." In: *Applications of Evolutionary Computation. EvoApplications 2011: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC, Torino, Italy, April 27-29, 2011, Proceedings, Part I*. Vol. 6624. Springer Berlin Heidelberg, pp. 43–52.

Rothman, D.L., S.C. Stearns, and R.G. Shulman (2021). "Gene expression regulates metabolite homeostasis during the Crabtree effect: Implications for the adaptation and evolution of Metabolism." In: *Proceedings of the National Academy of Sciences* 118.2, e2014013118.

Rybczak, M., N. Popowniak, and A. Lazarowska (2024). "A Survey of Machine Learning Approaches for Mobile Robot Control." In: *Robotics* 13.1, p. 12.

Samuel, D.J. and F. Cuzzolin (2021). "Unsupervised Anomaly Detection for a Smart Autonomous Robotic Assistant Surgeon (SARAS) Using a Deep Residual Autoencoder." In: *IEEE Robotics and Automation Letters* 6.4, pp. 7256–7261.

Sanjuán, R. et al. (2010). "Viral Mutation Rates." In: *Journal of Virology* 84.19, pp. 9733–9748.

Schaal, S. (2002). "Learning Robot Control." In: *The handbook of brain theory and neural networks*. Cambridge, Massachusetts: The MIT Press, pp. 983–987.

Schranz, M. et al. (2020). "Swarm Robotic Behaviors and Current Applications." In: *Frontiers in Robotics and AI* 7, p. 36.

Shaheen, K. et al. (2022). "Continual Learning for Real-World Autonomous Systems: Algorithms, Challenges and Frameworks." In: *Journal of Intelligent & Robotic Systems* 105.1, p. 9.

Silva, F., L. Correia, and A.L. Christensen (2014a). "Speeding Up Online Evolution of Robotic Controllers with Macro-neurons." In: *Applications of Evolutionary Computation*. Vol. 8602. Springer Berlin Heidelberg, pp. 765–776.

— (2015a). "A Case Study on the Scalability of Online Evolution of Robotic Controllers." In: *Progress in Artificial Intelligence*. Vol. 9273. Springer International Publishing, pp. 189–200.

— (2016). "Leveraging Online Racing and Population Cloning in Evolutionary Multirobot Systems." In: *Applications of Evolutionary Computation*. Vol. 9598. Springer International Publishing, pp. 165–180.

— (2017). "Evolutionary online behaviour learning and adaptation in real robots." In: *Royal Society Open Science* 4, p. 160938.

Silva, F., P. Urbano, and A.L. Christensen (2012a). "Adaptation of Robot Behaviour through Online Evolution and Neuromodulated Learning." In: *Advances in Artificial Intelligence – IBERAMIA 2012. 13th Ibero-American Conference on AI, Cartagena de Indias, Colombia, November 13-16, 2012, Proceedings*. Vol. 7636. Springer Berlin Heidelberg, pp. 300–309.

— (2014b). "Online Evolution of Adaptive Robot Behaviour." In: *International Journal of Natural Computing Research* 4.2, pp. 59–77.

Silva, F. et al. (2012b). "odNEAT: An Algorithm for Distributed Online, Onboard Evolution of Robot Behaviours." In: *ALIFE 2012. The Thirteenth International Conference on the Synthesis and Simulation of Living Systems*. The MIT Press, pp. 251–258.

Silva, F. et al. (2015b). "odNEAT: An Algorithm for Decentralised Online Evolution of Robotic Controllers." In: *Evolutionary Computation* 23.3, pp. 421–449.

Silver, D. et al. (2016). "Mastering the game of Go with deep neural networks and tree search." In: *Nature* 529.7587, pp. 484–489.

Sims, K. (1994). "Evolving 3D Morphology and Behavior by Competition." In: *Artificial Life* 1.4, pp. 353–372.

Singh, B., R. Kumar, and V.P. Singh (2022). "Reinforcement learning in robotic applications: a comprehensive survey." In: *Artificial Intelligence Review* 55.2, pp. 945–990.

Soltoggio, A. et al. (2008). "Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios." In: *Artificial Life XI. Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*. The MIT Press, pp. 569–576.

Sommer, R.J. (2020). "Phenotypic Plasticity: From Theory and Genetics to Current and Future Challenges." In: *Genetics* 215.1, pp. 1–13.

Sutton, R.S. and A.G. Barto (2018). Cambridge, Massachusetts – London, England: The MIT Press.

Szpirer, J., D.G. Ramos, and M. Birattari (2024). "Automatic design of robot swarms that perform composite missions: an approach based on inverse reinforcement learning." In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Institute of Electrical and Electronics Engineers, pp. 5791–5798.

Trianni, V. et al. (2003). "Evolving Aggregation Behaviors in a Swarm of Robots." In: *Advances in Artificial Life. 7th European Conference, ECAL 2003, Dortmund, Germany, September 14-17, 2003, Proceedings*. Vol. 2801. Springer Berlin Heidelberg, pp. 865–874.

Tuci, E. et al. (2008). "Evolving Homogeneous Neurocontrollers for a Group of Heterogeneous Robots: Coordinated Motion, Cooperation, and Acoustic Communication." In: *Artificial Life* 14.2, pp. 157–178.

Uriot, T. et al. (2022). "Spacecraft collision avoidance challenge: Design and results of a machine learning competition." In: *Astrodynamics* 6.2, pp. 121–140.

Urzelai, J. and D. Floreano (1999). "Incremental Evolution with Minimal Resources." In: *First International Khepera Workshop (IKW'1999)*.

— (2001). "Evolution of Adaptive Synapses: Robots with Fast Adaptive Behavior in New Environments." In: *Evolutionary Computation* 9.4, pp. 495–524.

Urzelai, J. et al. (1998). "Incremental Robot Shaping." In: *Connection Science* 10.3-4, pp. 341–360.

Van de Maele, T. et al. (2021). "Active Vision for Robot Manipulators Using the Free Energy Principle." In: *Frontiers in Neurorobotics* 15.

Ven, G.M. van de, T. Tuytelaars, and A.S. Tolias (2022). "Three types of incremental learning." In: *Nature Machine Intelligence* 4.12, pp. 1185–1197.

Wang, D., H. Deng, and Z. Pan (2020). "MRCDRL: Multi-robot coordination with deep reinforcement learning." In: *Neurocomputing* 406, pp. 68–76.

Wang, L. et al. (2024). "A Comprehensive Survey of Continual Learning: Theory, Method and Application." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.8, pp. 5362–5383.

Watkins, C.J.C.H. and P. Dayan (1992). "Q-learning." In: *Machine Learning* 8.3, pp. 279–292.

Wen, Y. et al. (2020). "Online Reinforcement Learning Control for the Personalization of a Robotic Knee Prosthesis." In: *IEEE Transactions on Cybernetics* 50.6, pp. 2346–2356.

Williams, G.C. (2019). *Adaptation and Natural Selection: A Critique of Some Current Evolutionary Thought*. 2nd ed. Princeton, New Jersey, United States: Princeton University Press.

Ye, H. et al. (2024). "Person Re-Identification for Robot Person Following With Online Continual Learning." In: *IEEE Robotics and Automation Letters* 9.11, pp. 9151–9158.

Zhang, K. et al. (2020). "Continuous reinforcement learning to adapt multi-objective optimization online for robot motion." In: *International Journal of Advanced Robotic Systems* 17.2, p. 1729881420911491.

Zheng, K. et al. (2024). "Adaptive collision avoidance decisions in autonomous ship encounter scenarios through rule-guided vision supervised learning." In: *Ocean Engineering* 297, p. 117096.

Part II

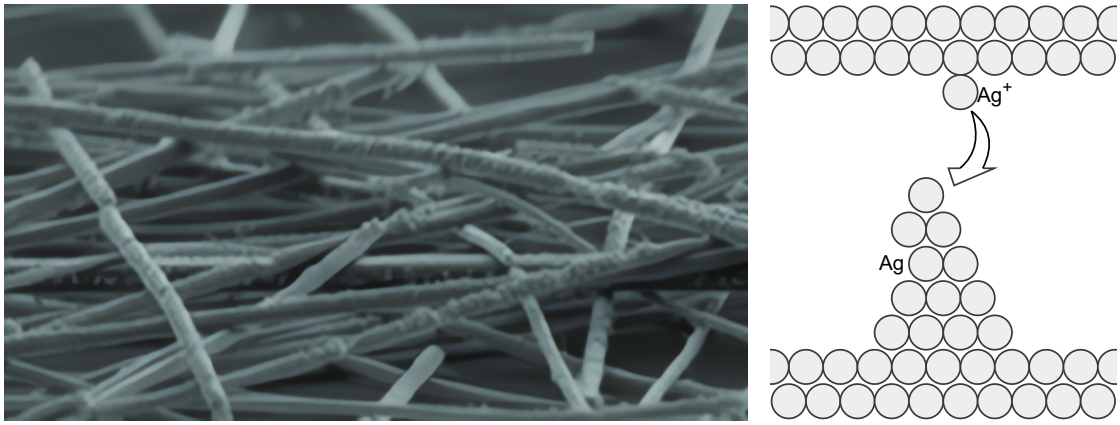
Experiments on adaptation

2 Static conditions

Online adaptation is a relatively new methodology for the creation of robot control software that has not yet received significant validation. In order to understand its potential utility, we first need to assess whether or not it is a suitable approach to learn desired behaviors. Therefore, we start our investigation by proposing an adaptive system able to work on minimalistic robots with reduced computational power and memory. This permits focusing on fundamental properties of adaptation, that could be shrouded by too advanced computational capabilities. Additionally, we need to start formalizing some characteristics and requirements related to the use and analysis of online adaptation. With this goal in mind, we start focusing on a relatively simple case: the adaptation of control software to perform missions in static, unknown conditions. This type of situation occurs when we know the task we want to achieve but not the environment in which the robot will act. For simplicity, in this case, we assume the environment to be static.

This chapter leverages the results obtained in Baldini et al. (2023a) and Baldini et al. (2023b), comparing the effectiveness of adaptation with respect to a random search algorithm. We begin by describing the system used to control the robots, the adaptation mechanism, the evaluation tasks, and the experimental setting. Then, we present and discuss the results, drawing some considerations for subsequent explorations on online adaptation.

Nanowire network and its working principle



(a) Nanowire network microscopy picture **(b) Silver bridge formation**

Figure 2.1: (a) picture of a nanowire network and (b) formation of an ion-silver bridge. Picture (a) is taken from Milano et al. (2020) by courtesy of the authors.

2.1 Controller and adaptive mechanism

In this chapter, we start exploring the properties of online adaptation; we do so by experimenting with minimalistic robots. Drawing inspiration from *in materia* computing (Ricciardi et al., 2022; J.F. Miller et al., 2002; J. Miller et al., 2014), we opt to use control systems based on physical materials, that can be computationally powerful, small, run on low energy, and even cheap to produce. Specifically, the controller we use in these experiments leverages a nanowire network, a promising device for all the contexts that require complex computations and have strict consumption constraints, like edge computing and robotics (Milano et al., 2020).

Nanowire networks are a novel kind of nanoscale electrical circuits composed of nanometric silver wires (see Figure 2.1a) that display neuromorphic behavior (Jo et al., 2010; Milano et al., 2023). This is given by their self-organizing property and intrinsic structural and functional plasticity, mimicking biological networks (Milano et al., 2020; Milano et al., 2022). The similarity resides in what is known as Hebbian theory (Hebb, 1949), a property that is commonly summarized as "neurons wire together if they fire together" (Löwel et al., 1992; Milano et al., 2019). The artificial equivalent of synaptic strengthening is here represented by the behavior of the ion-silver bridges connecting the wires of the network (see Figure 2.1b). When subject to a

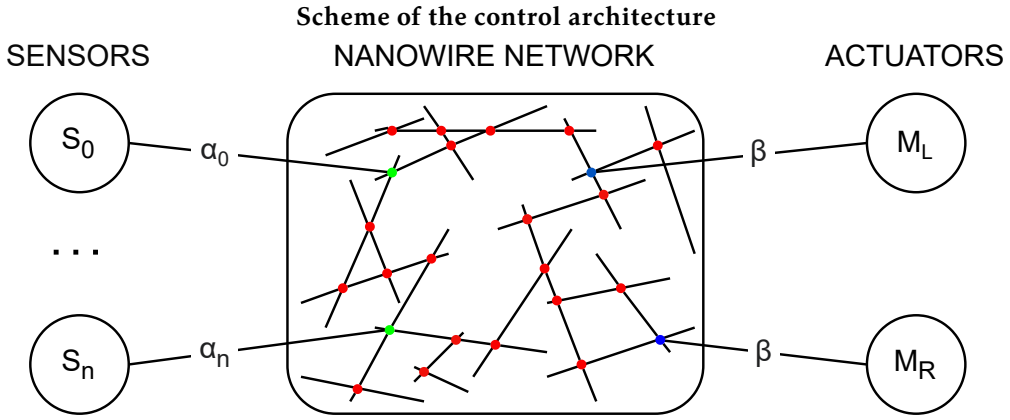


Figure 2.2: Schematic representation of the control architecture. $[S_0, ..., S_n]$ represent robot sensors; M_L, M_R represent left and right motors. $[\alpha_0, ..., \alpha_n]$ are the weighting factors, different for each sensor signal. β represents the influence of the motor resistance in the electrical equivalent system.

voltage difference, the ions aggregate in the junction increasing its conductance. When the stimulus is removed, the network slowly returns to its stable state. This dynamic rules the network plasticity and can be seen as a short-term memory useful to process spatio-temporal data (Demis et al., 2016; Fu et al., 2020; Christensen et al., 2022). In this investigation, we consider the voltage stimulation to be the only way to modify the internal state of the nanowire network.

The controller we propose works by perturbing the nanowire network state and by extracting output signals to control the motion of the robot. To do so, we connect the sensors and actuators to different nanowires of the network (see Figure 2.2). The sensors perturb the nanowire network state with a signal proportional to the perception, inducing a voltage change on output nanowires. We refer to the sensor-to-nanowire and nanowire-to-actuator connections as *couplings*. No coupling insists on the same nanowire, so that inputs do not overwrite each other signal and outputs consist of distinct values. Additionally, we force all the inputs to be at least two nanowires away from the outputs, so as to avoid control without processing[1]. We assign a weight to each input coupling, possibly enhancing or reducing the signal strength. This enables discriminating sensors, enhancing or reducing their influence in the computation without the need to affect the nanowire network directly. An example is that of the frontal sensors in a collision avoidance task, that are more important than the back ones in achieving the desired

[1] As it happens, for instance, in Braitenberg architectures (Braitenberg, 1986).

task	initialization		update		threshold	epoch	epochs	replicas
	μ	σ	μ	σ		duration	count	count
area avoidance	1	2.50	0	1.0	70	100 s	300	250
collision avoidance	1	5.00	0	2.5	40	20 s	300	250
T-maze	1	2.50	0	1.5	15	50 s	300	150
foraging	1	2.50	0	2.5	0	150 s	500	150

Table 2.1: Configuration parameters of each experiment.

behavior. Alternatively, this might help to balance the influence of different types of perceived properties (e.g., proximity, light intensity, temperature, etc.). The weights are selected from a normal distribution with $\mu = 1$ and σ depending on the task (see Table 2.1). We also assign a weight to each motor representing its load, and thus its influence, on the equivalent electrical circuit of the control system. This means that this weight is not used to enhance or reduce the output value directly, but instead it modifies the voltage distribution across the network. This value can be set by adding a resistance in series to the motor; however, it is realistic that it remains unchanged after being set at the start of the experiment. We use a value of 1 kΩ in area avoidance and T-maze tasks, and of 10 kΩ in collision avoidance and foraging tasks.

The initial controller consists of a set of couplings connected to randomly selected nanowires with corresponding random input weights. The robot runs the controller and collects a performance measure, representing how well it behaved during the evaluation epoch. Subsequently, the robot undergoes an adaptation process affecting the couplings. In this investigation, we consider two different adaptation strategies: (i) random and (ii) local. The *random adaptation* generates a new set of input couplings and weights at each epoch. This means that it does not take advantage of previously discovered controllers, but instead it searches stochastically in the entire solution space. The *local adaptation* searches instead for a solution similar to that of the best controller found since the start of the experiment, if greater than a minimum threshold, otherwise the strategy generates a completely new controller. The threshold avoids wasting adaptation cycles over poorly behaving controllers, and its value depends on the task and corresponding evaluation function (see Table 2.1). This delay in local adaptation caused by the threshold can be seen as a bootstrap phase employing random adaptation.

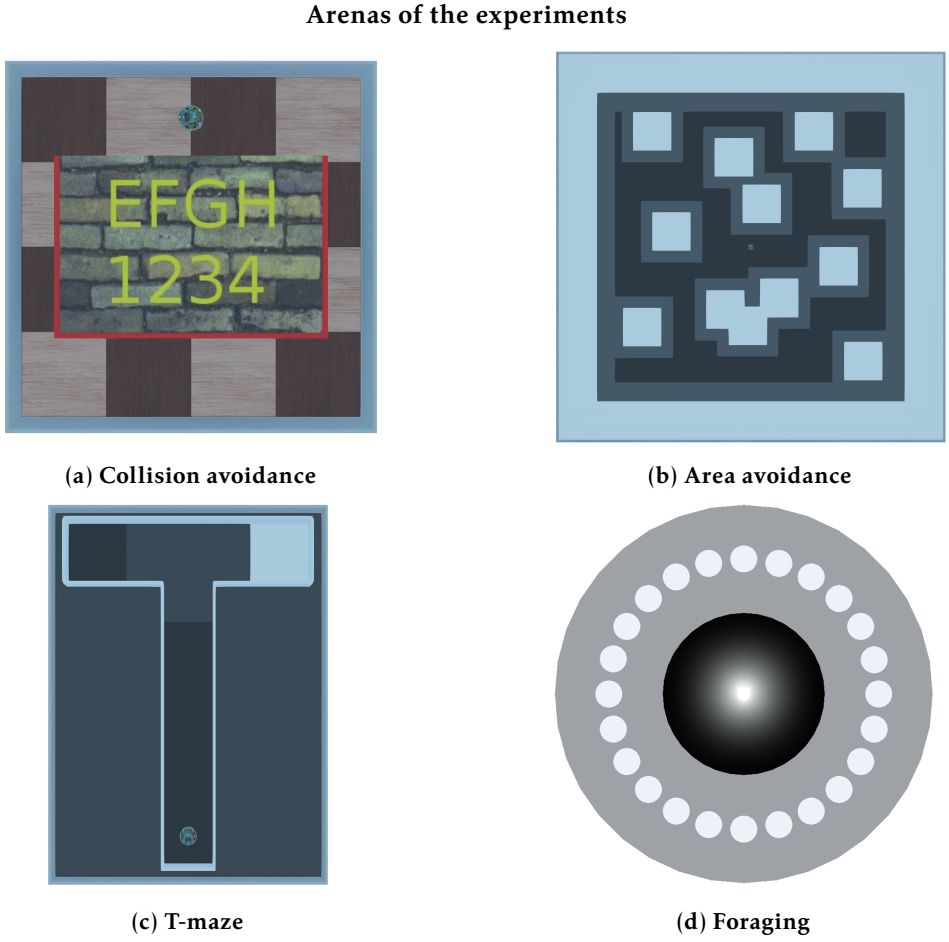
In order to search for a behavior similar to that of the best controller, the robot selects 10 to

40 percent of its total input couplings and reconnects them to different points of the nanowire network. Additionally, it modifies their weights, potentially enhancing or reducing the input signal strength and thus balancing the influence of specific sensors in the computation. This is done by adding to the selected weight a random value sampled from a normal distribution with $\mu = 0$ and σ depending on the task (see Table 2.1). This procedure is possible because the robot memorizes the set of couplings and weights of the best controller found since the start of the experiment, according to the evaluation function. The proposed strategy is a variant of a methodology used for the adaptation of robots controlled by Boolean networks (Braccini et al., 2022c). The novelty resides in the weighting of the input signals, in the use of thresholds to postpone the adaptation, and in the use of nanowire networks. Additionally, this variant takes advantage of the analogic functioning of the nanowire network, that adds both complexity and potential compared to Boolean networks.

2.2 Experimental setting

In order to evaluate the ability to produce suitable behaviors, we test the adaptive system on multiple tasks: (i) collision avoidance; (ii) area avoidance; (iii) T-maze; (iv) foraging. Each task takes place in a specific arena (see Figure 2.3). The robot used for the experiments is an *e-puck* (Mondada et al., 2009). All the experiments run within the Webots simulator, version R2021b (Michel, 1998; Michel, 2004), and consist in the adaptation of multiple unique nanowire networks: the experimental replicas. Each experiment uses a different configuration of parameters depending on the task complexity, on the arena size, and on qualitative analyzes (see Table 2.1 and Table 2.2). Specifically, we select the distributions from which to sample and modify the weights of the input couplings, the number and duration of adaptation epochs, and the size of the nanowire network. The idea is that more complex or harder-to-evaluate tasks require more adaptation time and greater computational power (i.e., larger nanowire networks). Finally, we limit the number of replicas in computationally expensive tasks.

We define an evaluation function for each task, indicating how well the robot performs at it: the higher, the better. This drives the adaptation and permits selecting the best controllers to adapt. In the following, we describe each task and the corresponding evaluation function.



(a) Collision avoidance

(b) Area avoidance

(c) T-maze

(d) Foraging

Figure 2.3: The corresponding arena of each task: (a) collision avoidance, (b) area avoidance, (c) T-maze, and (d) foraging. The T-maze arena presents a dual version in which the colored area at the beginning of the maze is white rather than black.

task	package size	mean nanowire length	network density
area avoidance	50	10.00	5.00
collision avoidance	50	10.00	7.50
T-maze	125	14.00	5.00
foraging	125	14.00	5.00

Table 2.2: Parameters of the nanowire networks used to control the robots. The number of nanowires and junctions depends on the size of the package, the mean length of the nanowires, and the network density.

2.2.1 Collision avoidance

In the collision avoidance task, the robot navigates the arena as fast and straight as possible while avoiding collisions with obstacles. To fulfil this goal, the robot needs to perceive and

react in the environment. It does so by relying on 8 proximity sensors and 2 wheeled motors. The evaluation function also relies on this information to evaluate the behavior, according to the following formula:

$$P^{C.A.}_{epoch} = 100 \cdot \frac{1}{k} \sum_{i=1}^{k} P^{C.A.}_{step,\ i} \tag{2.1}$$

$$P^{C.A.}_{step} = (1 - \sqrt{p_{max}}) \cdot (1 - |v_l - v_r|) \cdot \frac{v_l + v_r}{2} \tag{2.2}$$

where:

- k is the number of steps per epoch;
- i is the index of the step under consideration;
- $p_{max} \in [0, 1]$ is the normalized obstacle proximity, with 0 indicating a far object, and 1 indicating a near one;
- $v_l, v_r \in [0, 1]$ are respectively the normalized left and right motor speeds, with 0 representing an anticlockwise revolution, 0.5 a still state, and 1 a clockwise revolution.

Equation 2.2 considers the speed, the direction, and the proximity to obstacles of the robot. The former two promote a fast movement in a straight trajectory, avoiding excessive turns. The latter penalizes robots staying near to obstacles. We try to give more importance to this latter goal by putting the proximity value under the square root, so as to increase its weight in the equation[2].

The arena of the collision avoidance task consists of a central obstacle and a set of external walls (see Figure 2.3a). The result is a circuit in which the robot has to run while avoiding collisions. The passages on the left and right are stricter than the top and bottom ones, adding a small complexity to the task.

2.2.2 Area avoidance

The area avoidance task is similar to the collision avoidance, but requires the robot to avoid virtual forbidden areas identified only by the ground color. The obstacles stop being physical and become instead intangible. This complicates the tasks because the robot has to react faster. A slow response pushes the robot indeed deep into the forbidden area, collecting a considerable

[2]The square root "increases" values in the range [0, 1], such as those produced by proximity sensors.

step penalty and reducing the performance attainable during the epoch. In this task, the robot moves by means of two wheeled motors and perceives the environment by means of a single ground sensor, attempting to show that it is possible to obtain a wandering behavior and still avoid obstacles with a computationally limited system. The evaluation function for the area avoidance task is the following:

$$P^{A.A.}_{epoch} = \frac{1}{1.01} \cdot \left(100 + \frac{1}{k} \sum_{i=1}^{k} P^{A.A.}_{step,\, i}\right) \tag{2.3}$$

$$P^{A.A.}_{step} = (1 - |v_l - v_r|) \cdot \frac{v_l + v_r}{2} - 100 \cdot c \tag{2.4}$$

where:

- k is the number of steps per epoch;
- i is the index of the step under consideration;
- $c \in \{0, 1\}$ is 1 if the robot is on the illegal area, 0 otherwise;
- $v_l, v_r \in [0, 1]$ are respectively the normalized left and right motor speeds, with 0 representing an anticlockwise revolution, 0.5 a still state, and 1 a clockwise revolution.

Equation 2.4 strongly penalizes a robot hovering an illegal area. This is independent of the direction of the robot or its speed. The goal is mainly to promote behaviors that avoid illegal areas, with its quality depending on the speed and direction of the movement. Due to the presence of the penalty factor, the range of performance per step is $[-100, 1]$. In order to put it back in $[0, 100]$, Equation 2.3 sums 100 to the average performance and then divides the result by 1.01.

The arena for the area avoidance task consists of a few illegal areas that the robot should not hover (see white blocks in Figure 2.3b). Additionally, it is limited by the presence of virtual and physical borders, assigning a deserter robot negative scores while preventing it from going too distant. The goal is to permit every configuration to reach the legal area, even if it starts in a disadvantageous position. Finally, every illegal area is surrounded by a neutral zone that does not give any penalty but informs the robot that it is approaching a forbidden space.

2.2.3 T-maze

In the T-maze task, the robot navigates a T-shaped maze to reach the correct end-point (see Figure 2.3c). The goal destination depends on the color of the floor at the start of the run: right for black, left for white. Nevertheless, after leaving the starting area, the robot loses track of the indication of direction. Therefore, the controller should use the cue to bias the behavior to correctly navigate the maze. This could be achieved by exploiting the nanowire network plasticity to memorize the initial input and to behave accordingly also when it stops being perceived. In order to test the behavior, the robot is periodically kidnapped and placed back at the start of the maze. In this task, the robot perceives the environment through 8 proximity sensors and 1 ground sensor, and acts by means of 2 wheeled motors. The evaluation function for the T-maze task is the following:

$$P^{T.M.}_{epoch} = 100 \cdot \frac{1}{3} \cdot \left(1 + \frac{1}{k} \sum_{i=1}^{k} P^{T.M.}_{step,\, i}\right) \tag{2.5}$$

$$P^{T.M.}_{step} = 2 \cdot (1 - \alpha) \cdot (1 - \beta) - \alpha \tag{2.6}$$

where:

- k is the number of steps per epoch;
- i is the index of the step under consideration;
- $\alpha \in \{0, 1\}$ is 1 if the ground color is the same as the starting one;
- $\beta \in \{0, 1\}$ is 1 if the ground is gray.

Equation 2.6 considers the presence in the correct and wrong areas to bestow prices and penalties. The performance decreases if the robot remains in the starting area or if it reaches the wrong endpoint. Instead, if the robot reaches the correct endpoint, its score increases by 2 at each step of permanence. This helps in slightly reducing the simulation time. Additionally, the evaluation function also highlights the presence of a gray color. This is the color of the ground outside the starting and ending points. In this region, the fitness of the robot does not change. Equation 2.6 does not explicitly consider the speed of the robot. Nevertheless, the design of the task in combination with the epoch duration implicitly requires the robot to move fast: in the opposite case, it would not have enough time to reach the end point and to balance the penalty

collected at the start of the epoch. The range of performance per step in the T-maze task is $[-1, 2]$. In order to put it in $[0, 100]$, Equation 2.5 sums 1 to the average performance, divides the result by 3, and then multiplies it by 100.

2.2.4 Foraging

Foraging is a classical robotic task that requires the agent to find and bring prey to the nest. Thin white plates homogeneously distributed in the arena represent the prey that the robot has to collect. They are passive, meaning that they do not have any active behavior and can be moved only by other entities. The robot perceives the environment through 8 light sensors and 2 ground sensors, one of which is negated. It acts in the environment by means of 2 wheeled motors and 1 virtual gripper. The former are controlled by continuous values, while the state of the latter changes if the control signal from the nanowire network exceeds a given threshold[3].

The evaluation function drives the adaptation in order to select behaviors that capture prey in the hunt field and subsequently release them on the nest. This has access to the same perception of the robot and to the control output that it uses to command the actuators. We define the evaluation function for the foraging task as follows:

$$P^{\,F.O.}_{epoch} = \sum_{i=1}^{k} P^{\,F.O.}_{step,\,i} \tag{2.7}$$

$$P^{\,F.O.}_{step} = \begin{cases} 50 & if \quad A \wedge \neg\, \Theta_i \wedge f(p) \\ 100 & if \quad A \wedge \Theta_i \wedge f(n) \wedge g(p) \\ -50 & if \quad A \wedge \Theta_i \wedge f(h) \wedge g(p) \\ 0 & otherwise \end{cases} \tag{2.8}$$

where:

- k is the number of steps per epoch;
- i is the index of the step under consideration;
- $A \in \{False, True\}$ is $True$ if the gripper state changed, $False$ otherwise;

[3]We choose a threshold of 0.5 for an output in the range $[0, 1]$.

- $\Theta \in \{False, True\}$ is *True* if the gripper is currently open, *False* otherwise;
- $f(x)$ is a function that returns *True* if the color of the ground before the gripper action was equal to the parameter x, *False* otherwise;
- $g(x)$ is a function that returns *True* if the color of the ground after the gripper action is equal to the parameter x, *False* otherwise;
- p, h, n are respectively the colors perceived when hovering a *prey*, *hunt field*, and *nest*.

The given policy defines a capture as the closure of the gripper while the robot is on a white ground (i.e., the prey). Differently, it identifies a release by looking at the color of the floor before and after the gripper opens. If, after the release, the color is different from white, then there has not been any release. It is important to underline that the robot (and therefore the evaluation function) does not know if the gripper is open or close, nor if it is carrying a prey. Indeed, there is no feedback from the actuators to the control system, and assumptions can be made only by looking at how actions influence the environment.

Another aspect that emerges from the evaluation function is that a release in the hunt field is penalized by the same amount used to reward a capture. This prevents both increments and decrements of the performance in case of a *capture-release* sequence outside the nest. Direct and interleaved prey deliveries will therefore have the same final evaluation. The only difference may consist in the total delivery time, possibly permitting additional captures to the fastest behavior. This choice prevents discarding acceptable solutions, despite permitting the perpetuation of possibly inefficient ones. In the long run, the expected result is the reduction of inefficient sequences in order to increase the delivery speed.

It is important to highlight that a capture in the nest is not possible; indeed, the prey vanishes almost immediately after the release. The permanence time before disappearing is just sufficient for performance calculation and does not permit a second capture.

The arena of the foraging task is composed of two areas: (i) the nest; (ii) the hunt field. The nest is where the captured prey shall be brought, while the hunt field is where the prey are initially located. In our implementation, we use a circular arena with an external hunt field and an internal nest (see Figure 2.3d). Each area is characterized by a different color that the robot can distinguish through its ground sensors. A light source is positioned at the center of

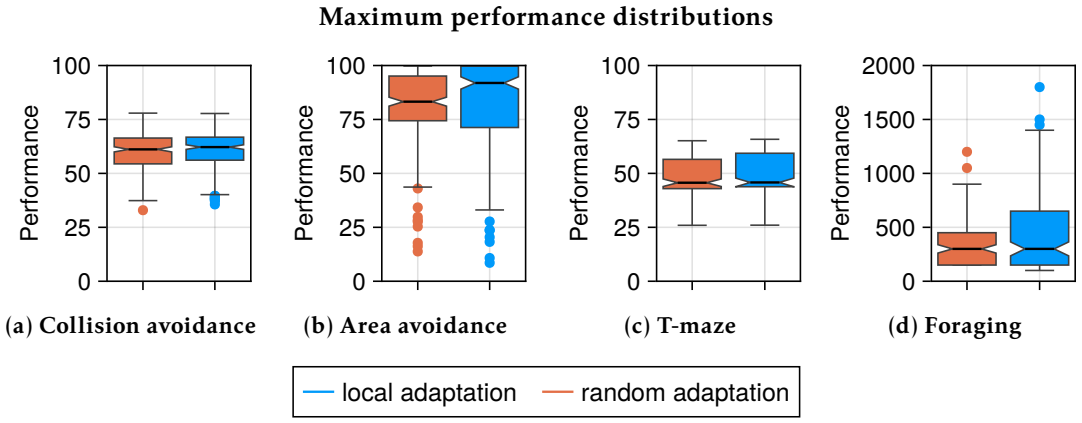


Figure 2.4: Distribution of the maximum performance attained by each replica in the four tasks. The performance for (a), (b), and (c) is expressed as the percentage of the maximum performance attainable. The performance for (d) does not have an easily computable and informative upper bound, and it is thus displayed raw.

the arena, providing a landmark that the robot can perceive through its light sensors.

Finally, in the foraging task, the local adaptation strategy does not make use of the threshold. Indeed, preliminary analyzes did not show any advantage in using this bootstrap phase in this task. We claim that this is because intermediate scores are difficult to obtain, and good and bad behaviors are too clearly separated. This also complicates the adaptation, that can hardly follow a gradient of performance to improve the behavior.

2.3 Results

As this chapter investigates the use of online adaptation to create flexible controllers for unknown static environments, we begin analyzing the *maximum* performance attained by the robot throughout the experiment. Figure 2.4 shows the results in the collision avoidance, area avoidance, T-maze, and foraging tasks. It is visible that the random and local adaptation attain similar performance distributions, with the latter performing just slightly better. The motivation is that they both explore the landscape of solutions similarly: the former tests random controllers at each iteration, while the bootstrap phase of the latter permits exploring until a good enough solution is found. Additionally, this suggests that many possible solutions to the problem exist. Indeed, in the opposite case, we would expect the random adaptation to have

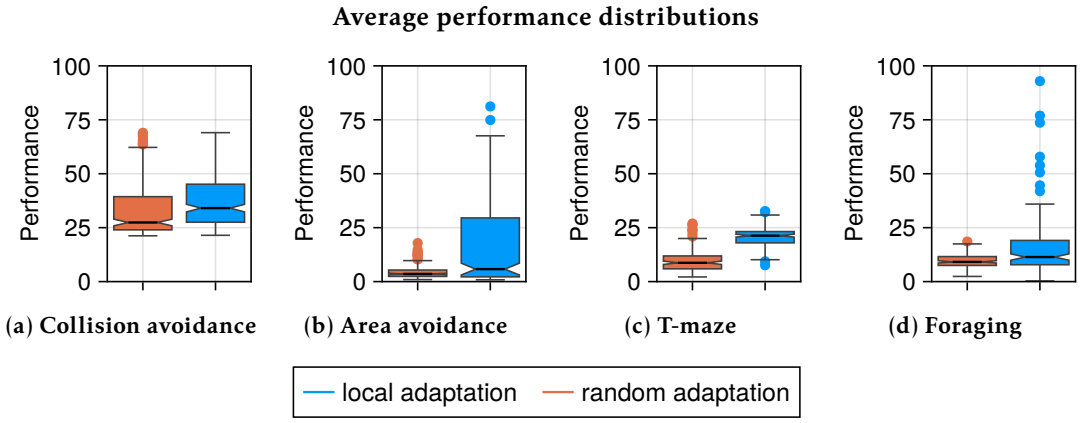


Figure 2.5: Distribution of the average performance attained by each replica in the four tasks. The performance for (a), (b), and (c) is expressed as the percentage of the maximum performance attainable. The performance for (d) does not have an easily computable and informative upper bound, and it is thus displayed raw.

more trouble finding them, as it cannot exploit the gradient of the landscape during the search. Overall, we can say that online adaptation seems to be capable of often finding effective solutions, frequently attaining 50% or more of the maximum performance, which, based on direct observations, already indicates an acceptable behavior.

The maximum performance attained is useful to understand how effectively the mechanism explores the landscape of solutions. Nevertheless, this does not tell us how well it maintains and improves discovered controllers during the experiment. Indeed, the best performance could be a unique achievement in the life of the robot, not indicating systematic effectiveness. To investigate this aspect, we examine the distribution of the *average* performance during the run of each replica (see Figure 2.5). From this analysis, we observe some interesting results that differ from task to task. The performance in collision avoidance halves, but still remains at a mean level representing the ability to perform some basic obstacle avoidance[4] (see Figure 2.5a). Additionally, the top performing solutions (i.e., the best behaving ones) experience just a minimal decrease. Comparing random and local adaptation, we observe that the difference increases slightly in favor of the latter, causing the distributions to differ statistically (p-value < 0.05 with the Wilcoxon test). Meanwhile, in the area avoidance, the performance decreases significantly, dropping from an almost perfect behavior to a barely decent one even with local adaptation (see

[4]Usually, a performance around 30% of the maximum indicates an overall decent behavior.

Figure 2.5b). This clearly shows the difficulty of the task, that requires relying on just one input sensor to control the motion. Nevertheless, the local adaptation succeeds in maintaining some replicas working, completely outperforming the random adaptation (p-value < 0.05 with the Wilcoxon test). This trend is similar also in the T-maze task (see Figure 2.5c). In this case, the mean performance attained by local adaptation throughout the experiment halves, with the one attained by the random adaptation almost zeroing. Notably, the top performances of the latter are lower than the third quartile of the former. Also in this case, the distributions differ significantly (p-value < 0.05 with the Wilcoxon test). Finally, in the foraging task, we notice a very substantial drop, hidden by the change in scale of the plot (see Figure 2.5d). Overall, the performances seem to decrease by more than 90%. We claim that this is due to the complexity of the task, that requires a long exploration time in order to find a suitable solution. Consequently, the initial epochs of the search contribute almost negligibly to the overall performance. Still, the two distributions are statistically different (p-value < 0.05 with the Wilcoxon test), with the local adaptation dropping less than the random one.

In order to verify our claim about the drop between the maximum and average performance of the experiment, especially in the foraging task, we decide to investigate the average performance *throughout* the experiment. To do so, we average the performance of all the replicas at each epoch, so as to produce a curve displaying the overall trend of performance over time. Additionally, we smooth the curve by averaging it over a 25-epoch window: this helps in understanding the general trend while ignoring noise. In this analysis, due to computational constraints, we consider only the T-maze and foraging tasks. The results clearly highlight the difference between random and local adaptation, with the latter trend often increasing and performing better than the former (see Figure 2.6). This is clear in the foraging task, in which the local adaptation displays a growing trend while the random adaptation does not improve at all. Instead, the trend of the T-maze task is peculiar: here, the local adaptation increases the performance in the first few epochs and then stagnates. We hypothesize that this is due to the difficulty of the task, with many controllers succeeding in reaching one of the correct destinations but very few reaching both. Additionally, working controllers are probably sensitive to changes, as even successful epochs are not often replicated (i.e., the local adaptation does not find other valid derived controllers). Overall, analyzing the performance over time gives us additional infor-

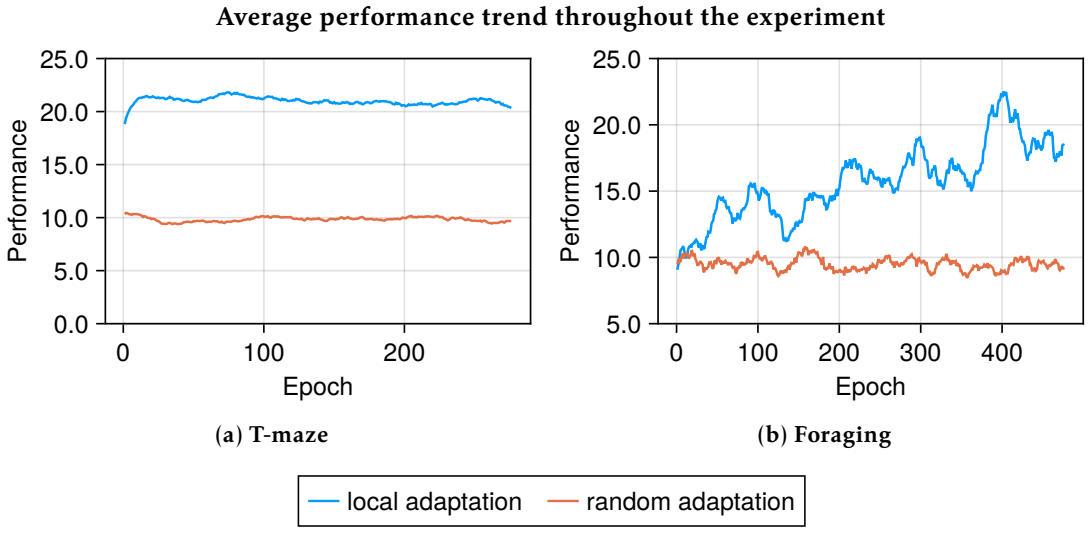


Figure 2.6: Average performance trend throughout the experiment in the (a) T-maze and (b) foraging tasks. We smooth away the noise by substituting each epoch-value with its average with the 24 subsequent epochs.

mation about the behavior of the two methods, permitting to explain the previously witnessed distributions better.

We noticed that the trend of the T-maze performance over time is quite peculiar. In order to better understand this type of behavior, we propose a different way to analyze the results. Specifically, we propose to evaluate the controllers and the adaptation in function of *achievements*, and not of the performance itself. This means counting how many "goals" the replica attains in its life. Nevertheless, this type of analysis cannot be done for every type of task. For instance, classifying goals in a collision avoidance task is quite pretentious. Therefore, we just focus on the T-maze and foraging tasks, that have the most straightforward way to calculate achievements. For the former, we consider an achievement reaching both correct destinations in one epoch. For the latter, we consider an achievement the foraging of at least one prey. Figure 2.7 shows the "empirical cumulative distribution function" (eCDF), indicating the ratio of replicas achieving a number of successes lower than or equal to a certain value of the x-axis. Both mechanisms have almost the same share of replicas completely failing (i.e., attaining 0 successes), indicating a possible difficulty of the tasks or the impossibility of producing the desired behavior with some nanowire networks. Of the remaining, only a few are able to find multiple

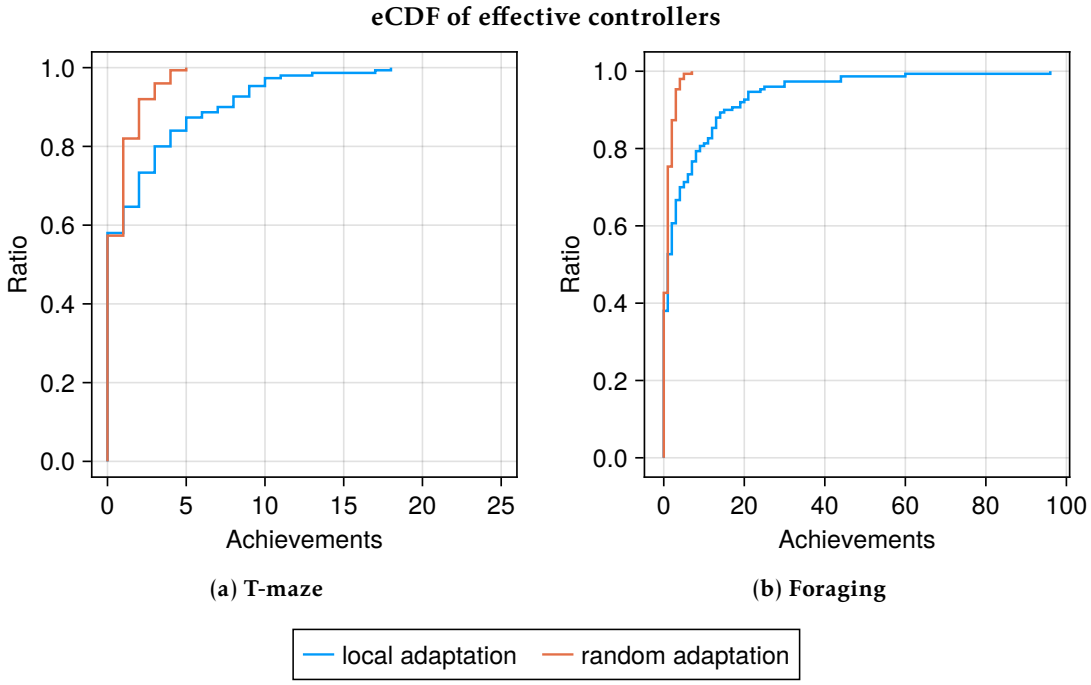


Figure 2.7: Empirical cumulative distribution function. It indicates the ratio of replicas finding a number of effective controllers lower than or equal to the value indicated by the x-axis in (a) the T-maze and (b) the foraging tasks. In the T-maze, a successful controller is one that succeeds in reaching both endpoints of the maze. In the foraging, a successful controller is one bringing back to the nest at least one prey.

working controllers (indicated by values greater than 1 on the x-axis), supporting the thesis of task complexity. Overall, we can see that, once it finds a working controller, local adaptation is able to produce working variations more frequently than random adaptation. This is visible from the gap between the blue and orange trends, with lower values indicating a larger share of replicas attaining a number of successes greater than that indicated on the x-axis. Furthermore, since the random adaptation reaches 1 "earlier" than the local one, we can state that the former finds a much more limited number of working controllers (e.g., the maximum found in one of its replicas of the T-maze task is 5). We conclude that modifying the best controller helps to improve and maintain the performance, but also that random search is not effective enough as a bootstrap strategy[5], permitting to find only around 50% successful controllers.

[5]We recall that the local adaptation searches randomly until it finds a controller that attains a performance greater than the selected threshold.

2.4 Discussion

In this chapter, we propose two adaptation mechanisms and use them for adapting to static unknown environments. This application scenario is the simplest, requiring only the development of a controller for a specific task in a specific arena. The optimal behavior and its performance thus remain fix throughout the experiment, permitting the adaptive mechanism to simply maximize the performance. Differently, it should consider the performance attainable in a specific instance of time and space, and adapt accordingly. This assumption is perfect to investigate whether the mechanisms we propose have the potential to effectively adapt the system to complete a specific task. Therefore, in this chapter, we focus on how these mechanisms explore the solution space, discussing their effectiveness and the best way to evaluate them.

The first measure we propose for assessing the exploratory capabilities of the adaptive mechanisms is the *maximum* performance attained. As a general consideration, we can state that attaining high values in this measure suggests an effective exploration, while attaining low values suggests an ineffective exploration. Our results show high distributions for both the local and random adaptation, indicating that they explore the solution space quite effectively. Nevertheless, we also observe that the former is higher than the latter in three out of four tasks (i.e., area avoidance, T-maze, and foraging; p-value < 0.05 with the Wilcoxon test). This indicates that the way the local adaptation explores the solution space is more effective than the way the random adaptation does it. Considering that the local adaptation can follow a gradient of performance thanks to the modification to the best controller, we can assume, therefore, that the landscape of solutions in the tasks is smooth but not flat. We claim that this is due to the fact that, in most of these tasks, the perception of the robot is highly correlated, permitting different sensors to be used for the same role. Furthermore, the working principles of nanowire networks (Caravelli et al., 2023) suggest that small changes in the couplings cause small changes in the network state. We claim that these characteristics of the solution space are reasonably common in many tasks, and thus that the local adaptation should generally have an advantage over the random one.

Despite being informative, considering only the maximum performance attained is misleading. Our application scenario is that of a robot continuously modifying its controller in production, and thus needing to maintain *overall* good performance. For this motivation, we propose to

analyze the *average* performance during the run, so as to understand how effectively it behaves and how many duties it can complete. Both mechanisms display a significant drop in performance when compared to the maximum one. This is because they focus on exploring possible solutions with the goal of finding the best one in a limited amount of time. Nevertheless, this approach causes the use of many suboptimal controllers. A more realistic approach would instead frequently use the best controller so as to maintain an acceptable overall performance, adapting as needed to attempt improving it. Alternatively, it could modulate the variation of the best controller according to the performance, so as to explore aggressively with a poor behavior and gently with a better one. Another motivation for the drop is related to the number of epochs considered. Indeed, in a static scenario, we expect the average performance to increase over time as the controllers get refined, until stabilizing to a *plateau*. Considering a long experiment permits the average performance to approach the value of the *plateau*, decreasing the influence of initial epochs displaying poor behaviors. Differently, in a short experiment, the average performance during the run can be misleading. For this motivation, we propose to consider also the average performance *throughout* the run. This provides additional information, such as the time to reach the *plateau*, its value, and the potential decrease in performance in case of catastrophic forgetting. With this measure, we observe that the drop in the average performance is due to the low number of epochs considered and not necessarily to a weakness of the adaptive mechanism. Indeed, the trend of the local adaptation keeps increasing in almost every task, while that of the random adaptation remains stable. We conclude that the average performance of a period of time is a powerful tool to rapidly understand the quality of an adaptive mechanism, but it requires attention to avoid misinterpretation. Instead, the average performance over time conveys more information, resulting more difficult to misunderstand but more demanding to interpret.

The final measure we propose considers achievements during the run. This means counting the number of times the robot completed a desired action or achieved a desired result. From the point of view of mere effectiveness, this measure is the most informative. Indeed, while the performance can sometimes be hardly interpretable, counting goals is more straightforward. Nevertheless, this measure is not applicable in every situation. An example is that of collision avoidance, whether the definition of success is blurred: is turning on itself considered a success? Many questions such as this complicate defining achievements. Therefore, we apply this

measure only to the T-maze and foraging tasks. This shows that the local adaptation performs better than the random one, supporting the previous considerations on its effectiveness.

From the previous considerations, we can outline the utility of the measures we propose while considering online adaptation. We find it useful to consider the *maximum* performance attained, but only with the aim to estimate if the mechanism is effective in producing working solutions. Indeed, this metric does not permit understanding if the adaptive mechanism sustains the performance over time, nor if the score depends on a single fortuitous situation. For instance, the robot might be in a favorable position to earn a high evaluation without having really performed the desired way. Imagine a robot navigating the area avoidance arena (see Figure 2.3b). The robot could just find itself in a position suitable to go straight without ever encountering a forbidden area. This would give it a perfect score without having really learned how to behave from the perspective of the designer. Instead of the maximum performance, we consider more important the *average* performance *of* and *throughout* life. The former indicates how effective our robot is in the environment, while the latter informs about the stability of the behavior and the time required to find a good solution. Indeed, while the average performance *of* life depends on the evaluation time[6], the performance *throughout* life tells how the adaptation is going. Seeing a *plateau* in the performance trend would, for instance, indicate a stagnation in the improvement, and thus suggest what the long-term performance of the robot could be. Finally, we also analyze the performance of the controllers in terms of *achievements*, that indicate the completion of desired goals. This approach is more straightforward and could be used even during runtime operations to drive the adaptation, similarly to what happens in reinforcement learning. Nevertheless, evaluating the behavior in terms of achievements is not always possible. Additionally, this evaluation is less general, depending on a very specific definition of "goal". These limits affect its applicability both to drive the adaptation and to analyze the results. Still, we believe it could be useful in some specific cases.

Towards the end of this chapter, we want to discuss some technical details. In these experiments, we used a threshold to control when to adapt locally and when to explore the solution space extensively. Even though the idea seems to be working, it requires a non-negligible num-

[6]We expect the performance of the robot to improve over time, and thus its average performance to increase with the number of epochs.

ber of trials to find a good threshold value. Additionally, this strongly depends on the task, on the environment, and on the characteristics of the evaluation function. Since its only goal is to speed up the adaptation, we believe its use is not justified. Rather, we suggest allocating more computational resources or performing the bootstrap phase offline. We face a similar problem while deciding the duration of an epoch (i.e., the evaluation time of a solution). Indeed, this strongly influences the computed performance and the exploration of the search space. Obviously, the longer the trial, the more accurate the evaluation, but also the time required to adapt. In a real-world scenario where time is a crucial factor, we need to balance the accuracy of the evaluation and the time it requires. Therefore, also the evaluation time is an important parameter requiring careful tuning, but, unlike the threshold, it cannot be avoided.

In this chapter, we performed some preliminary investigations to understand whether a simple online adaptation mechanism could be used to adapt a controller to perform a specific task in a static environment. Therefore, we mainly concentrated on the attainable results rather than proposing a system suitable for real-world applications. This permitted us to understand some important aspects and requirements of robotic adaptation. In the next chapter, we will try to improve our approach by addressing a more realistic *dynamic* scenario. Specifically, we will consider a system with the goal of improving and maintaining the overall performance rather than just searching for better behaviors constantly. This should also consider the possibility that the "best controller" loses effectiveness over time, and thus needs to be replaced. We will explore all of that in a scenario in which we expect online adaptation to have great potential: fault recovery.

2.5 Chapter highlights

- Online adaptation seems a valid approach for finding effective behaviors for many robotic tasks.
- Constantly exploring new solutions could not be suitable for real-world operations, that require maintaining an acceptable performance over time.
- Classical metrics for the evaluation of design mechanisms are not sufficient to assess adaptive mechanisms operating at runtime.

- The *maximum performance* is useful to evaluate the capability to explore the solution space.
- The *average performance* permits to simply consider the overall effectiveness of the robot, also during exploration of alternative behaviors.
- The *performance over time* provides clearer insights into the adaptation process, highlighting increase, stagnation, and decrease of the trend.
- Considering *achievements* can be effective to assess the quality of a behavior, but it is not always applicable.

Chapter bibliography

Baldini, P., M. Braccini, and A. Roli (2023a). "Online Adaptation of Robots Controlled by Nanowire Networks: A Preliminary Study." In: *Artificial Life and Evolutionary Computation. 16th Italian Workshop, WIVACE 2022, Gaeta, Italy, September 14–16, 2022, Revised Selected Papers*. Vol. 1780. Springer Nature Switzerland, pp. 171–182.

Baldini, P., A. Roli, and M. Braccini (2023b). "On the Performance of Online Adaptation of Robots Controlled by Nanowire Networks." In: *IEEE Access* 11, pp. 144408–144420.

Braccini, M. et al. (2022c). "On the Criticality of Adaptive Boolean Network Robots." In: *Entropy* 24.10, 1368:1–1368:21.

Braitenberg, V. (1986). Cambridge, Massachusetts – London, England: The MIT Press.

Caravelli, F. et al. (2023). "Mean Field Theory of Self-Organizing Memristive Connectomes." In: *Annalen der Physik* 535.8, p. 2300090.

Christensen, D.V. et al. (2022). "2022 roadmap on neuromorphic computing and engineering." In: *Neuromorphic Computing and Engineering* 2.2, p. 22501.

Demis, E.C. et al. (2016). "Nanoarchitectonic atomic switch networks for unconventional computing." In: *Japanese Journal of Applied Physics* 55.11, 1102B2.

Fu, K. et al. (2020). "Reservoir Computing with Neuromemristive Nanowire Networks." In: *2020 International Joint Conference on Neural Networks (IJCNN)*. Institute of Electrical and Electronics Engineers, pp. 1–8.

Hebb, D.O. (1949). *The organization of behavior: a neuropsychological theory*. New York, NY: John Wiley & Sons, Inc.

Jo, S.H. et al. (2010). "Nanoscale memristor device as synapse in neuromorphic systems." In: *Nano letters* 10.4, pp. 1297–1301.

Löwel, S. and W. Singer (1992). "Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity." In: *Science* 255.5041, pp. 209–212.

Michel, O. (1998). "Webots: Symbiosis Between Virtual and Real Mobile Robots." In: *Virtual Worlds. First International Conference, VW'98 Paris, France, July 1–3, 1998 Proceedings*. Vol. 1434. Springer Berlin Heidelberg, pp. 254–263.

— (2004). "Cyberbotics Ltd. Webots™: Professional Mobile Robot Simulation." In: *International Journal of Advanced Robotic Systems* 1.1, p. 5.

Milano, G., E. Miranda, and C. Ricciardi (2022). "Connectome of memristive nanowire networks through graph theory." In: *Neural Networks* 150, pp. 137–148.

Milano, G. and C. Ricciardi (2023). "Nanowire memristor as artificial synapse in random networks." In: *Intelligent Nanotechnology*. Elsevier, pp. 219–246.

Milano, G. et al. (2019). "Recent Developments and Perspectives for Memristive Devices Based on Metal Oxide Nanowires." In: *Advanced Electronic Materials* 5.9, p. 1800909.

Milano, G. et al. (2020). "Brain-Inspired Structural Plasticity through Reweighting and Rewiring in Multi-Terminal Self-Organizing Memristive Nanowire Networks." In: *Advanced Intelligent Systems* 2.8, p. 2080071.

Miller, J., S.L. Harding, and G. Tufte (2014). "Evolution-in-materio: evolving computation in materials." In: *Evolutionary Intelligence* 7.1, pp. 49–67.

Miller, J.F. and K. Downing (2002). "Evolution in materio: looking beyond the silicon box." In: *Proceedings 2002 NASA/DoD Conference on Evolvable Hardware*. Institute of Electrical and Electronics Engineers, pp. 167–176.

Mondada, F. et al. (2009). "The e-puck, a Robot Designed for Education in Engineering." In: *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*. Vol. 1. 1. IPCB: Instituto Politécnico de Castelo Branco, pp. 59–65.

Ricciardi, C. and G. Milano (2022). "*In Materia* Should Be Used Instead of *In Materio*." In: *Frontiers in Nanotechnology* 4.

3 Internal changes

Online adaptation is a technique used to modify the behavior at runtime. Technically, it could be used to develop a suitable behavior for a task from scratch, as we did in Chapter 2. However, its strength relies on its ability to adapt an existing behavior to a new situation. One of the circumstances in which online adaptation could prove useful is facing individual changes. In this chapter, we explore the use of online adaptation to address exactly one of these triggers: faults. We initially investigate the level of performance that the robot can attain while damaged. Then, we explore the performance attainable while adapting to overcome runtime faults (i.e., how effective is the induced fault recovery).

This chapter builds on the results obtained in Braccini et al. (2024a) and Baldini et al. (2025a). We begin by describing the system used to control the robots, the adaptation mechanism, and the tasks. Then, we present and discuss the results, collecting some considerations for subsequent explorations on online adaptation.

3.1 Controller and adaptive mechanism

The controller used in these experiments leverages a Boolean network to perform computation. Boolean networks are an abstract model of gene regulatory networks (Kauffman, 1969) that captures significant biological phenomena such as cell differentiation (Braccini et al., 2019; Braccini et al., 2021; Huang et al., 2005; Huang et al., 2009; Montagna et al., 2021; Serra et al., 2007;

Villani et al., 2011). Many works investigated their computational capabilities and dynamical properties, suggesting that they could be effectively used as a computational substrate. One interesting application is in the robotics domain, where they have been used to develop adapting and evolving[1] robot controllers (Braccini et al., 2022c; Roli et al., 2011).

A Boolean network is technically represented as a discrete set of Boolean variables and functions. The state of a variable depends on the state of the neighbors and on the transition function that controls it. A specific class is that of random Boolean networks, that are created randomly according to some rules, such as the (possibly average) number of neighbors K of each node. Even the transition functions are randomly generated, usually creating a mapping from the state of the neighbors to the resulting state of the variable. This can be done by creating a table enumerating the possible input states and by randomly setting the corresponding output. The setup can be influenced by a bias p representing the probability of an output being set to 1. Different combinations of values for p and K change the dynamics of the network towards different dynamical regimes (Luque et al., 1997). Specifically, we consider two main regimes: *ordered* and *chaotic*. In the ordered region, a usually short sequence of states periodically repeats, defining an attractor. When perturbed, the dynamics of the network deviates from the attractor, but often returns to it in a short time. Indeed, the ordered region is characterized by a short propagation of the signals and by the tendency of the system to return to a stable state. The chaotic regime really exists only for asynchronous Boolean networks, but we can often refer to pseudo-chaotic dynamics also when talking about synchronous Boolean networks. In the former, the state changes erratically, making it impossible to predict the system state some steps later. In the latter, the attractor is long enough to make it often incomputable. When subject to perturbations, the chaotic network enhances and favors their spread, often permanently changing the future evolution of the system over time. At the boundary between the two regimes lies the *critical* region, characterized by an interesting combination of the two. In this condition, the attractors are usually longer than those of the ordered regime and shorter than those of the chaotic regime. Additionally, the perturbations propagate more than in the ordered region, but fade away faster than in the chaotic one. Many works suggest that critical Boolean networks favor computation, and indeed have been successfully used to complete different tasks, such as

[1] This is mostly thanks to their simple encoding and mutation.

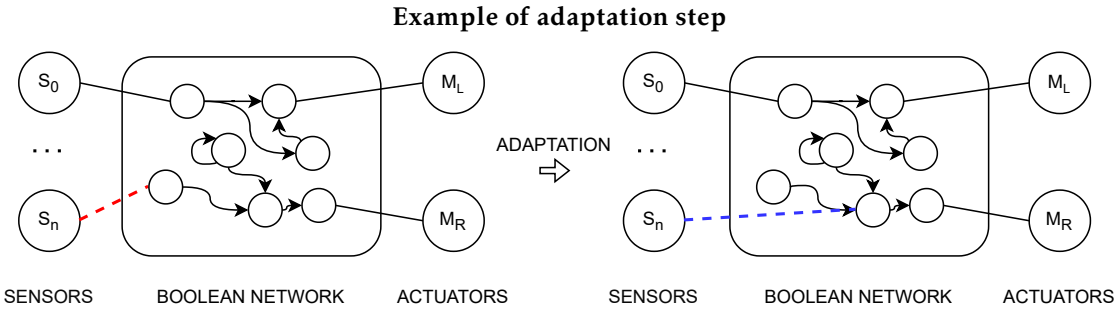


Figure 3.1: Example of an adaptation step. The coupling to reconnect is indicated by a red dashed line on the left. The reconnected coupling is represented by a blue dashed line on the right. As is visible, the adaptation affects only the node to which a sensor connects, and leaves unchanged the topology of the Boolean network.

classification, filtering, and control (Roli et al., 2018). Even Boolean networks evolved to solve specific tasks were often found to be critical (Benedettini et al., 2013).

The controller we propose works by perturbing the Boolean network state and by extracting outputs to control the motion of the robot. To do so, we connect the sensors and actuators to different nodes of the network. We refer to the sensor-to-node and node-to-actuator connections as *couplings*. The sensors perturb the Boolean network state with a value depending on the perception. Specifically, to match the Boolean network computing domain, we convert every sensory signal to a binary value. The output value to control the motors is instead multiplied by the selected motor speed. The initial controller consists of a set of couplings insisting on different Boolean network nodes. This avoids inputs from overwriting each other and outputs from receiving direct input cues or the same values. The robot runs the controller and collects a performance measure, representing how well it behaved during the evaluation epoch. Subsequently, the robot undergoes an adaptation process affecting the couplings of the best known controller so as to produce a variant to test (see Figure 3.1). It does so by selecting 1 to 6 input couplings[2] and reconnecting them to different points of the Boolean network. The idea is that it is possible to perturb the state of the network in order to generate a desired response, that is, a desired output which is used to properly control the actuators. Interestingly, the adaptive process we propose modifies the response of the robot without affecting the Boolean network structure (i.e., its topology and transition functions).

[2]The total number of input couplings is 24.

In this chapter, we present two variants of the adaptive mechanism, that we respectively call *a* and *b*. Variant *a* permits multiple couplings to connect to the same node. Additionally, it continuously tries new solutions so as to investigate a pure exploratory strategy. Variant *b* forces each sensor to connect to a "free" node, that is, a node not connected with any other sensor or actuator. Additionally, variant *b* alternates the *exploration* of a new solution with the *exploitation* of the best one. This has a double goal: (i) re-evaluating the best controller in possibly changed conditions, (ii) improving the performance during life. The re-evaluation simply averages between the old and the new performance attained by the best controller.

3.2 Experimental setting

We evaluate the ability of the adaptive system to overcome damage on two tasks taking place in different simulated environments:

- Phototaxis;
- Collision avoidance.

Specifically, we perform some preliminary experiments with the adaptive mechanism *a* on the phototaxis task alone. Then, we extend the investigation by verifying the effect of faults occurring halfway through the experiment with the adaptive mechanism *b*. We also explore this last scenario in the collision avoidance task.

The robot used in the experiments is a *foot-bot*, simulated in ARGoS3 (Bonani et al., 2010; Pinciroli et al., 2012). The foot-bot is equipped with 24 light sensors placed in a circular ring around the chassis of the robot, and can move by means of two wheeled motors. Before being passed to the Boolean network, the values of the sensors are encoded in binary form according to a threshold of 0.2. Values greater than the threshold become 1, while lower values become 0. The outputs are either 0 or 1, converted respectively into fixed linear velocity of 0 and 2 $\mathrm{cm/s}$ to control the motors. Modulating the speed is possible by controlling the frequency of 1s in the sequence of output values over time.

For each type and intensity of damage, we test multiple robot controllers that represent our replicas of the experiment. The controllers are all different inside the same experiment, but

task & adapt. mechanism	Boolean network size	epoch duration	epochs count	replicas count
phototaxis *a*	100	60 s	1200	300
phototaxis *b*	500	45 s	960	2500*
collision avoidance *b*	500	150 s	960	2500*

Table 3.1: Configuration parameters of each experiment. The second experiment (i.e., the one employing adaptive mechanism *b*) considers only replicas attaining a performance that indicates an effective behavior. Therefore, out of the 2500 executed replicas, only a subset is considered in the results.

are the same across different ones. For instance, replica 3 from the experiment with 6 missing inputs has the same starting controller as replica 3 from the experiment with 24 fixed inputs. This guarantees that the experiments are comparable. Each experiment consists of a number of adaptation epochs that depends on the task difficulty. Additionally, also the size of the Boolean network depends on the task complexity. The duration of each epoch depends instead on the task and on the size of the arena. Table 3.1 reports all the parameter configurations for each experiment.

In the first set of experiments involving robots starting already damaged, we consider all the replicas. In the second set of experiments involving robots that undergo faults halfway through the run, we consider only robots that learned the behavior without damage. Indeed, being able to complete the task is a necessary condition to talk about *fault recovery*, and since the controller of the robot uses a random Boolean network with fixed output couplings, it is possible that for some replicas a working controller does not exist at all. This might be due to topological pathologies or to the connection of the motors to fixed output nodes. The result is that for these pathological cases, the adaptive mechanism that we employ is hopeless. In order to avoid considering these ill situations, we set a threshold for each task, discriminating between robots that learned the behavior and robots that did not. We select the thresholds considering the characteristics of the evaluation functions and empirically observing multiple runs. Specifically, in the collision avoidance, a performance greater than 500 indicates robots capable of avoiding obstacles, even though not necessarily efficiently. In the phototaxis task, we select 0 as the threshold, identifying robots that, on average, move towards the light during the experiment. Higher performances just indicate more effective behaviors. We consider only replicas

whose average performance during exploitation epochs[3] in the first phase of the experiment (i.e., without damages) is higher than the threshold.

3.2.1 Phototaxis

In the phototaxis task, the robot moves as fast as possible towards a light source, and stays close to it once reached. The effectiveness of the behavior depends on the distance travelled towards the light since the start of the evaluation epoch. Specifically, for simplicity, we consider directly the distance from the light obtained by the simulator. This is a proxy to the equivalent local evaluation function employing the light sensors to understand the distance. Indeed, the robot can understand if it is moving in the correct direction and the distance from the light thanks to changes in its perception. An increase in the maximum perceived intensity indicates a movement towards the light, while a decrease indicates that it is going away from it. Similarly, the change in the radiation intensity between the start and the end of an epoch indicates the distance travelled towards the light. The mathematical formulation of the evaluation function for the phototaxis task is the following:

$$P^{Ph}_{epoch} = \delta_{start} - \delta_{end} \tag{3.1}$$

where:

- δ_{start} is the distance of the robot from the light at the start of the epoch;
- δ_{end} is the distance of the robot from the light at the end of the epoch.

The arena of the phototaxis task consists of a square arena. The robot starts in the bottom-left corner and moves towards the light. In the first set of experiments using the adaptive mechanism *a*, the light is at the center of an arena having an edge equal to $100\,\mathrm{m}$ (see Figure 3.2a). In the second set of experiments using the adaptive mechanism *b*, the light is in the top-right corner of an arena having an edge equal to $1000\,\mathrm{m}$ (see Figure 3.2b).

[3] An exploitation epoch uses the best behavior discovered by the robot until that moment.

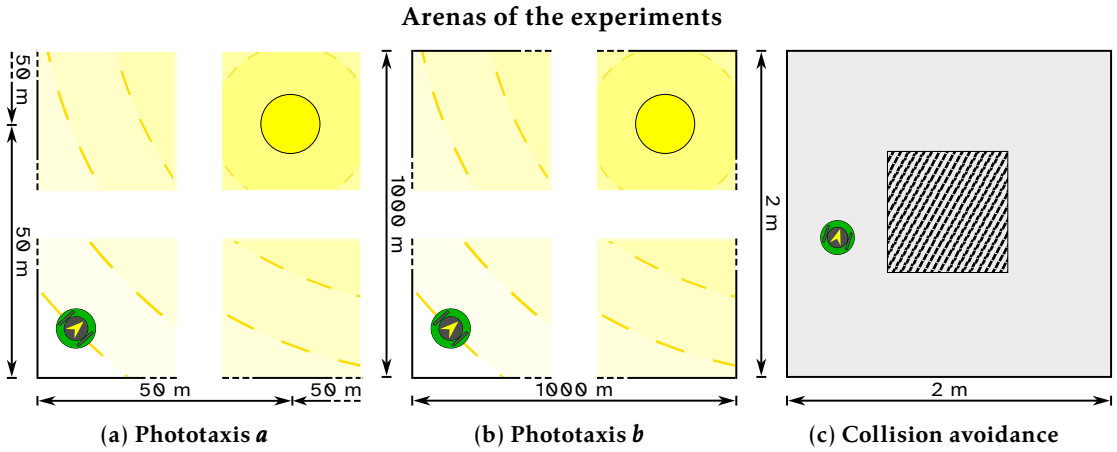


Figure 3.2: Representation of the arenas used in the experiment. (a) and (b) represent the arenas for the phototaxis task. The yellow circle represents the light source, and the concentric lines starting from it represent the gradient of luminous radiation. As the arena is too big to be fully represented, here we just show its main parts. (a) is the arena used in the task using the adaptive mechanism *a*; (b) is the arena used in the task using the adaptive mechanism *b*. The difference consists in the position of the light source and in the size of the arena. (c) represents the arena for the collision avoidance task. The block in the center represents an obstacle to be avoided. All arenas contain a green entity representing the robot.

3.2.2 Collision avoidance

In the collision avoidance task, the robot moves as fast and straight as possible while avoiding collisions. The evaluation function must therefore consider the maximum proximity of perceived obstacles, and the speed and direction of the movement. The function evaluating the performance of the robot performing collision avoidance is the following:

$$P^{C.A.}_{epoch} = \sum_{i=1}^{k} P^{C.A.}_{step,\,i} \tag{3.2}$$

$$P^{C.A.}_{step} = \left(1 - prox_{max}\right) \times \left(\frac{a_l + a_r}{2}\right) \times \left(1 - \sqrt{|a_l - a_r|}\right) \tag{3.3}$$

where:

- k is the number of steps in the epoch;
- i is the index of the step under consideration;
- $prox_{max}$ is the maximum proximity perceived by the robot, in $[0, 1]$;

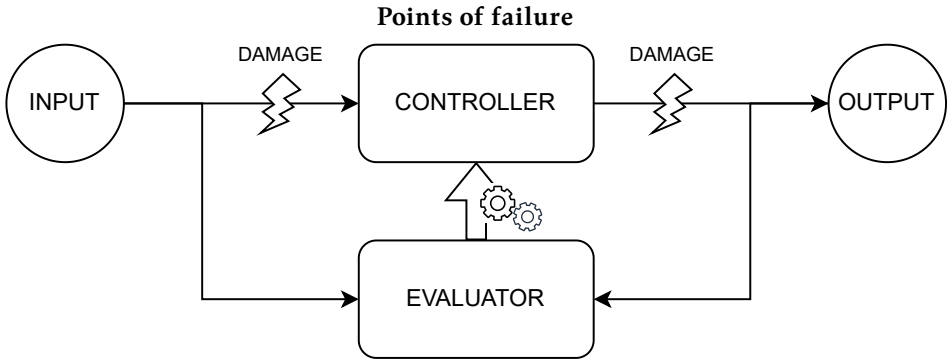


Figure 3.3: Representation of the points of occurrence of the damages considered in the experiments.

- a_l and a_r indicate whether the left and right motors of the robot are active, with 0 indicating a still motor and 1 indicating a turning one.

We experiment with the collision avoidance task in a square arena with a central block (see Figure 3.2c). The robot starts in a random position of the arena, and has to travel the circuit as fast as possible in a straight line. The best strategy consists in going straight through the side corridors and turning as fast as possible at the corners.

3.2.3 Damages

A real robot may undergo many types of faults. Here, we analyze how the performance changes in different situations. Specifically, we focus on damages happening on the couplings connecting sensors and actuators to the Boolean network (see Figure 3.3). The motivation is that, in order to operate, online adaptation needs to be able to evaluate the robot behavior. In our scenario, the evaluation function uses the readings of the sensor and the control values of the actuators to compute the performance value. If the damage occurs on the sensors themselves, then the evaluation function itself might have difficulties evaluating the behavior and thus guiding the adaptation to the desired outcome. Considering faults on the couplings permits to start exploring the potential of online adaptation for fault recovery in a clear and controlled scenario.

The first type of damage that we explore considers missing inputs to the Boolean network controller. This may be due to the breaking of the sensor or the disconnection of a cable. In this chapter, we consider the latter case. The result is that the sensor does not perturb the

internal state of the Boolean network anymore, making it blind to the corresponding cue. For this motivation, we refer to this type of damage as *missing* input. The second scenario considers the input of the Boolean network to block to a fixed random value. This simulates the case of a short circuit in which the output connection touches the source voltage or the ground. Unlike before, those still perturb the Boolean network internal state, but presumably in a detrimental way. We refer to this type of damage as *fixed* input. The third case is that of inputs consisting of random values, simulating the presence of noise masking the signal. Also in this case, those still perturb the Boolean network internal state. However, their signal can be considered noise and is therefore expected to be detrimental. We call this type of damage *noisy* input. Finally, only in the second set of experiments (i.e., those using adaptive mechanism *b*), we investigate the effect of damages on the actuators. Specifically, we consider that a motor might slow down due to consumption or reduced grip, affecting the trajectory of the robot. We refer to this type of damage as *slowed* actuator.

In the set of experiments using adaptive mechanism *a*, the damages can affect randomly picked sensors or a contiguous set of sensors. The aim is to represent both random and correlated faults. For instance, a short circuit could affect a near set of couplings causing localized damage. Differently, a fault due to consumption or age could affect random sensors.

We tested the working conditions with 0, 3, 6, 9, . . ., 24 faulty sensors. In the set of experiments using adaptive mechanism *a*, those are chosen at random at the beginning of each replica. In the set of experiments using adaptive mechanism *b*, those happen halfway through the run. We collected results on multiple replicas for each type of damage (see Table 3.1).

In this investigation, we are also interested in the recovery time of the performance when adapting a previously working controller. To this goal, we decide to compare the results with those of a controller designed from scratch for the damaged robot. We do so in the second set of experiments using the adaptive mechanism *b*. We refer to experiments in which the robot starts without damage as *informed*, and to the others as *clueless*. This is to represent the ability of the robot when the damage occurs. Clueless robots start with the same Boolean network topology and in the same position as the informed ones when the damage occurred, but not with the same Boolean network state and couplings[4]. When discarding replicas that did not learn the behavior

[4]We do not save the state of the Boolean network but only its creation seed. Therefore, we can restore only the

in the first half of the informed experiment, we also discard the corresponding clueless replicas. This is to compare the adaptation on the same controllers in these two different scenarios. We give clueless robots the same time to adapt to the damage as informed ones. Specifically, informed robots have 480 epochs to learn the behavior and another 480 to adapt it to overcome the faults. Clueless robots have 480 epochs to develop a controller from scratch considering the damage.

3.3 Results

In this chapter, we investigate the results with different metrics. We begin by considering the average distance of the robots from the light at the end of the run. Then, we consider the run length distribution (RLD), indicating how the percentage of replicas attaining a minimum level of performance that we consider good changes throughout the run. We continue by considering the distance from the light in the epochs. We analyze these three metrics in the phototaxis task with the adaptive mechanism *a*. Following, we discuss the performance in life, the time required to recover the performance, and whether informed robots have any advantage in recovering the performance with respect to clueless ones (i.e., does starting from a previously working controller have any benefit on the recovery?). We analyze these three metrics in the phototaxis and collision avoidance tasks with the adaptive mechanism *b*.

3.3.1 Attainable performance

We expect online adaptation to induce a graceful degradation of performance with the increase in damage. Specifically, we can use the final distance to the light source to directly understand the impact of faults. This estimates the capability of the robot to overcome damages in the long run. Overall, the results support the hypothesis that the attainable performance decreases gracefully as the damages increase (see Figure 3.4). Apart from some fluctuations due to variance in the experiments, the higher the number of faulty sensors, the greater the final distance from the light source. Notably, the performance decreases sensibly when the number of faulty sensors is high, following a non-linear trend.

Boolean network but not its state.

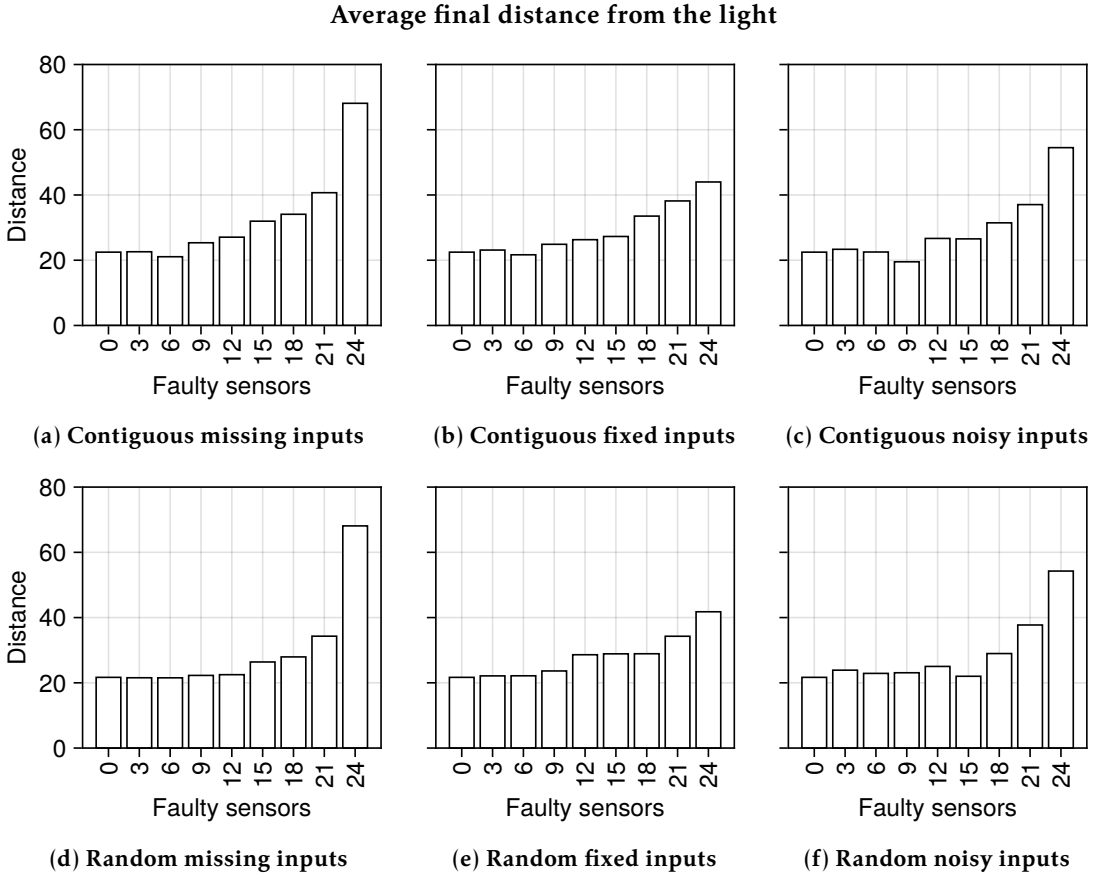


Figure 3.4: Final distance from the light, averaged across 300 replicas. Figure (a), (b), and (c) represent damages happening on contiguous sensors. Figure (d), (e), and (f) represent damages happening on random sensors.

In order to investigate the difficulty and time to reach a desired result, we compute the run length distribution. Given a target distance from the light, this tells us how many replicas got nearer since the start of the experiment. This provides an estimation of the efficiency of the adaptive process to exploit the (limited) resources the robots can use: the steeper and higher the curve, the more effective the adaptation. We compute the run length distribution with thresholds equal to 1 m and 5 cm. The trends are depicted in Figure 3.5 and Figure 3.6, respectively. Also in this case, we observe a gradual decrease in the ratio of successful robots as the number of defective sensors increases. Nevertheless, a striking difference emerges between the cases involving damages to random sensors and to contiguous ones: the run length distribution of the latter does not decrease until a large fraction of sensors is touched. In some cases, it seems even

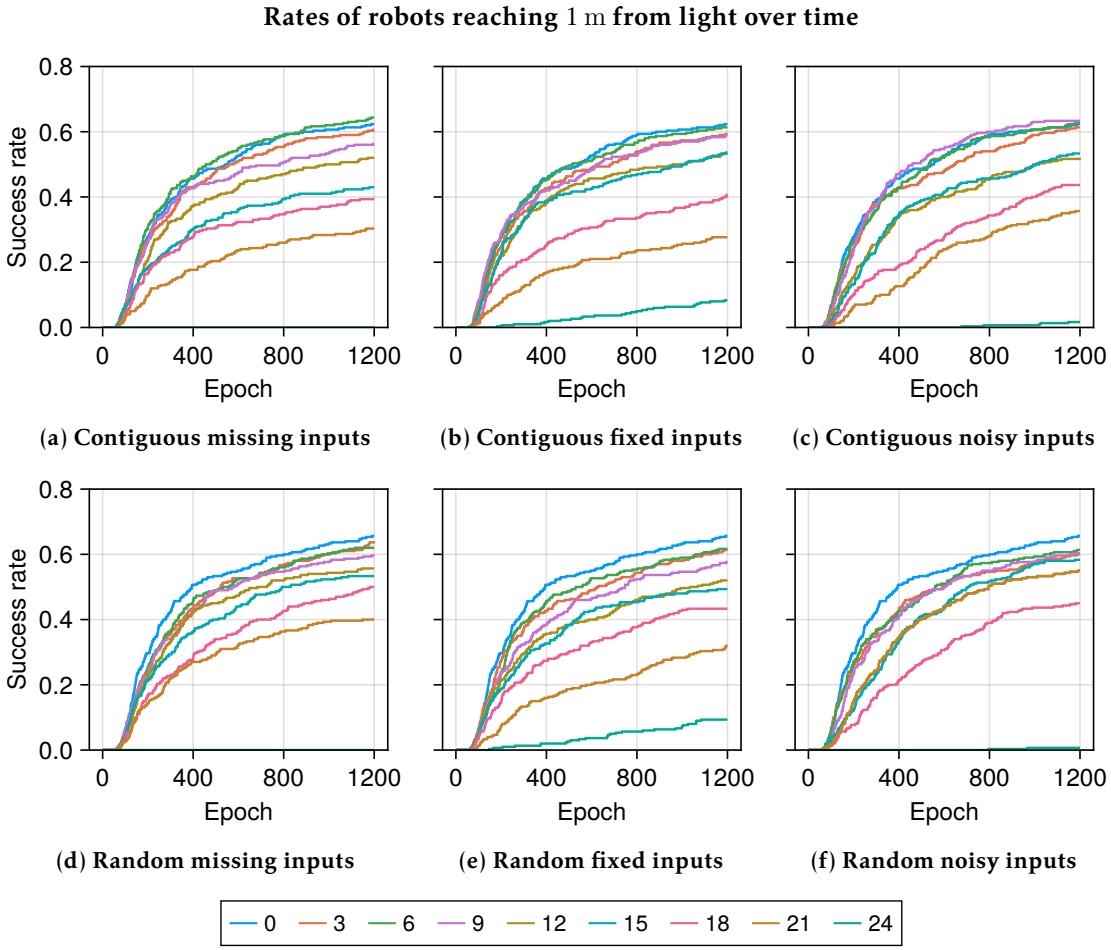


Figure 3.5: Run length distribution for target distance equal to 1 m. A point in the plot represents the fraction of replicas that achieved a distance less than or equal to 1 m in any epoch between the start of the experiment and the considered epoch. Figure (a), (b), and (c) represent damages happening on contiguous sensors. Figure (d), (e), and (f) represent damages happening on random sensors.

better to have a few damaged sensors. A possible explanation for this phenomenon depends on the way the controller perceives the environment. Due to the similarity of near sensorial perceptions and the binarization threshold, the Boolean network mainly discriminates between sensors facing or not facing the light source. Specifically, the robot can produce the best behavior by adjusting the direction when sensors that should be facing the light source no longer perceive it (or *vice versa*). This makes sensors about the sides of the robot the most important for navigation, as they permit detecting any deviation from the desired trajectory early on. Con-

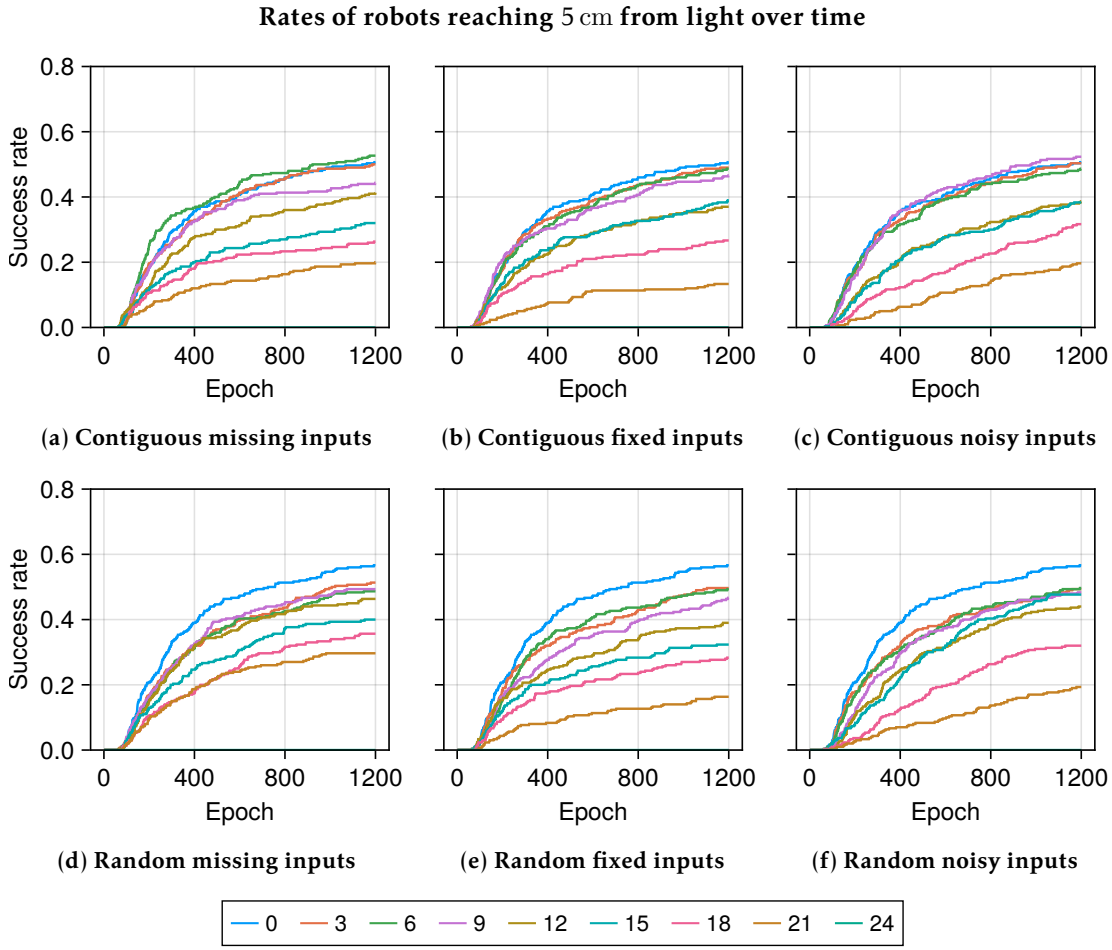


Figure 3.6: Run length distribution for target distance equal to 5 cm. A point in the plot represents the fraction of replicas that achieved a distance less than or equal to 5 cm in any epoch between the start of the experiment and the considered epoch. Figure (a), (b), and (c) represent damages happening on contiguous sensors. Figure (d), (e), and (f) represent damages happening on random sensors.

tiguous damages can affect both sides of the robot only when the number of faults increases notably. Differently, random faults can affect both sides of the robot even with a few damages, potentially reducing the effectiveness of the action more frequently. Therefore, it seems that having at least one important side of the robot still behaving correctly permits the generation of more effective behaviors. We thus conclude that, in our specific case, contiguous damages affect the performance less than random ones when considering a few faults (no more than 9), losing then their advantage as the damage increases.

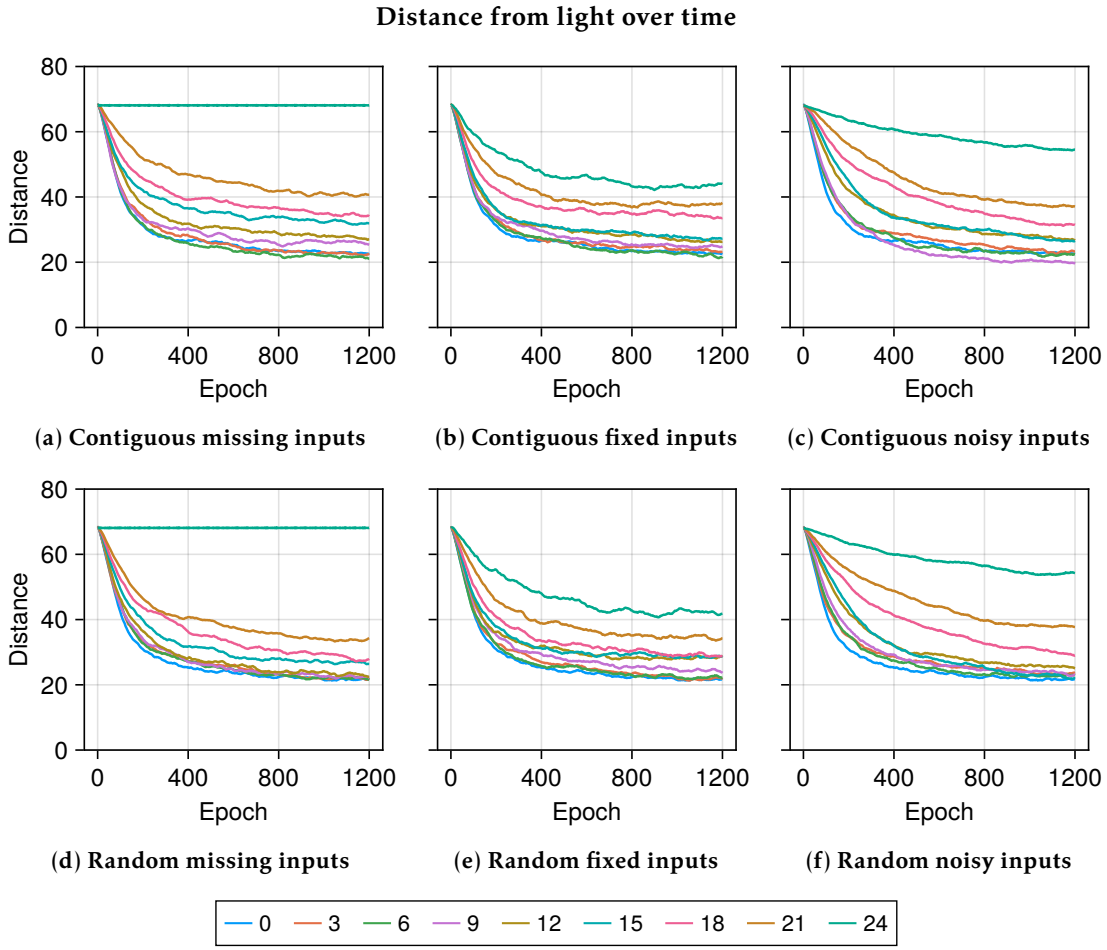


Figure 3.7: Distance from the light along adaptation epochs, averaged across 300 replicas. Figure (a), (b), and (c) represent damages happening on contiguous sensors. Figure (d), (e), and (f) represent damages happening on random sensors.

Finally, the distance from light at each epoch provides a complementary view of robot performance. This captures the performance *over time*, taking into account also unsuccessful adaptation attempts. Indeed, while the run length distribution considers only the best values attained until that moment, the distance over time permits identifying even non-monotonic trends. Figure 3.7 depicts the trend of the average distance from the light. We can observe a qualitative behavior analogous to the previous cases. It is remarkable that, also under this evaluation, the performance with a few damaged sensors is still equivalent to, or even better than, that in which all sensors are working. This observation reinforces our previous hypothesis, as it provides an

online view of the impact of having less information. We conclude by observing that with all 24 sensors damaged, the distance from the light decreases anyway. This is due to the fact that the faults we consider happen on the couplings between the sensors and the Boolean network, allowing the evaluation function to still evaluate the performance. The result is that the adaptation exploits even the erroneous perceptions, perturbing the Boolean network so as to control the motors appropriately. This also explains why robots with 24 missing inputs do not statistically move, as their absence cannot perturb the network properly.

3.3.2 Performance recovery

After investigating the attainable performance, here we assess the capability of online adaptation to recover the behavior after a fault. In our analysis, we consider only exploitation epochs, that are those using the best solution, and we ignore the exploration ones. The motivation is that, in this analysis, we are not interested in the exploration of solutions, but in the improvement in different conditions. The results show that online adaptation is generally able to recover the robot performance after undergoing faults (see Figure 3.8). Nevertheless, we can distinguish the impact on the maximum recovered performance caused by different types of damage in each of the two tasks. In the collision avoidance, having a damaged actuator seems to improve the performance (see Figure 3.9). The difference in the two wheels speed tends indeed to curve the trajectory of the robot, inducing a turn. Since the arena is a circuit, a turning behavior effectively navigates the environment. Normally, this is discouraged by the evaluation function, that penalizes unnecessary turns. However, this considers only if the motors are active, and not their effective rotation speed. The result is that the evaluation function we employ is not able to discriminate a turn due to a slowed motor from a straight movement, and thus cannot penalize it. This allows the robot to attain a higher performance despite showing a suboptimal behavior. This hypothesis holds when considering two faulted actuators. In this case, the robot does not display the turning behavior anymore. Instead, all its movements slow down by half. This increases the time to perform a turn and move away whenever it encounters an obstacle, decreasing the performance with respect to the undamaged scenario. Additionally, online adaptation does not recover the performance in the two-damages scenario. Instead, this decreases

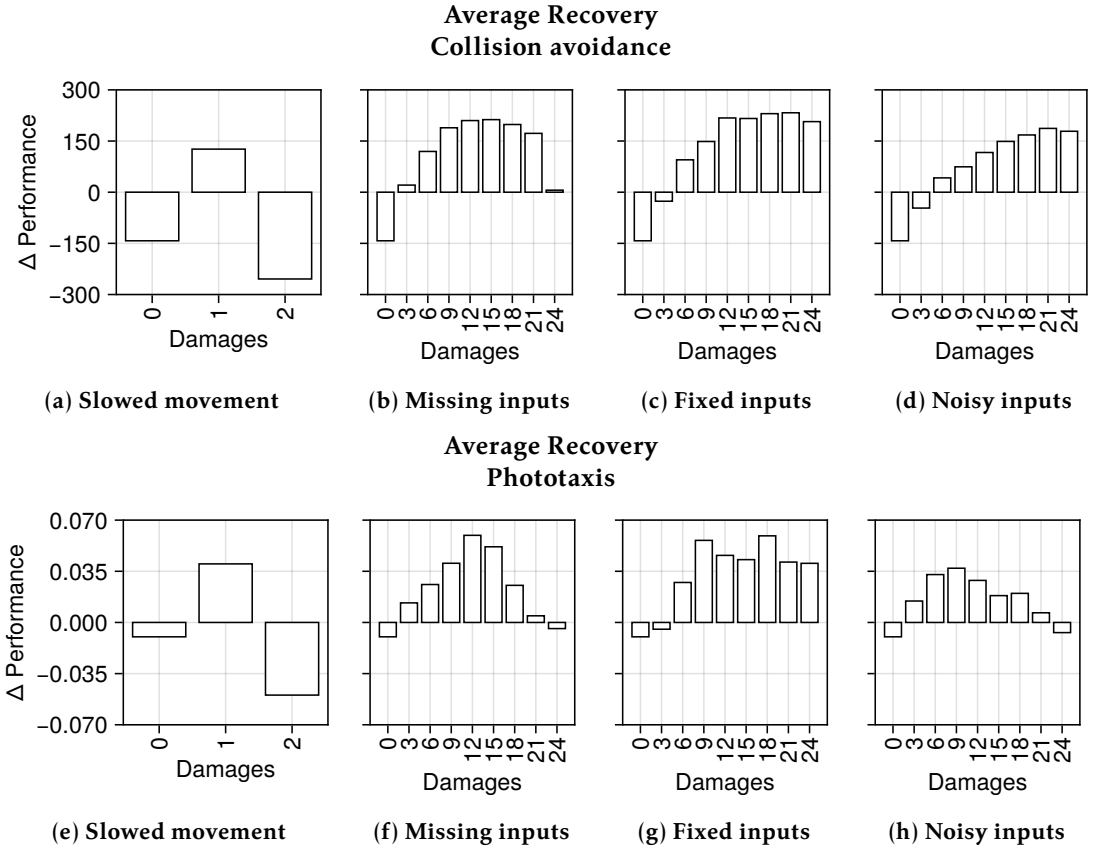


Figure 3.8: Average recovered performance, calculated as the difference between the average performance at the end of the experiment and that of the exploitation epoch immediately after the occurrence of damage. Figures (a), (b), (c), and (d) show the recovery in the collision avoidance task. Figures (e), (f), (g), and (h) shows the recovery in the phototaxis task.

slowly until it stabilizes.

Still in the collision avoidance, the occurrence of sensorial faults causes a sudden drop in the performance (see Figures 3.10 to 3.12). Nevertheless, online adaptation recovers the behavior in a short time regardless of the type of damage incurred. The differences consist in the intensity of the performance drop immediately after the damage and in the recovered performance. Specifically, a robot with missing inputs experiences a substantial decrease with the slowest recovery (see Figure 3.10). Fixed inputs cause a comparable drop of performance after damage, but recover faster (see Figure 3.11). Additionally, the performance attained after adaptation is slightly higher than that of robots with missing inputs, indicating a more manageable fault. Finally, noisy inputs cause a reduced drop in performance with respect to the two other types of

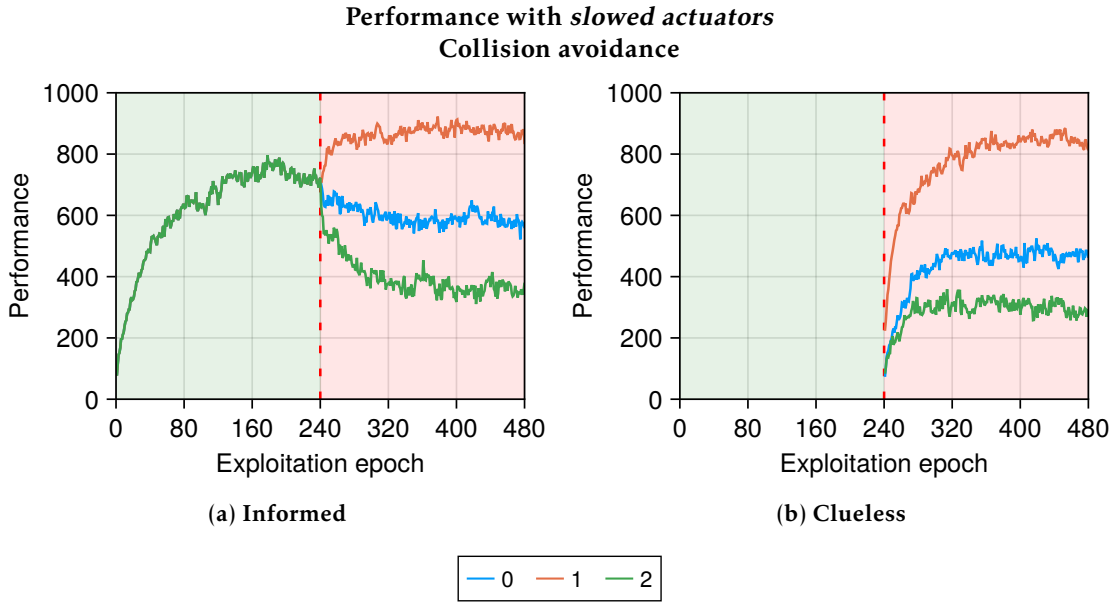


Figure 3.9: Performance according to the intensity of the damage to the actuators in the collision avoidance task. The trend is the average of the replicas that learned the behavior. Overall, a performance greater than 500 indicates a successful robot.

damage (see Figure 3.12). Additionally, the recovery is faster and leads to a performance similar to that of the undamaged robot.

In the phototaxis task, the peculiar trend displayed in presence of actuator damage did not replicate (see Figure 3.13). Indeed, the performance decreases gracefully according to the intensity of the fault. This is because, unlike the collision avoidance task, the phototaxis arena does not favor turning behaviors, but instead requires a straight trajectory towards the light to attain the highest evaluation. In the same way, a slower movement decreases the maximum travelled distance. This also limits the performance that the robot can attain, as it is now bound to half of the maximum without damage. The results show a relatively good recovery of performance in the case of just one damage, with a slight decrease in the case of two damages.

The performance trend in the phototaxis task with faulty sensors differs slightly from that of the collision avoidance (see Figures 3.14 to 3.16). When damages occur, we still observe a drop in performance. However, this recovers more slowly throughout the entire duration of the experiment, often without stabilizing. Interestingly, with some degree of damage, the recovery seems even enhanced. Indeed, slightly and heavily damaged robots recover less and

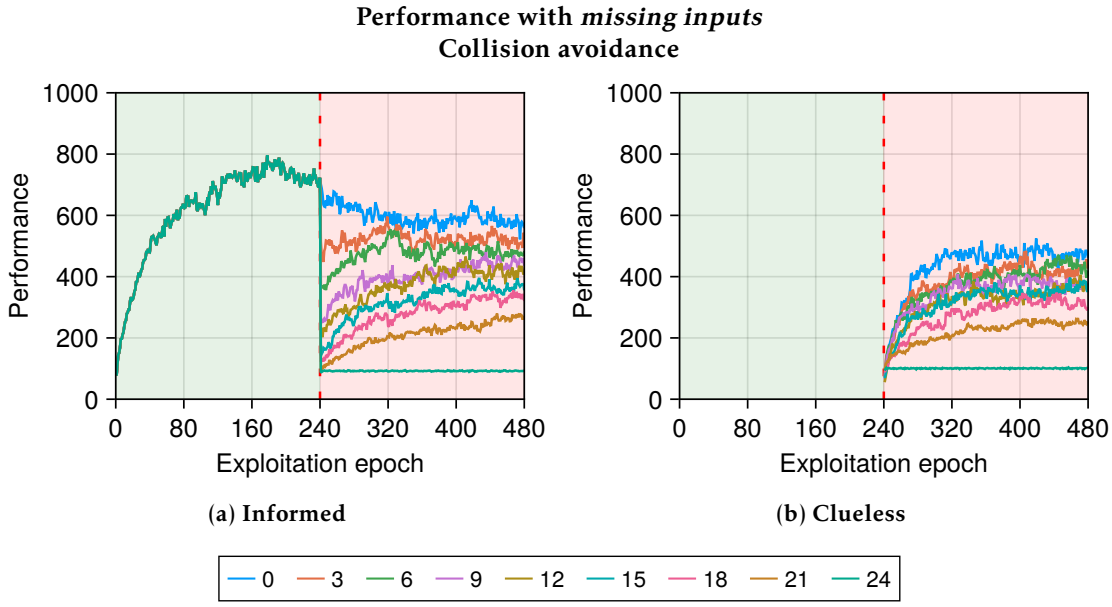


Figure 3.10: Performance according to the number of missing inputs in the collision avoidance task. The trend is the average of the replicas that learned the behavior. Overall, a performance greater than 500 indicates a successful robot.

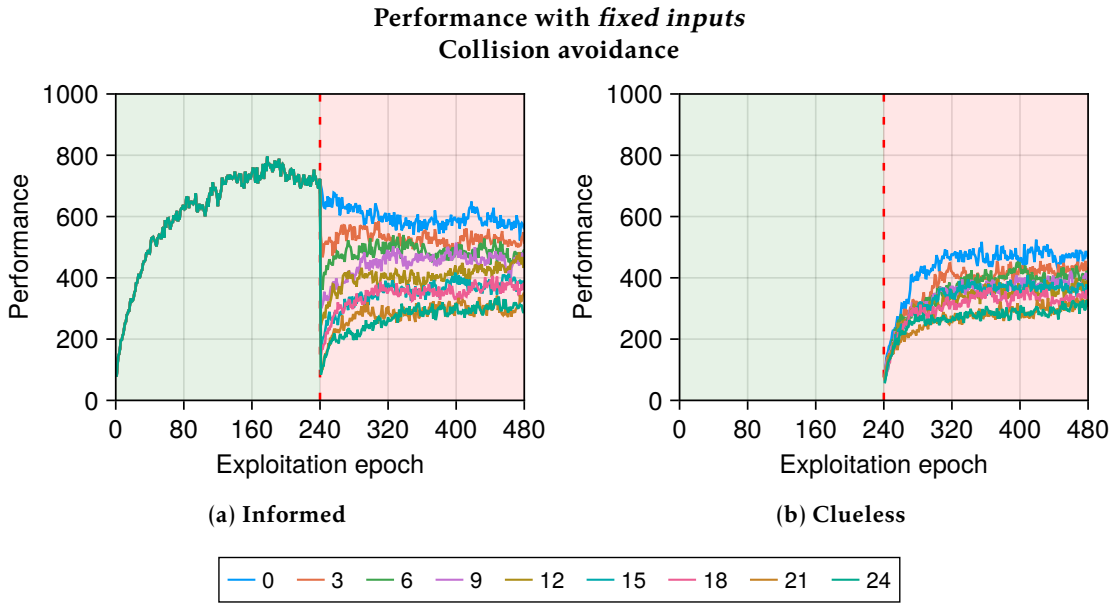


Figure 3.11: Performance according to the number of fixed inputs in the collision avoidance task. The trend is the average of the replicas that learned the behavior. Overall, a performance greater than 500 indicates a successful robot.

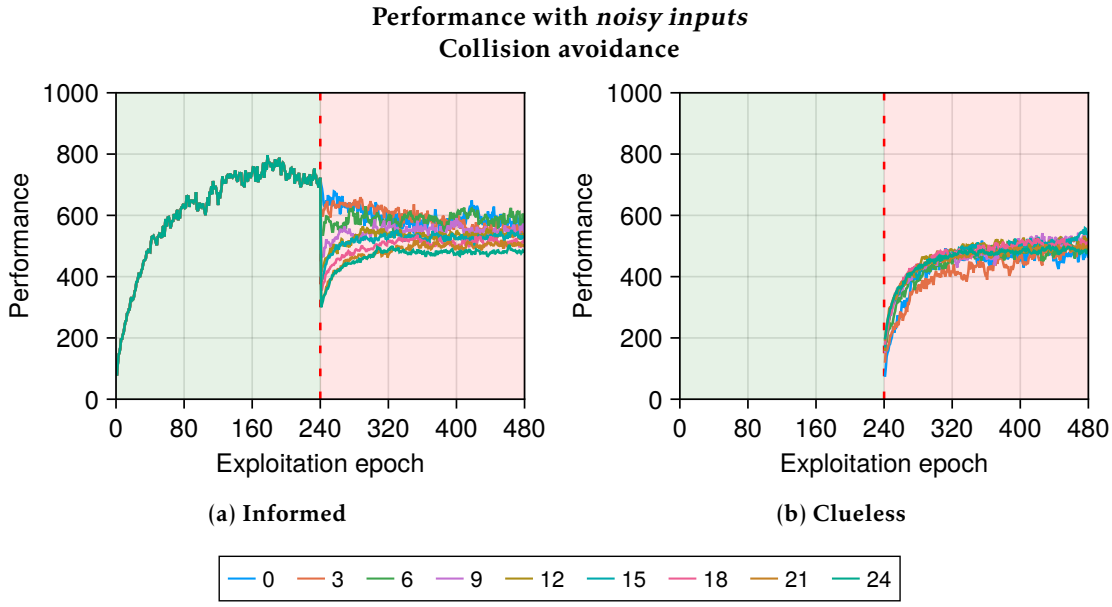


Figure 3.12: Performance according to the number of noisy inputs in the collision avoidance task. The trend is the average of the replicas that learned the behavior. Overall, a performance greater than 500 indicates a successful robot.

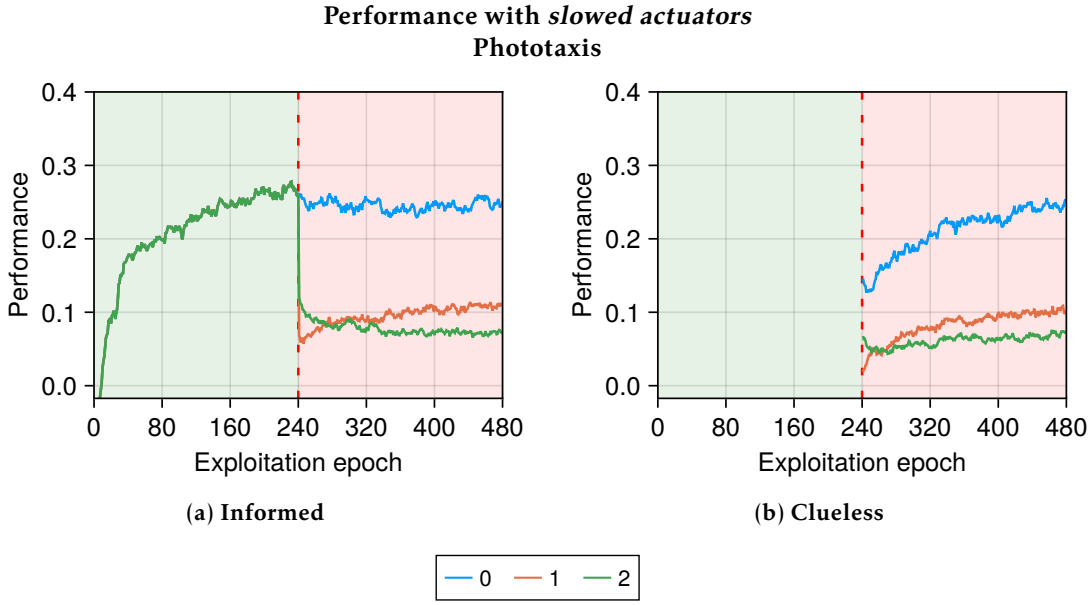


Figure 3.13: Performance according to the intensity of the damage on the actuators in the phototaxis task. The trend is the average of the replicas that learned the behavior. Overall, a performance greater than 0 indicates a successful robot.

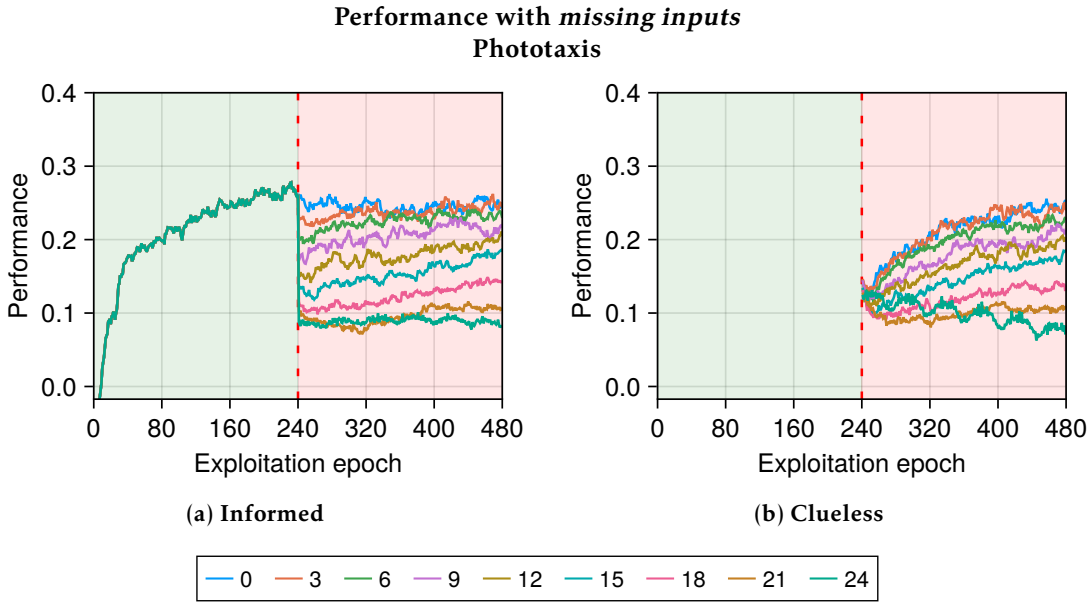
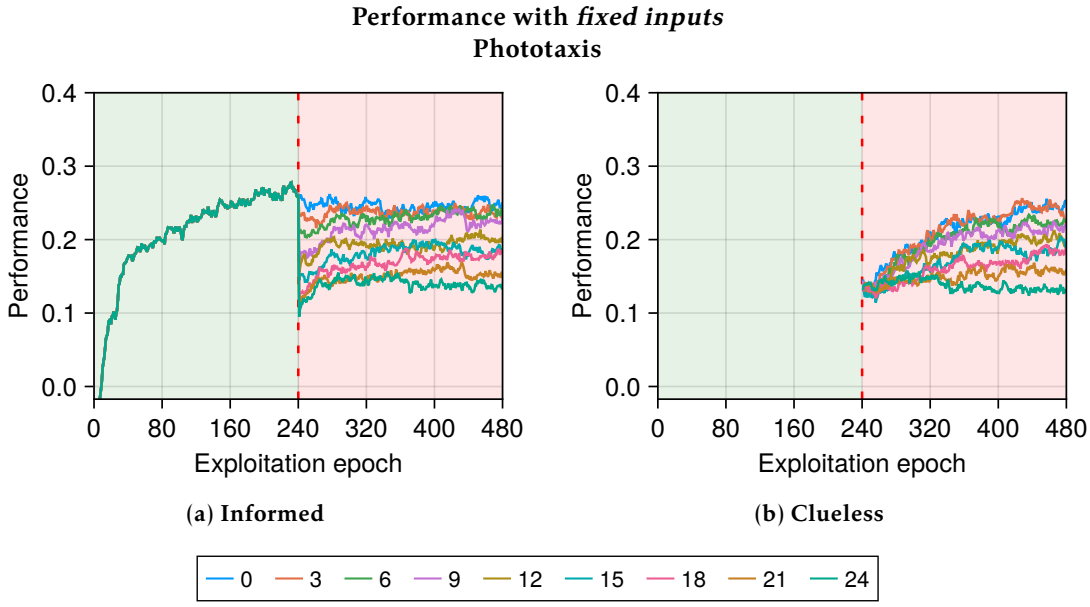


Figure 3.14: Performance according to the number of missing inputs in the phototaxis task. The trend is the average of the replicas that learned the behavior. Overall, a performance greater than 0 indicates a successful robot.

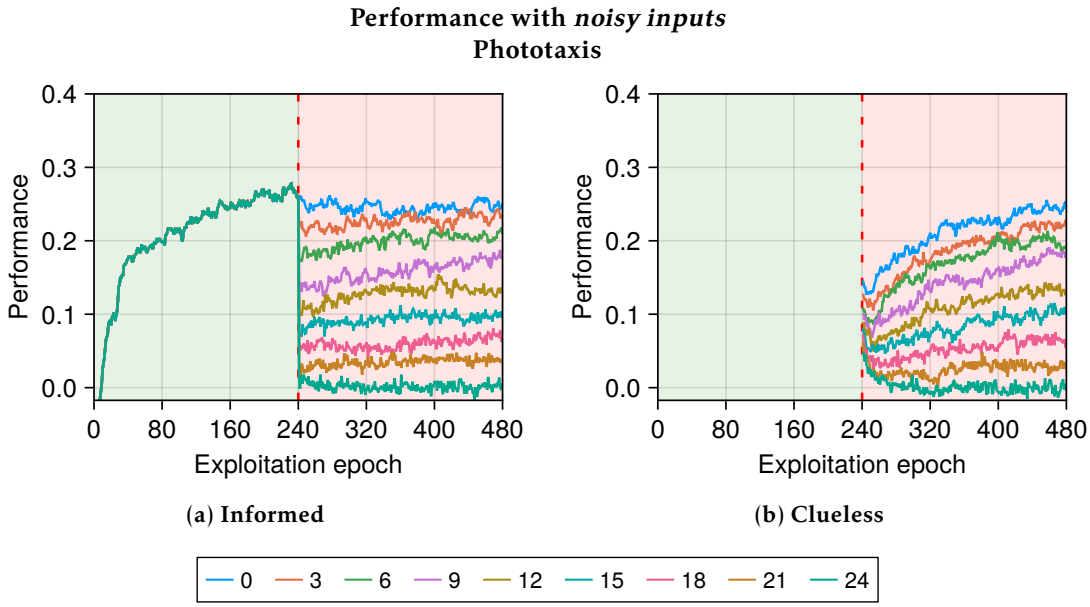
more slowly than moderately damaged ones. For instance, the recovery when subject to 12 or 15 missing inputs is much greater than that with 3 or 24 faulted sensors. This trend is robust across all the different experiments. The motivation is probably that few damages cause a negligible drop in performance and therefore a limited possible recovery. Differently, with many damages, recovering the performance becomes harder. A mean amount of damage causes a sensible drop, but still permits recovering the performance to a great extent.

Also in this case, we can distinguish the effect of different types of damage on the performance. With missing inputs, the performance drop is significant and does not recover notably for slightly or heavily damaged robots (see Figure 3.14). Differently, the recovery seems to be at its maximum with about half the sensors faulted. The experiments with fixed inputs show a similar trend, but even highly damaged robots are able to recover their performance (see Figure 3.15). Additionally, the recovered performance is greater than that of robots with missing inputs, indicating a more manageable fault. Finally, with noisy inputs, the drop in performance is the greatest and recovery is minimal (see Figure 3.16). Indeed, online adaptation seems to be able to recover the performance only for slightly faulted robots. This differs from the results



(a) Informed (b) Clueless

Figure 3.15: Performance according to the number of fixed inputs in the phototaxis task. The trend is the average of the replicas that learned the behavior. Overall, a performance greater than 0 indicates a successful robot.



(a) Informed (b) Clueless

Figure 3.16: Performance according to the number of noisy inputs in the phototaxis task. The trend is the average of the replicas that learned the behavior. Overall, a performance greater than 0 indicates a successful robot.

obtained in the collision avoidance task, in which noisy inputs are the type of fault affecting the behavior less.

In addition to the trends just analyzed, we also investigate the effects of adaptation in *clueless* robots, that start damaged and with a completely random input couplings (see Figures 3.9 to 3.16). This permits comparing the recovery time of *informed* robots that start with a previously working behavior, and to verify that this does not trap the controller into an ill solution. All experiments show that the performance *attained* starting from a working controller (i.e., informed) is similar to that attained starting from a clueless robot. This suggests that adapting a working controller does not prevent a successful exploration of alternative solutions. Instead, it *speeds up* their search, attaining better performance soon after the damage. This is especially true for limited faults that affect the performance less and thus require fewer modifications to become effective again. Also, this is especially true for the collision avoidance task, as in the phototaxis task the recovery speed-up is reduced.

3.4 Discussion

In this chapter, we investigate the performance of a robot subject to damage, and how it can recover from it. We do so by employing two different adaptive mechanisms: "*a*" always trying new solutions, and "*b*" periodically reusing its best one. Each permits focusing on a specific aspect of the investigation. The former considers only exploration, permitting investigating the effect of different types of damage with a purely exploratory adaptive mechanism. The latter permits identifying a decrease in performance, and thus adapting to it. Indeed, preliminary experiments highlighted that fixing the best evaluation as done by mechanism *a* does not work in changing conditions such as the ones considered in the experiments with adaptive mechanism *b*. A "best controller" at time t_1 might not be the best at time t_2, and if the maximum performance attainable in t_2 is lower than that attained in t_1, then the adaptation will keep searching for an impossible improvement. Re-evaluating the best controller makes it possible, therefore, to identify when it loses effectiveness, and thus allows replacing it with a better one. Calibrating the re-evaluation weight (i.e., how much our evaluation of the controller depends on its last

assessment) finally determines the repulsion to change of the robot[5].

The final performance and its trend show that online adaptation induces graceful degradation, meaning that the performance decreases as the number of faults increases. This suggests that the lost perception is indeed useful for the robot to behave properly. Nevertheless, this has some notable exceptions. In fact, the results also show that, in some conditions, the lack of a few sensors increases the performance of the robot, or the reduction is negligible. Our hypothesis is that removing some cues to the Boolean network simplifies the correct use of the remaining. In other words, removing some irrelevant signals may help the controller to focus on more relevant ones. Additionally, some specific types of damage, such as the fixed input, may be used to modify the dynamics of the network advantageously. These aspects address another crucial point in online adaptation: the contingent optimization of the resources available to the robot.

Online adaptation seems, therefore, effective in finding some working solutions in the presence of damage. Nevertheless, the main interest resides in using online adaptation to enable fault *recovery*. This means starting from a working solution and then adapting so as to overcome damages. Investigating this aspect raises some additional questions. Can online adaptation really recover the performance of a damaged robot? To what degree can the performance recover? Is it faster to develop a new solution from scratch or to adapt an existing one? Does starting from a previously working solution bias or limit the adaptation? We find that online adaptation is a valid strategy to overcome damages in most situations. It permits the recovery of the performance by an amount proportional to the intensity of the fault. Indeed, limited damages cause an almost negligible drop in performance, which is soon recovered to a slightly better value. Differently, more faults induce a greater drop in performance, which is usually followed by a recovery proportionally greater than that of fewer damages. In the phototaxis task, this trend is slightly different, with heavily damaged robots recovering their performance less. We also observe that different types of damage produce different effects in the recovery. Specifically, we notice that the fixed input is the type of damage that affects the robot less in the two tasks (i.e., the drop is limited and the recovery is fast). This is somewhat surprising, as we would expect missing inputs to do so. The motivation is probably that the former per-

[5]In our experiments, the re-evaluation weight is 0.5, meaning that we perform the average between the previous and new evaluation.

mits modifying the dynamics of the Boolean network in favor of the robot, eliciting the desired behavior in a specific environment. Not modifying the controller, the missing inputs lose the possibility to influence its dynamics, and therefore to modify the intrinsic behavior of the robot. Noisy inputs produce different results in the collision avoidance and phototaxis tasks. Specifically, in the former, they permit attaining the highest recovered performance, while in the latter, they produce the worst behavior. The motivation for this difference remains unclear. Finally, damages to the actuation are generally the hardest to recover, but can enhance the performance by exploiting peculiarities of the evaluation function. While this does not actually imply an effective improvement in the generated behavior, it is still an interesting point to consider while devising the function itself. Interestingly, this peculiar trend has been observed even in other works (Silva et al., 2017).

The aforementioned differences in the impact of different types of damage are not universal, but depend on the environment in which the robot is acting. Clearly, a robot with all faulty sensors should not be able to act properly. Nevertheless, it succeeds in doing so thanks to online adaptation, that exploits peculiarities of the environment to select a suitable behavior. Similarly, when a robot is just partially damaged, online adaptation permits using both sensory perception and peculiarities of the environment to improve the performance. This can be seen as a sort of overfitting of the behavior to a specific environment and to a specific type of damage. Usually, this would be considered negative, but due to the characteristics of online adaptation, this is not particularly problematic. Indeed, the ability to generalize is useful to face unexpected challenges, but it decreases in importance when we can directly adapt to them. Obviously, generalizing remains important, but it is no longer fundamental to succeed. All these considerations suggest that the difference in the performance with different types of damage could arise from their synergy with the environment and by how easily online adaptation exploits them. Notable example is that of one faulted actuator in collision avoidance, that curves the trajectory simplifying the navigation of the circuit arena.

A further interesting result is in the analysis of the time to recover the performance after the occurrence of damage. Indeed, trying to fix an already known controller might slow down the recovery, or even trap it in an unfavorable section of the search space. If that possibility actually existed, searching from scratch for a controller might attain better performance faster.

Instead, our results show that modifying a known solution to fix the behavior leads to a faster performance recovery with respect to learning from scratch. This even suggests that starting from known behaviors might speed up the discovery of controllers for different tasks.

Finally, one aspect that we believe is important to highlight is the peculiar condition that we investigate and its consequences. With the adaptive mechanism b, we consider faults occurring on the couplings connecting the sensors to the Boolean network nodes, and the Boolean network nodes to the actuators. Despite realistic, this type of damage is not the only one that might occur. For instance, the damage might affect the sensors and actuators themselves. In this case, the evaluation function that we employed would also be affected by the fault. Indeed, it also relies on the robot perception and control commands, meaning that it is also sensitive to faults. Since online adaptation relies on the evaluation function to work properly, this situation is potentially detrimental. We believe that in order to create more robust systems, there is a need to use more general evaluation functions relying on multiple sensors, on external evaluation, and / or on an abstract measure of "well-being" (e.g., maintaining essential variables in survival range as proposed by Ashby, 1962). We expect that these would drive the adaptation remaining more tolerant to faults. Additionally, the latter option could even enable the emergence of different behaviors in different environmental conditions.

In this chapter, we performed some investigations to understand whether online adaptation could be used to adapt a controller to overcome faults. We found positive results, suggesting that online adaptation is indeed suitable to adapt the behavior so as to tackle internal changes. Nevertheless, internal changes are not the only type of variation that the robot can experience. Of primary importance are also external changes, and the way online adaptation addresses them. In the next chapter, we will investigate the use of online adaptation in a dynamic environment where the conditions change unexpectedly. Specifically, we will propose an evaluation function based on a measure of "well-being", demonstrating that this leads to the emergence of two opposite types of behavior in different conditions.

3.5 Chapter highlights

- Online adaptation is a valid approach to overcome performance drops due to faults.

- Adapting to internal changes requires re-evaluating the best behavior of the robot, so as to forget it if it loses effectiveness.
- Periodically using the best behavior increases the overall performance of the robot, making it more valuable for real-world operations.
- It is faster to recover the performance of a controller than to find a new one from scratch.
- The characteristics of online adaptation potentially even permit exploiting faults to the robot advantage.
- If the faults affect the self-evaluation of the robot, the ability of online adaptation to recover the performance is hindered.

Chapter bibliography

Ashby, W.R. (1962). "Principles of the self-organizing system." In: *Principles of Self-Organization: Transactions of the University of Illinois Symposium*. Vol. 7. Pergamon Press, pp. 255–278.

Baldini, P., M. Braccini, and A. Roli (2025a). "Fault Recovery Through Online Adaptation of Boolean Network Robots." In: *Sensors* 25.18, p. 5849.

Benedettini, S. et al. (2013). "Dynamical regimes and learning properties of evolved Boolean networks." In: *Neurocomputing* 99, pp. 111–123.

Bonani, M. et al. (2010). "The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research." In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Institute of Electrical and Electronics Engineers, pp. 4187–4193.

Braccini, M., P. Baldini, and A. Roli (2024a). "An Investigation of Graceful Degradation in Boolean Network Robots Subject to Online Adaptation." In: *Artificial Life and Evolutionary Computation. 17th Italian Workshop, WIVACE 2023, Venice, Italy, September 6–8, 2023, Revised Selected Papers*. Vol. 1977. Springer Nature Switzerland, pp. 202–213.

Braccini, M. et al. (2019). "A simplified model of chromatin dynamics drives differentiation process in Boolean models of GRN." In: *2019 Conference on Artificial Life, ALIFE 2019*. The MIT Press, pp. 211–217.

Braccini, M. et al. (2021). "Dynamical properties and path dependence in a gene-network model of cell differentiation." In: *Soft Computing* 25.9, pp. 6775–6787.

Braccini, M. et al. (2022c). "On the Criticality of Adaptive Boolean Network Robots." In: *Entropy* 24.10, 1368:1–1368:21.

Huang, S., I. Ernberg, and S. Kauffman (2009). "Cancer attractors: A systems view of tumors from a gene network dynamics and developmental perspective." In: *Seminars in Cell & Developmental Biology* 20.7, pp. 869–876.

Huang, S. et al. (2005). "Cell Fates as High-Dimensional Attractor States of a Complex Gene Regulatory Network." In: *Physical Review Letters* 94.12, p. 128701.

Kauffman, S.A. (1969). "Metabolic stability and epigenesis in randomly constructed genetic nets." In: *Journal of theoretical biology* 22.3, pp. 437–467.

Luque, B. and R.V. Solé (1997). "Phase transitions in random networks: Simple analytic determination of critical points." In: *Phys. Rev. E* 55.1, pp. 257–260.

Montagna, S., M. Braccini, and A. Roli (2021). "The impact of self-loops on Boolean networks attractor landscape and implications for cell differentiation modelling." In: *IEEE/ACM transactions on computational biology and bioinformatics* 18.6, pp. 2702–2713.

Pinciroli, C. et al. (2012). "ARGoS: a Modular, Parallel, Multi-Engine Simulator for Multi-Robot Systems." In: *Swarm Intelligence* 6.4, pp. 271–295.

Roli, A. et al. (2011). "On the design of Boolean network robots." In: *Applications of Evolutionary Computation. EvoApplications 2011: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC, Torino, Italy, April 27-29, 2011, Proceedings, Part I*. Vol. 6624. Springer Berlin Heidelberg, pp. 43–52.

Roli, A. et al. (2018). "Dynamical Criticality: Overview and Open Questions." In: *Journal of Systems Science and Complexity* 31.3, pp. 647–663.

Serra, R. et al. (2007). "Why a simple model of genetic regulatory networks describes the distribution of avalanches in gene expression data." In: *Journal of Theoretical Biology* 246.3, pp. 449–460.

Silva, F., L. Correia, and A.L. Christensen (2017). "Evolutionary online behaviour learning and adaptation in real robots." In: *Royal Society Open Science* 4, p. 160938.

Villani, M., A. Barbieri, and R. Serra (2011). "A dynamical model of genetic networks for cell differentiation." In: *PLoS ONE* 6.3, e17703.

4 External changes

In the previous chapter, we explored the use of online adaptation to face unexpected changes happening inside the robot. Nevertheless, it is likely that most of the changes the robot will face are not internal, but rather external. The environment and, in general, all external conditions follow complex dynamics and are a source of uncertainty that we cannot fully prepare for. Therefore, in this chapter, we investigate the use of online adaptation to face changes in the environment. We do so by considering a simple task requiring the maintenance of a variable at a desired value, in a process akin to homeostasis. Additionally, we structure the arena so as to require the use of two different behaviors when the environment changes. This makes it possible to investigate whether online adaptation permits to find the best strategy for a specific environmental condition (or niche).

This chapter begins by describing the system used to control the robots, the adaptation mechanism, and the task. Then, we present and discuss the results, collecting some considerations for subsequent explorations on online adaptation.

4.1 Controller and adaptive mechanism

For this experiment, we propose to use a controller leveraging an Elman network, also called simple recurrent neural network, to control the robot (Elman, 1990). An Elman network is one of the simplest recurrent neural network architectures, consisting of a single hidden layer and

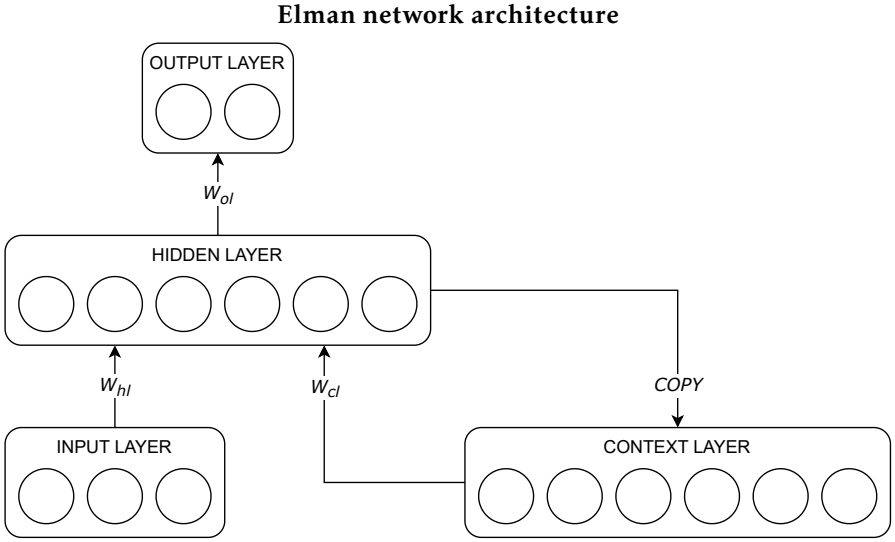


Figure 4.1: The Elman network architecture. Specifically, the version used in this investigation consists of 3 input nodes, 6 hidden nodes, and 2 output nodes. Each hidden node has a context unit copying its value. The hidden node computation considers, therefore, the inputs and the context unit containing its value at the previous iteration.

some context units acting as a memory (see Figure 4.1). Specifically, a context node is associated with a specific hidden node, of which it copies the value. During output generation, each hidden node receives as input the values from the input layer and from its context unit. The value of the hidden nodes is then used to compute the output of the network and to update the context layer. Mathematically, we can express the output of an Elman network as follows:

$$\begin{aligned} o_i &= \tanh\left(W_{ol} \cdot \mathrm{B}\left(h_i\right)\right) \\ h_i &= \tanh\left(W_{hl} \cdot \mathrm{B}\left(i_i\right) + w_{cl} \odot c_i\right) \\ c_i &= h_{i-1} \end{aligned} \tag{4.1}$$

where:

- o_i is the column vector of the output layer values at time i;
- h_i and h_{i-1} are the column vectors of the hidden layer values, respectively, at time i and $i-1$;
- i_i is the column vector of the input layer values at time i;
- c_i is the column vector of the context layer values at time i;

- W_{ol} is the weight matrix connecting the hidden layer to the output layer;
- W_{hl} is the weight matrix connecting the input layer to the hidden layer;
- w_{cl} is the weight column vector connecting the context layer to the hidden layer (each hidden node considers only its corresponding context node);
- B is a function that appends a bias to a given vector;
- tanh applies the hyperbolic tangent activation function to each element of a vector or matrix.

We generate the initial controller by randomly setting its weights to a value in $[-2,\ 2]$. Exceptions are the context- and output-layer weights, which are respectively set to one and zero. The former are due to the original architecture proposed by Elman, that directly copies the weight of the hidden node to its context unit. The latter are due to preliminary investigations revealing that the mechanism we propose adapts zeroed weights more easily. Finally, the initial state of the context units is set to zero. In this experiment, we create six hidden and context nodes. The input and output layers contain, instead, a node for each sensor and actuator, respectively.

After creation, the initial controller is evaluated in the task and dynamic environment that we present later. The robot records the performance, that represents its first "best" score. Then, the robot starts alternating between exploitation and exploration phases. During exploitation, the robot uses the best controller found so far, updating its evaluation by averaging the new and old performance. During exploration, the robot tests a new controller created by changing some weights of the best one. If the new controller is better than the previous one, then it becomes the new "best" and its performance is recorded. Specifically, the adaptive mechanism modifies a weight with a 20% probability, changing its value according to the performance attained by the best controller: the lower the performance, the greater the weight change. This enables the exploration of distant solutions of the search space when the performance is low, and near solutions when it is high. The update rule of a selected weight w is the following:

$$w \ = \ \min(2,\ \max(-2,\ w \ + \ e^{-5P_{best}} \ \cdot \ \text{rand}(-1.0,\ 1.0))) \tag{4.2}$$

where:

- w is the weight to update;

- P_{best} is the performance of the best controller.

This function ensures that the weight is always in the range $[-2, 2]$. Additionally, due to the multiplicative factor given by the exponential, it adds greater values with lower performance. For instance, with a performance of 1, the multiplicative factor is almost 0; with 0.5, it is about 0.08; and with 0, it is 1. As is visible, the multiplicative factor greatly modifies the intensity of the variation, affecting poorly behaving robots more. It is important to notice that the update procedure does not affect the hidden-to-context layer weights, that again just copy the value of the hidden nodes to their corresponding context unit.

4.2 Experimental setting

We assess the ability of the online adaptation mechanism to adapt to external changes in a simple task taking place in a dynamic environment. Specifically, we imagine a planet with significant thermal excursion between seasons and a robot having to remain in the warm area of the world to avoid freezing. For this motivation, we call this task "migration". We do not specify *how* the robot should achieve the goal (i.e., the behavior it displays), but only *what* it should achieve (i.e., remaining at the desired temperature). Despite some differences (e.g., self-regulation vs behavioral adaptation), the process of adapting in order to maintain an internal variable in a desirable range recalls the concept of homeostasis; therefore, we decide to refer to the evaluation function permitting this sort of response to changes as "homeostatic". To enable the operations, we permit the robot to perceive the surrounding temperature and the light in the environment. We use the light sensors to simulate a positioning system indicating the north of the planet[1]. This enables the robot to navigate the arena consciously, without relying on the gradient of temperature only, which in real life might change due to local conditions (e.g., altitude). The evaluation function used to assess the quality of a behavior is the following:

$$P_{epoch} = T_{end} \tag{4.3}$$

where:

[1] The light sources are all located at the top of the arena.

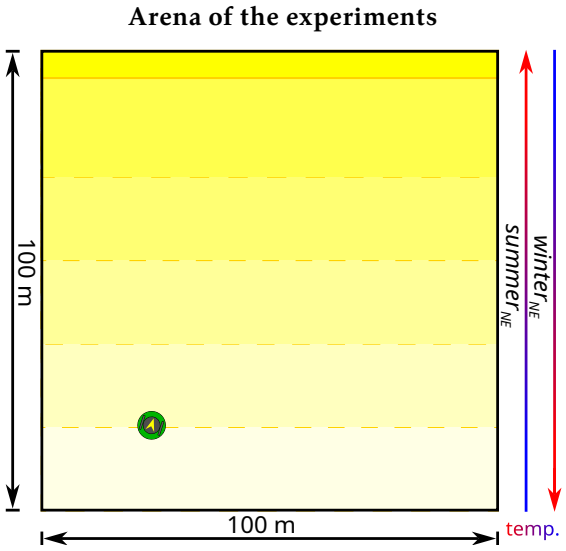


Figure 4.2: The arena of the migration task. The yellow bar at the top represents the light sources. The lighter yellow bands with dashed borders represent the luminous gradient, decreasing as the distance from the source increases. The green entity in the environment represents the robot, that is not in scale with the environment: it would be too small. Finally, the colored arrows on the right represent the temperature according to the season. Red indicates a high temperature, blue indicates a low temperature.

- T_{end} is the temperature at the last step of the epoch.

This function only considers the temperature reached by the robot at the end of the epoch. As this changes automatically due to the intrinsic dynamics of the environment, we can keep the evaluation function extremely simple. Although rather uncomplicated, this task is potentially relevant to enable more complex missions. For instance, we could require the robot to forage materials on an alien planet while seeking the best working conditions. Additionally, it highlights the optimal usage of an evaluation function, which should abstract away from the low-level characteristics of the task and/or environment, focusing only on the final goal. Unfortunately, this is often difficult to achieve, and it is currently possible only in very simple scenarios such as the one presented here.

The arena for the migration task is a square with edges equal to 100 m (see Figure 4.2). We refer to its top half as *Northern Hemisphere* (NE), and to the bottom half as *Southern Hemisphere* (SE). A series of multiple light sources indicates the "north of the world". We avoid using just one light source in order not to bias the robot necessarily towards a specific area of

the arena. For the remaining, the arena is completely empty, allowing the robot to move freely in it. Nevertheless, due to the duration of the experiment, the robot cannot move more than $2\,\mathrm{m}$ away or towards the light during an epoch.

The robot used in the experiments is a *foot-bot* (Bonani et al., 2010), simulated in ARGoS3, beta 59 (Pinciroli et al., 2012). We give the robot access to one virtual temperature sensor and two light sensors: the front and back ones. The robot moves by means of two wheeled motors that permit a maximum speed of $10\,\mathrm{cm/s}$. Modulating the speed is possible by controlling the output of the Elman network controller at each step. An entire experiment consists of 1000 replicas, each composed of 1440 epochs lasting $20\,\mathrm{s}$ each. At half experiment, the seasons in the environment switch, reversing the temperature gradient. We experiment starting from both the summer and winter seasons, so as to avoid biases. Following, Equation 4.4 reports the temperature value T at a virtual latitude y according to the season:

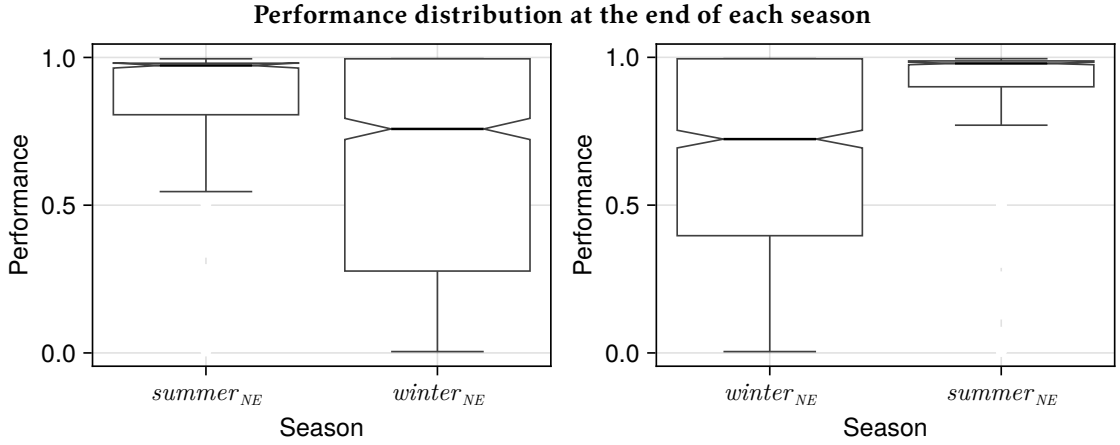
$$T_i = \begin{cases} (\ \ y + 50) \ / \ 100 & \textit{if } \text{season}(i) = \textit{summer}_{NE} \\ (-y + 50) \ / \ 100 & \textit{if } \text{season}(i) = \textit{winter}_{NE} \end{cases} \tag{4.4}$$

where:

- i is the index of the evaluation step of the experiment;
- $y \in [-50, 50]$ is a vertical position in the arena, representing a latitude;
- season is a function returning the season at step i;
- $\textit{summer}_{NE}$ indicates the summer season in the Northern Hemisphere;
- $\textit{winter}_{NE}$ indicates the winter season in the Northern Hemisphere.

4.3 Results

As this chapter investigates whether online adaptation is suitable to tackle external changes, we begin by analyzing the performance attained by the robot before and after the variation of the environmental conditions. This makes it possible to understand if the robot is able to adapt to the change, or if it instead remains anchored to the previous behavior that is now likely ineffective. To do so, we consider the distribution of the replica performances at the end of the

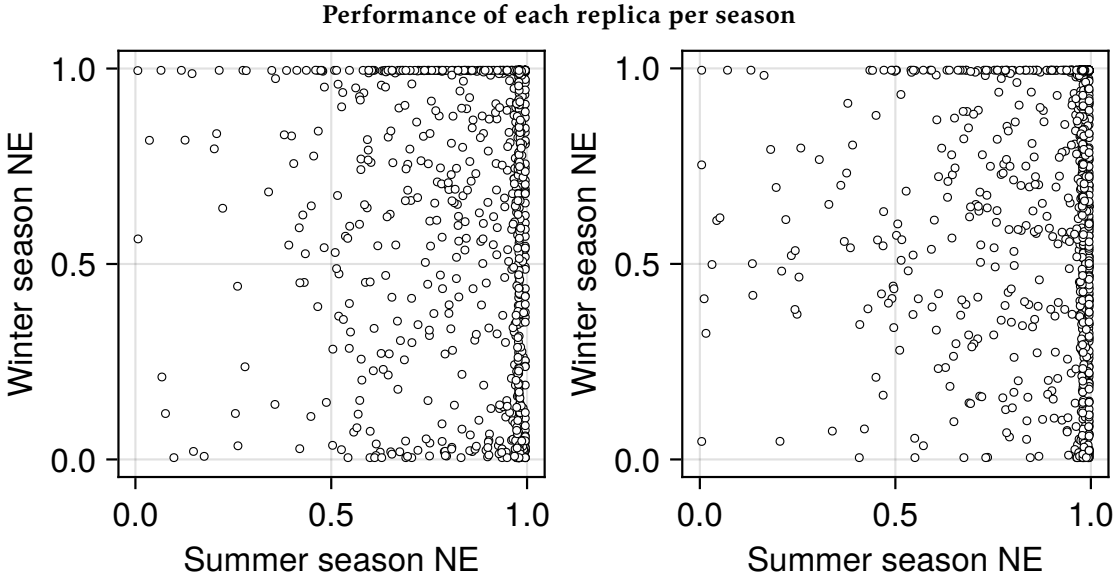


(a) *summer* $_{NE}$ $\rightarrow$ *winter* $_{NE}$ **(b)** *winter* $_{NE}$ $\rightarrow$ *summer* $_{NE}$

Figure 4.3: Performance distribution at the end of each season. (a) shows the experiment starting from the summer season in the Northern Hemisphere. (b) shows the experiment starting from the winter season in the Northern Hemisphere.

first and second seasons of the experiment (see Figure 4.3). The results show that the robot is generally able to reconfigure its behavior so as to attain good performances also in the changed environment. Nevertheless, it is also clear that the two distributions differ. Specifically, the behavior adapted to the *summer* $_{NE}$ phase attains a higher performance distribution than that adapted to the *winter* $_{NE}$ phase. This indicates that learning the phototaxis behavior (i.e., the best in the *summer* $_{NE}$ season) is easier than learning the anti-phototaxis behavior (i.e., the best in the *winter* $_{NE}$ season).

Comparing the performance distributions gives us a good idea of the effectiveness of the adaptive mechanism. Nevertheless, it is possible that some replicas behave well in one part of the experiment and not in the other. To consider this possibility, we create a scatter plot representing on one axis the final performance of a replica in the first season of the experiment, and on the other axis the final performance of the same replica in the second season of the experiment (see Figure 4.4). The results show that a reasonably large fraction of the robots is able to adapt their behavior to both seasons. However, this is slightly more marked in the *winter* $_{NE}$ to *summer* $_{NE}$ experiment, where the replicas distribute around the top-right part of the plot more. This suggests that passing from the anti-phototaxis behavior to the phototaxis one is easier than the other way around.

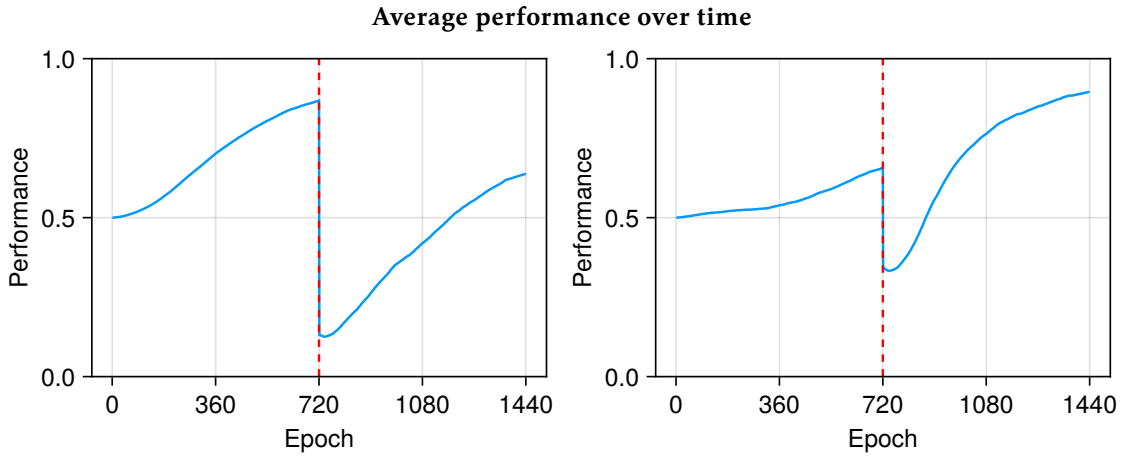


(a) $summer_{NE} \rightarrow winter_{NE}$ **(b)** $winter_{NE} \rightarrow summer_{NE}$

Figure 4.4: Performance at the end of each season per replica. (a) shows the experiment starting from the summer season in the Northern Hemisphere. (b) shows the experiment starting from the winter season in the Northern Hemisphere.

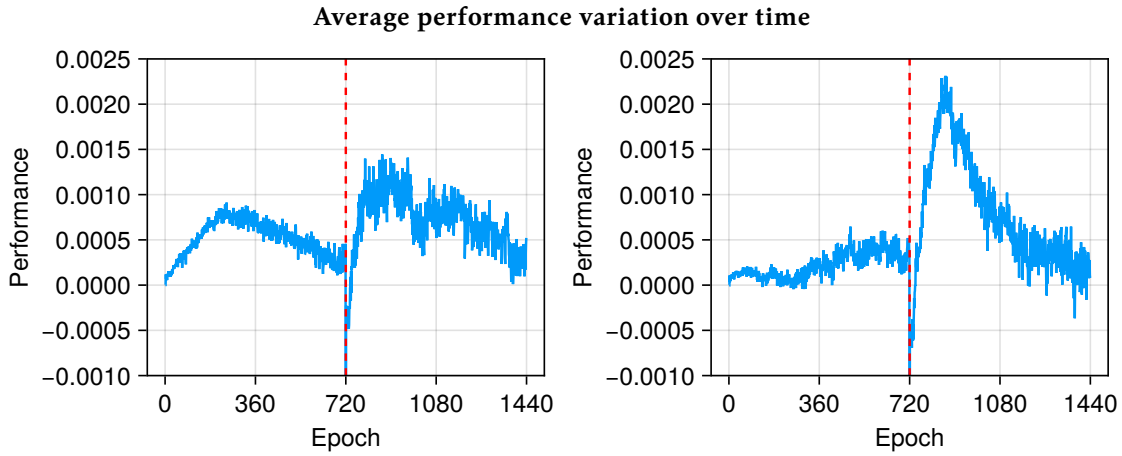
The previous analyses permit us to understand how effective the adaptation mechanism is in the case we consider. Nevertheless, it is also interesting to analyze the average performance over time. Indeed, this gives us additional information on the effectiveness of online adaptation. Specifically, it shows how fast the adaptation process operates, that is, how long it takes to modify the behavior so as to improve the performance. Starting from scratch might indeed change the adaptation time with respect to starting from an already known controller; indeed, this seems to be the case, as the performance increase in the former case is less than that in the latter (see Figure 4.5). This suggests that starting from an agnostic controller requires more work than changing an already adapted one, even if it displays an antipodal behavior.

Finally, since the evaluation function employed in this investigation produces a performance value strongly depending on past epochs, we find it reasonable to consider also the performance *variation* in a specific epoch. Basically, this considers if the robot moves towards a point of higher or lower performance during the epoch. We compute this value by simply subtracting the performance at the start of an epoch from the performance at the end. The results show that the performance increases rapidly at the beginning of a season, decreasing later as time



(a) *summer* $_{NE}$ → *winter* $_{NE}$ **(b)** *winter* $_{NE}$ → *summer* $_{NE}$

Figure 4.5: Average performance over time. (a) shows the experiment starting from the summer season in the Northern Hemisphere. (b) shows the experiment starting from the winter season in the Northern Hemisphere.



(a) *summer* $_{NE}$ → *winter* $_{NE}$ **(b)** *winter* $_{NE}$ → *summer* $_{NE}$

Figure 4.6: Average performance variation over time, calculated as the difference of performance between the end and the start of an epoch. (a) shows the experiment starting from the summer season in the Northern Hemisphere. (b) shows the experiment starting from the winter season in the Northern Hemisphere.

advances (see Figure 4.6). This trend is due to some robots reaching the end of the arena, thus *maintaining* the performance rather than increasing it. Interestingly, this highlights a disadvantage of evaluation functions considering a performance *variation* during the epoch: if, such as in this case, performance improvements are unattainable, the evaluation becomes unrepre-

sentative of the real capabilities of the robot. Consequently, due to a progressive decrease in performance over time, the learned behavior could be forgotten, even if it is still the best for the current circumstances. These types of situations are more likely to occur when using task-oriented evaluation functions, which typically consider the performance variation from the start of the epoch. Conversely, with an evaluation function such as the one proposed herein, the robot is more likely to retain the performance and the discovered behavior without forgetting it. The risk of this simplification, however, is to have less control over the resulting behavior, potentially inducing the emergence of unintended ones. Finally, it is interesting to note that the increase in performance in the plots is always positive, except for immediately after the seasonal change, when the robot still displays the behavior effective in the first season; however, even in this case, the performance recovers soon, indicating a rapid behavioral change.

4.4 Discussion

In this chapter, we consider adaptation to face changing environments. We do so by employing an adaptive mechanism alternating exploration and exploitation. Indeed, as in the case of internal changes investigated in Chapter 3, the robot needs to be able to recognize when an adopted behavior stops being effective. As before, this is done by averaging the old and new performance of the best behavior known. This allows the behavior to be replaced if its performance decreases too much, indicating that it does not perform well in all situations. Alternatively, this can happen if the behavior is no longer effective due to changes in external conditions.

In this chapter, we also propose to modulate the intensity of the adaptation according to the performance. We do so by reducing the intensity of the weights perturbation when the performance is high. Practically, this helps to modulate the variability of solutions that the adaptive mechanism can produce starting from the best one, *à la* Variable Neighborhood Search (P. Hansen et al., 2001). The good results we obtain seem to support the utility of the mechanism, but further investigations are needed to confirm it. Nevertheless, here we would like to highlight a limitation of modulating the perturbation that we noticed during its use. Specifically, we highlight that it requires knowing the boundaries of the performance[2]. Indeed, with an un-

[2] In this experiment, respectively set to 0 and 1.

bounded performance, we would not be able to tell how the robot is behaving, and thus we would not be able to modulate the adaptation. This limitation can be overcome by specifying the thresholds of bad and desired performances, but this is not always possible. A clear example presenting this issue is the foraging task presented in Chapter 2, in which the maximum amount of captured prey is practically unknown[3] and a desired amount does not really exist. Therefore, we claim that modulating the perturbation can be a powerful mechanism to adapt better, but its applicability depends on multiple factors such as the environment, the task, and the evaluation function.

Our results show that the proposed online adaptation mechanism is able to adapt the behavior of the robot to changing conditions. After a sudden drop due to the variation, the adaptation rediscovers a suitable behavior for the new situation. The procedure seems quite fast in the problem we explore, leading to a recovery in a few hundred epochs. At the same time, we observe a difference in the recovery speed and intensity, probably due to the complexity of the behavior required in each environmental condition. Furthermore, the results highlight that passing from one behavior to another is not equivalent to the opposite transition.

The investigation we performed in this chapter also highlights some interesting aspects of the evaluation function. Indeed, using a homeostatic evaluation function shows some advantages and disadvantages with respect to task-oriented ones. Specifically, a task-oriented alternative of this evaluation function could consider seeking or fleeing the light source in the two environmental conditions, as it is directly related to the temperature in the environment. This could push the robot to increase the delta of luminous intensity from the beginning to the end of the epoch. On the one hand, this approach pushes the robot to improve its motion capabilities. On the other hand, this causes a problem when the robot effectively reaches the light, where the delta cannot improve anymore. When we consider the homeostatic evaluation function we propose for this task, we notice that the issues are opposite. It is harder to push the robot to improve its performance, as it does not consider any improvement during the epoch, but it also leaves it freedom in choosing its behavior. This simplifies the management when the robot reaches the light. Moreover, this type of homeostatic evaluation function enables a less constrained adaptation, for instance, making it possible to employ peculiar strategies or components not intended

[3] Or at least, it is very hard to compute and depends on the environment.

for a specific goal, akin to biological degeneracy and affordances. We claim that both types of evaluation function are reasonable, and that what to use depends on the situation we expect to face. Nevertheless, we also believe that future explorations should consider a way to improve the evaluation of the robot so as to drive the adaptation more generally but effectively.

In this chapter, we consider adaptation to an environment that changes at a relatively low rate. Considering that we aim to simulate migration, and that this usually begins after many months from the arrival in a specific area, this claim is reasonable. In the same way, this is common in many other situations in which the agent update rate is higher than that of its surroundings. Nevertheless, the rates of change of the robot and environment are crucial for the adaptation to be effective. If the latter changes faster than the former, the adaptation will not be able to keep pace. The result is a suboptimal performance, possibly even lower than keeping a fixed behavior. Therefore, we can state that the adaptation time is a fundamental point to consider in order to understand if online adaptation is a viable solution in our expected situation. Additionally, finding always faster algorithms should be a central point of effort in research focusing on online adaptation.

In this chapter, we considered online adaptation to adapt to a slowly changing environment. Nevertheless, related to external changes is also co-adaptation. Indeed, other individuals can be considered a part of the environment changing at a very fast rate. The main peculiarity is that they could act in favor, against, or neutral with respect to our actions and goals. In the next chapter, we will investigate the requirements for co-adaptation to succeed, assuming that all the entities involved support each other.

4.5 Chapter highlights

- Online adaptation seems to be a valid approach to adapt the behavior to changing external situations.
- Adapting to external changes requires re-evaluating the best behavior of the robot, so as to forget it if it loses effectiveness.
- In order to be effective, a successful adaptation must be faster than the rate of external changes.

- Modulating the adaptation intensity according to the performance can better drive the search for good solutions.
- Modulating the adaptation intensity according to the performance is possible only if "bad" and "good" performance thresholds can be defined.
- Homeostatic and task-oriented evaluation functions can generate completely different robot behaviors.
- Task-oriented evaluation functions tend to induce the emergence of specific behaviors, potentially limiting the adaptation.
- Homeostatic evaluation functions leave the adaptation free to operate, permitting the generation of different behaviors in different contexts and therefore potentially increasing the overall flexibility and effectiveness of the system.
- Switching from one behavior *a* to another behavior *b* is not necessarily equivalent as switching from the behavior *b* to the behavior *a*: the difficulty might change.
- Switching from one behavior to another can be faster than finding the latter from scratch.

Chapter bibliography

Bonani, M. et al. (2010). "The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research." In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Institute of Electrical and Electronics Engineers, pp. 4187–4193.

Elman, J.L. (1990). "Finding Structure in Time." In: *Cognitive Science* 14.2, pp. 179–211.

Hansen, P. and N. Mladenović (2001). "Variable neighborhood search: Principles and applications." In: *European Journal of Operational Research* 130.3, pp. 449–467.

Pinciroli, C. et al. (2012). "ARGoS: a Modular, Parallel, Multi-Engine Simulator for Multi-Robot Systems." In: *Swarm Intelligence* 6.4, pp. 271–295.

5 Co-adaptation

Adapting to internal and external changes is a fundamental capability of humans and other living beings. Nevertheless, in our everyday life, we have to deal also with other agents. Adapting to others' behavior is technically similar to adapting to changing external conditions, but at the same time, it presents some differences. Specifically, others' behavior can be adversarial and detrimental, supportive and beneficial, or disinterested and unaffecting. Moreover, others' behavior can change as fast as ours, complicating our reaction.

In order to start exploring this context, in this chapter, we investigate co-adaptation. This is the problem of finding a behavior enhancing both the collective and our own performance. This is supposedly easier than adapting to adversarial robots and permits to understand the requirements to use online adaptation for decentralized multi-robot operations.

This chapter builds on the results obtained in Baldini et al. (2026b). We begin by describing the system used to control the robots, the adaptation mechanism, and the task. Then, we present and discuss the results.

5.1 Controller and adaptive mechanism

The control system of our robot consists of a simple artificial neural network composed of a single layer of 2 neurons (see Figure 5.1). The network takes in input the perception values of 8 sensors and outputs the control values of 2 motors. Each neuron averages the weighted inputs

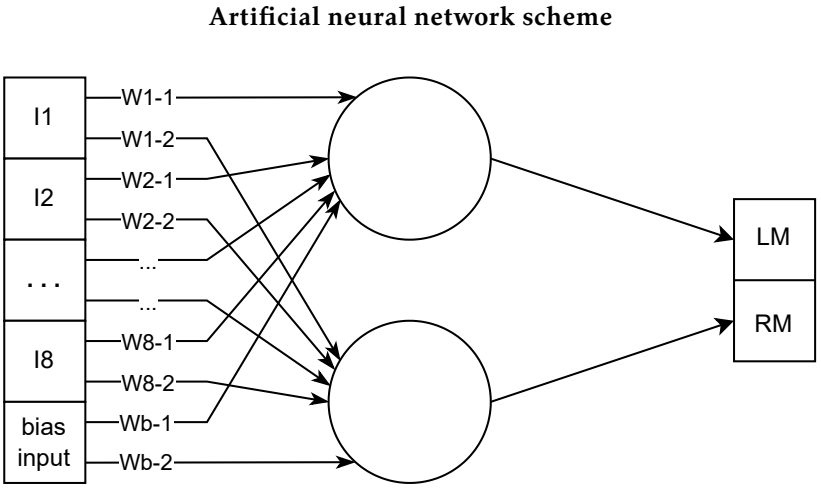


Figure 5.1: Schematic representation of the artificial neural network used as the controller of the robots. I1, I2, …, I8 represent the input values from the sensors (i.e., their perception). The bias input is fixed to 1. The adaptation mechanism modifies the weights $W1 - 1$, $W1 - 2$, $W2 - 1$, …$W8 - 2$, $Wb - 1$, $Wb - 2$ that multiply the correspondent input values. LM and RM represent, respectively, the left and right motor control values.

and bias value. The resulting value passes through a custom activation function F based on the sigmoid function S:

$$\begin{aligned} F(x) &= 1 - S(3x) \\ S(x) &= \frac{1}{1+e^{-x}} \end{aligned} \tag{5.1}$$

F mirrors the result of the sigmoid on the abscissa axis, and multiplies the input by an arbitrary value of 3 to increase the slope of the curve. We set the neural network initial weights by randomly sampling from a normal distribution centered at 0 and with a variance of 0.3. Therefore, each robot initial weights are different from those of the other robots in the group and replicas[1]. This permits to assert that the robots are heterogeneous since the beginning of the experiment.

The proposed adaptation mechanism is based on the idea of iteratively improving the best behavior known by the robot (Braccini et al., 2022a). This requires memorizing the best controller and its performance so as to be used as a starting point for the adaptation and to enable the robot to roll back in case the adapted configuration performs worse. If an adapted configuration performs better than the former best one, it becomes the new best. The search for

[1] We keep the same initial weights only across different experiment configurations. This is to ensure the comparability of the results with different parameters.

improved behaviors consists of stochastically modifying the weights of the best neural network controller and in testing them online (i.e., on the running robot; Baldini et al., 2023a). Modifying a weight consists in probabilistically adding a value sampled from a normal distribution with mean $\mu = 0$ and variance σ left as a parameter of the experiment. Specifically, we propose two different approaches: (i) using a fixed variance σ_F, (ii) using a dynamic variance σ_D depending on the performance. The former is the simplest approach; the latter builds on the idea that a controller accountable for a good behavior can be improved by minimally adjusting its weights, while a poor behavior requires a larger perturbation to improve. The result is that the dynamic variance causes the robot to adapt aggressively when its performance is low, and gently when its performance is high. Specifically, the dynamic variance σ_D depends on the performance at the previous evaluation according to the following equation:

$$\sigma_D = \sigma_{max} \cdot (1 - \sqrt{P_{epoch}}) \tag{5.2}$$

where σ_{max} is the maximum variance and $P_{epoch} \in [0, 1]$ is the performance value at the previous evaluation. We apply the root function to P_{epoch} in order to moderate the perturbation intensity for non-extreme values. For simplicity, from now on, when talking about the value of perturbation intensity, we will always refer to a generic parameter σ. This indicates σ_F in the case of the fixed approach and σ_{max} in the case of the dynamic approach, as they are the values directly in our control (i.e., that we can set as parameters of the experiment).

Adapting a single robot is different from adapting a swarm. A single robot can often assume that its actions have a direct impact on the evaluation that it obtains[2]. Differently, by concurrently adapting the robots of a swarm, we introduce *situationality*. The robot cannot distinguish anymore whether its evaluation depends on the effective quality of its behavior, or if it is instead affected by the interference of other robots. For instance, a robot with a behavior suitable for an aggregation task might still receive a poor evaluation if all its neighbors keep fleeing. Conversely, if the robot did not discover a good behavior but its neighbors block its movements, it will receive a high evaluation, perpetuating the poor behavior only thanks to the group. This issue is even worse in the case of very dynamic environments, where the external conditions do

[2]We assume the existence of a relatively static environment.

not depend only on other robots but also on the environment itself. To limit this problem, we propose two different countermeasures: (i) normally using the best controller, deciding whether to adapt stochastically, (ii) using a discount factor to forget situationally good behaviors.

The normal use of the best behavior known has a dual goal. First, it helps to sustain the life-long performance of the robot by avoiding constant adaptation. Continuously adapting would indeed prevent exploiting the found solution, focusing only on trying possibly ineffective variants of the controller. Second, a probabilistic and therefore asynchronous adaptation decreases the chance of detrimental interferences from other robots. Indeed, if most of the robots are using their best behavior, the adapting ones can tune their behavior to enhance, sustain, or at least avoid affecting the collective one. We test two different types of probabilistic adaptation: (i) with fixed probability p_F, (ii) with a probability p_D depending on the last evaluation. The former simply uses a fixed probability p_F to decide at each epoch whether to adapt. The latter dynamically modifies the adaptation probability p_D in function of the performance at the previous epoch, according to the following formula:

$$p_D = p_{max} \cdot (1 - P_{epoch}) \tag{5.3}$$

where p_{max} is an experiment parameter specifying the upper bound of the adaptation probability, and $P_{epoch} \in [0, 1]$ is the performance value at the previous evaluation. For simplicity, from now on, when talking about the value of adaptation probability, we will always refer to a generic parameter p. This indicates p_F in the case of the fixed approach and p_{max} in the case of the dynamic approach, as they are the values directly in our control (i.e., that we can set as parameters of the experiment).

A discount factor δ continuously "ruins" the evaluation of the best controller in order to replace it if its performance does not persist. This is done by updating its performance P_{best}, multiplying it by a value $\delta \in [0, 1]$:

$$P^{t}_{best} = \delta \cdot P^{t-1}_{best} \tag{5.4}$$

This permits a new controller with a performance lower than the original P_{best} but higher than

the *discounted* P_{best} to replace the best controller. However, iteratively discounting the best performance would eventually cause forgetting the best controller. We avoid this problem by updating its performance whenever it obtains a new higher evaluation[3]. This permits resetting the performance every time the best behavior obtains a new evaluation higher than its *discounted* performance. We propose the discount to tackle the problem of memorizing situationally good controllers; nevertheless, it also supports the adaptability of the robot to important changes in the environment requiring the discovery of completely different behaviors. For instance, if a controller loses effectiveness due to changes in the environment, its performance would not be confirmed and neither updated during re-evaluation; in this situation, the iterative discount would eventually decrease the controller evaluation enough for another, more effective behavior to take the title of "best" (assuming a better one exists).

5.2 Experimental setting

The task considered in this investigation aims at assessing the ability of the online adaptation mechanism to discover suitable collective behaviors in varying conditions. We imagine changes in the environment to affect the performance of the robots, requiring the emergence of novel *collective* behaviors. For simplicity, we abstract the problem to the discovery of two opposite collective behaviors: *aggregation* and *isolation*. The former requires the robots to gather and stay together. The latter requires each robot to avoid encounters. These two phases follow each other, requiring the group to behave differently during the course of the experiment.

To drive the adaptation, we endow each robot with an evaluation function that depends on the phase of the task. This considers the number of robots within the perception range (i.e., $10\,\mathrm{cm}$). Since two robots do not need to be in physical contact to be considered neighbors, the theoretical maximum number of neighbors a robot can perceive at the same time is eight, as the number of its sensors. This has some implications for the definition of the evaluation function. Indeed, maximizing the number of neighbors hinders the formation of compact clusters. This follows from the circle packaging problem, stating that the most dense way of packing "circles" is by hexagonal packing, that is, having six touching neighbors (Fukshansky, 2011). Addition-

[3] As previously said, the best controller is used by default.

ally, managing eight neighbors would require introducing non-linearity and asymmetry in the evaluation function, with six neighbors being the best value in the aggregation phase, and having six *or more* neighbors being the worst in the isolation phase. Therefore, we arbitrarily decide to relax the constraints, considering only up to four neighbors[4].

During the aggregation phase, each robot increases its performance by maximizing the number of neighbors over time, up to four. To calculate the performance value P_{step} at a specific instant, we compute the ratio between the perceived neighbors n and the maximum number of considered neighbors N (i.e., four). Finally, we limit P_{step} to a maximum value of one:

$$P_{step} = \min\left(\frac{n}{N}, 1\right) \tag{5.5}$$

During the isolation phase, each robot increases its performance by minimizing the number of perceived neighbors. In this latter scenario, the robot gets the minimum performance when it is surrounded by four or more neighbors. The performance calculation in this phase is therefore exactly the opposite of the one in the aggregation phase. This permits calculating the performance value P_{step} for the isolation phase as the dual of the performance in the aggregation phase:

$$P_{step} = 1 - \min\left(\frac{n}{N}, 1\right) \tag{5.6}$$

This dual evaluation function considers the performance of the robot in a specific instant of time: a step. However, the evaluation of a controller spans across multiple instants: an epoch. To evaluate a controller, the robot averages the performance over k multiple steps (see Equation 5.7). This means that a controller is evaluated for its overall behavior, and that the occurrence of lucky or unlucky step-evaluations is minimized.

$$P_{epoch} = \frac{1}{k}\sum_{i=0}^{k} P_{step,\,i} \tag{5.7}$$

The arena of the experiment is a dodecagon that approximates a circle of radius 2 m (see Figure 5.2). We chose an almost round shape to simplify the exit of the robots from corners, and

[4]Four neighbors are $2/3$ of the maximum number of neighbors touching the round chassis of the robot at the same time (i.e., six).

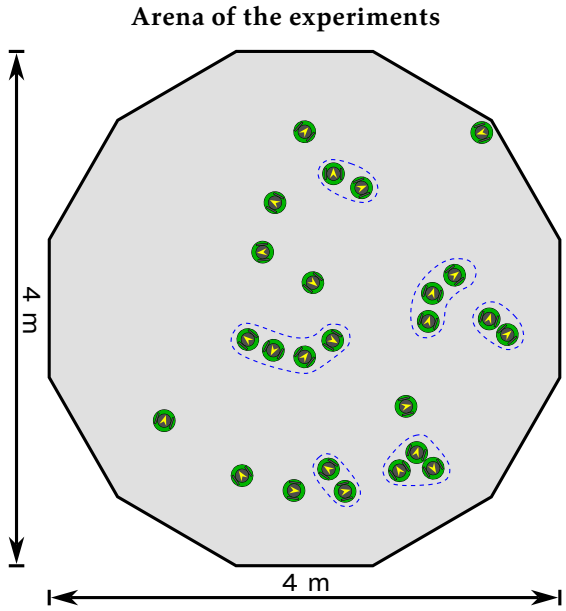


Figure 5.2: Representation of the arena of the experiment with the 25 robots inside. The blue dashed lines represent clusters of robots as computed during the simulation. Note that the walls are not considered while calculating the number of neighbors, and only prevent the robots from dispersing.

thus to avoid them blocking while trying new behaviors. The diameter is decided according to the epoch duration and the number of robots. Specifically, the robots have enough time to travel from one side of the arena to the other within the epoch duration. Additionally, this size permits the robots to find each other during the aggregation phase, while still being able to flee from encounters during the isolation phase.

We test the proposed model in the described task using a swarm of 25 *foot-bot* robots (Bonani et al., 2010; Bonani et al., 2013). Although relatively powerful, we constrain their sensing and acting capabilities in order to generalize to different types of robots and to simplify the adaptation. Specifically, the neural controller requires as input the values read by 8 proximity sensors placed homogeneously around the chassis of the robot, and outputs the control values of 2 differential steering wheel actuators. We set the perception range of the sensors to $10\,\mathrm{cm}$, and the maximum motor speed to $6.28\,\mathrm{cm/s}$. Additionally, we prevent the robots from communicating, avoiding the emergence of advanced adaptation protocols. We simulate the swarm in the ARGoS3 simulator, beta 59 (Pinciroli et al., 2012).

			Perturbation Intensity σ					
			Fixed			Dynamic		
			0.1	0.3	0.5	0.1	0.3	0.5
Adaptation Probability p	Fixed	0.1	$p = 0.1$ $\sigma = 0.1$	$p = 0.1$ $\sigma = 0.3$	$p = 0.1$ $\sigma = 0.5$	$p = 0.1$ $\sigma = 0.1$	$p = 0.1$ $\sigma = 0.3$	$p = 0.1$ $\sigma = 0.5$
		0.3	$p = 0.3$ $\sigma = 0.1$	$p = 0.3$ $\sigma = 0.3$	$p = 0.3$ $\sigma = 0.5$	$p = 0.3$ $\sigma = 0.1$	$p = 0.3$ $\sigma = 0.3$	$p = 0.3$ $\sigma = 0.5$
		0.5	$p = 0.5$ $\sigma = 0.1$	$p = 0.5$ $\sigma = 0.3$	$p = 0.5$ $\sigma = 0.5$	$p = 0.5$ $\sigma = 0.1$	$p = 0.5$ $\sigma = 0.3$	$p = 0.5$ $\sigma = 0.5$
	Dynamic	0.1	$p = 0.1$ $\sigma = 0.1$	$p = 0.1$ $\sigma = 0.3$	$p = 0.1$ $\sigma = 0.5$	$p = 0.1$ $\sigma = 0.1$	$p = 0.1$ $\sigma = 0.3$	$p = 0.1$ $\sigma = 0.5$
		0.3	$p = 0.3$ $\sigma = 0.1$	$p = 0.3$ $\sigma = 0.3$	$p = 0.3$ $\sigma = 0.5$	$p = 0.3$ $\sigma = 0.1$	$p = 0.3$ $\sigma = 0.3$	$p = 0.3$ $\sigma = 0.5$
		0.5	$p = 0.5$ $\sigma = 0.1$	$p = 0.5$ $\sigma = 0.3$	$p = 0.5$ $\sigma = 0.5$	$p = 0.5$ $\sigma = 0.1$	$p = 0.5$ $\sigma = 0.3$	$p = 0.5$ $\sigma = 0.5$

Table 5.1: The table with the 36 tested parameter configurations.

We are interested in comprehending the effect of different values of adaptation probability p and perturbation intensity σ in the proposed model. Additionally, we want to understand if fixed values are enough to lead the swarm to the desired behavior, or if dynamic values are needed. Therefore, we set up a grid search to evaluate the effect of the two types of adaptation probability and perturbation intensity, initialized with different values. Specifically, we test Fixed and Dynamic Adaptation Probability (APF and APD, respectively) and Fixed and Dynamic Perturbation Intensity (PIF and PID, respectively). We select the adaptation probability $p \in 0.1, 0.3, 0.5$ and the perturbation intensity $\sigma \in 0.1, 0.3, 0.5$. We set the discount factor of the best performance $\delta = 0.95$. Overall, we test 36 different configurations of the model (see Table 5.1).

Finally, from the results presented in Chapter 4, we consider the possibility that switching from the aggregation to the isolation phase is different from the other way around. To avoid an unfair analysis, we set our experiment as the sequence of aggregation, isolation, and again aggregation. This requires the swarm to find the antipodal collective behavior starting from both situations.

To collect stable results, we run 31 replicas of the experiment with different seeds. Each replica is composed of 30'000 epochs, each lasting 500 steps (i.e., 50 s).

5.3 Results

To correctly analyze the effectiveness of the adaptation, we consider the performance of the robots throughout the experiment. We confirm the compliance with the desired goal by verifying the size of the largest cluster formed. Additionally, we evaluate the heterogeneity of the robot controllers over time to understand if they tend to uniformize. This should help to understand if a unique solution to the problem exists, or if the desired behavior emerges from the interaction of heterogeneous individuals.

The first consideration that we can derive from the results is that some parameters permit the swarm to succeed in both the aggregation and isolation tasks (see Figure 5.3). The degree of success clearly varies according to the parameters used. Specifically, the parameter that seems to impact the most is the type of perturbation intensity: *fixed* or *dynamic* (see Figure 5.4). With a fixed perturbation intensity, we observe mostly stagnating or even decreasing performance. Conversely, with a dynamic perturbation intensity, the performance is always increasing. The presence of a dynamic adaptation probability also improves the performance and positively affects the trend driven by the perturbation intensity. Indeed, we obtain the best performance when both parameters are dynamic (see the bottom-right area of Figure 5.3). Even the upper bounds of adaptation probability and perturbation intensity seem to affect the performance of the swarm (see Figure 5.5). Overall, increasing the upper bounds seems to increase the average performance. This is accentuated in the perturbation intensity, while the adaptation probability shows differences mostly comparing 0.1 with 0.3 and 0.5. Another effect of selecting different upper bound values is the speed of the adaptation: 0.3 and 0.5 present indeed a faster increase in performance compared with 0.1.

To better appreciate the improvement of dynamic adaptation probability and perturbation intensity with upper bound 0.5, we need to look at their performance distribution (see Figure 5.6). Specifically, we focus on the performance at the beginning of the experiment and at the end of each task phase (i.e., aggregation and isolation). As is visible, the performance distribution of this configuration is overall better than all the others. Only upper bound values of 0.3 for both adaptation probability and perturbation intensity generate similar distributions, that are, however, worse when considering their median. This is somehow counterintuitive, as

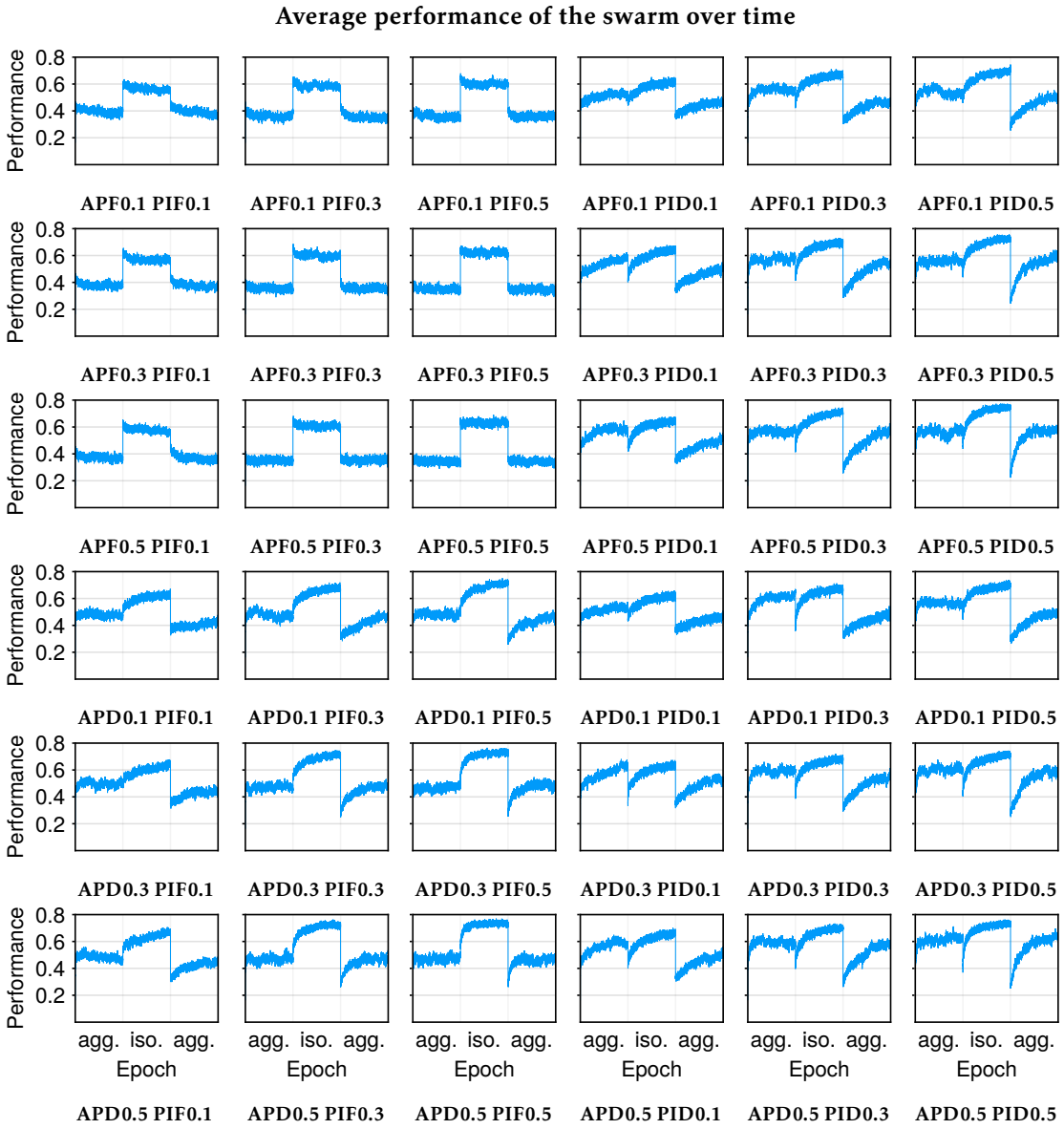
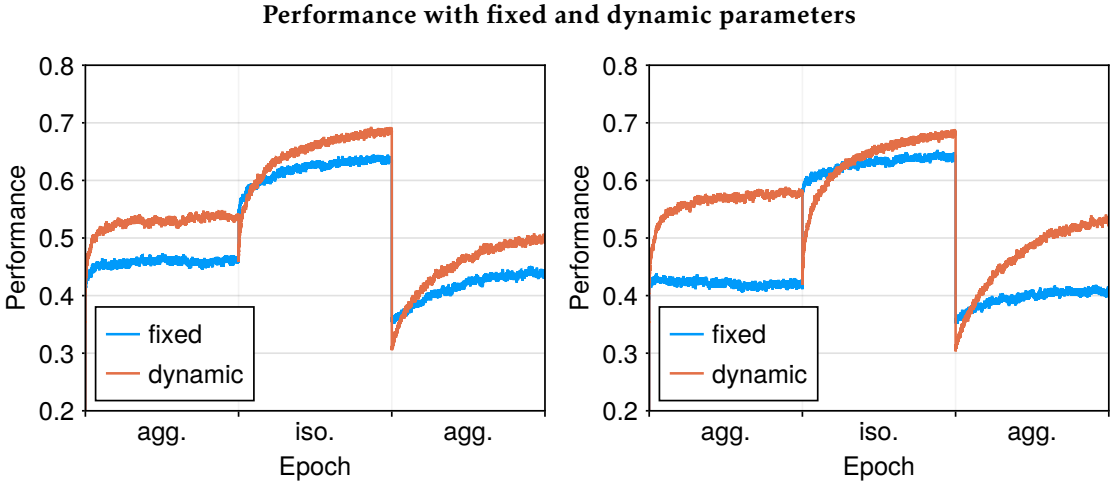


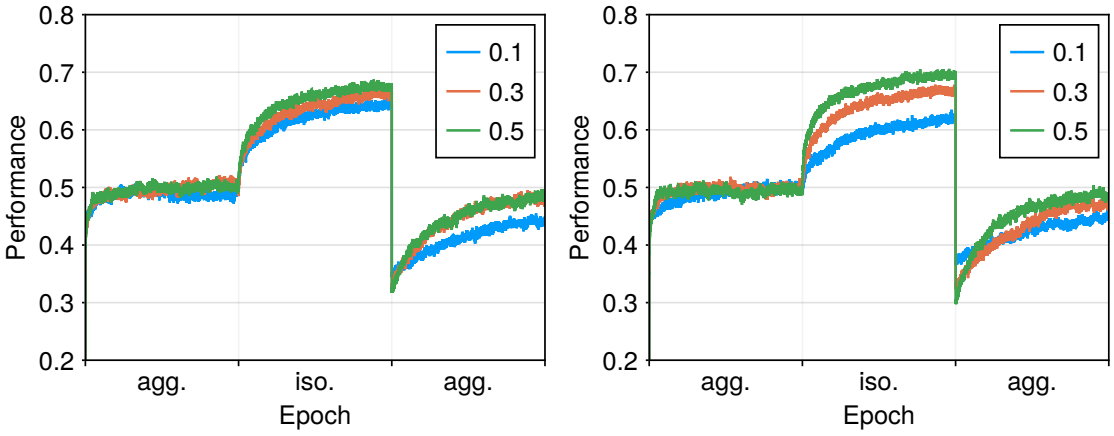
Figure 5.3: The average performance of the swarm over time according to the parameter settings. The columns contain the type and intensity of the perturbation: *fixed 0.1, 0.3, 0.5* and *dynamic 0.1, 0.3, 0.5*, respectively. The rows contain the type and probability of the adaptation: *fixed 0.1, 0.3, 0.5* and *dynamic 0.1, 0.3, 0.5*, respectively. All the plots have the same axes range and meaning, indicated on the left- and bottom-most plots.

we would expect a more gentle adaptation to perform better. The reason is probably that the dynamic adaptation probability p_D and perturbation intensity σ_D decrease with the increase in performance. For instance, with a dynamic adaptation probability with upper bound value



(a) Adaptation Probability **(b) Perturbation Intensity**

Figure 5.4: Performance comparison according to *fixed* and *dynamic* values of (a) adaptation probability and (b) perturbation intensity. The two trends in each plot are the result of averaging over the 18 different experimental configurations sharing the corresponding value: fixed and dynamic.



(a) Adaptation Probability **(b) Perturbation Intensity**

Figure 5.5: Performance comparison according to the upper bound values 0.1, 0.3, 0.5 of (a) adaptation probability and (b) perturbation intensity. The three trends in each plot are the result of averaging over the 12 different experimental configurations sharing the corresponding value: 0.1, 0.3, 0.5.

$p_{max} = 0.3$, a performance of 0.5 corresponds to a probability to adapt $p_D = 0.15$. Conversely, the same performance with dynamic adaptation probability with upper bound value

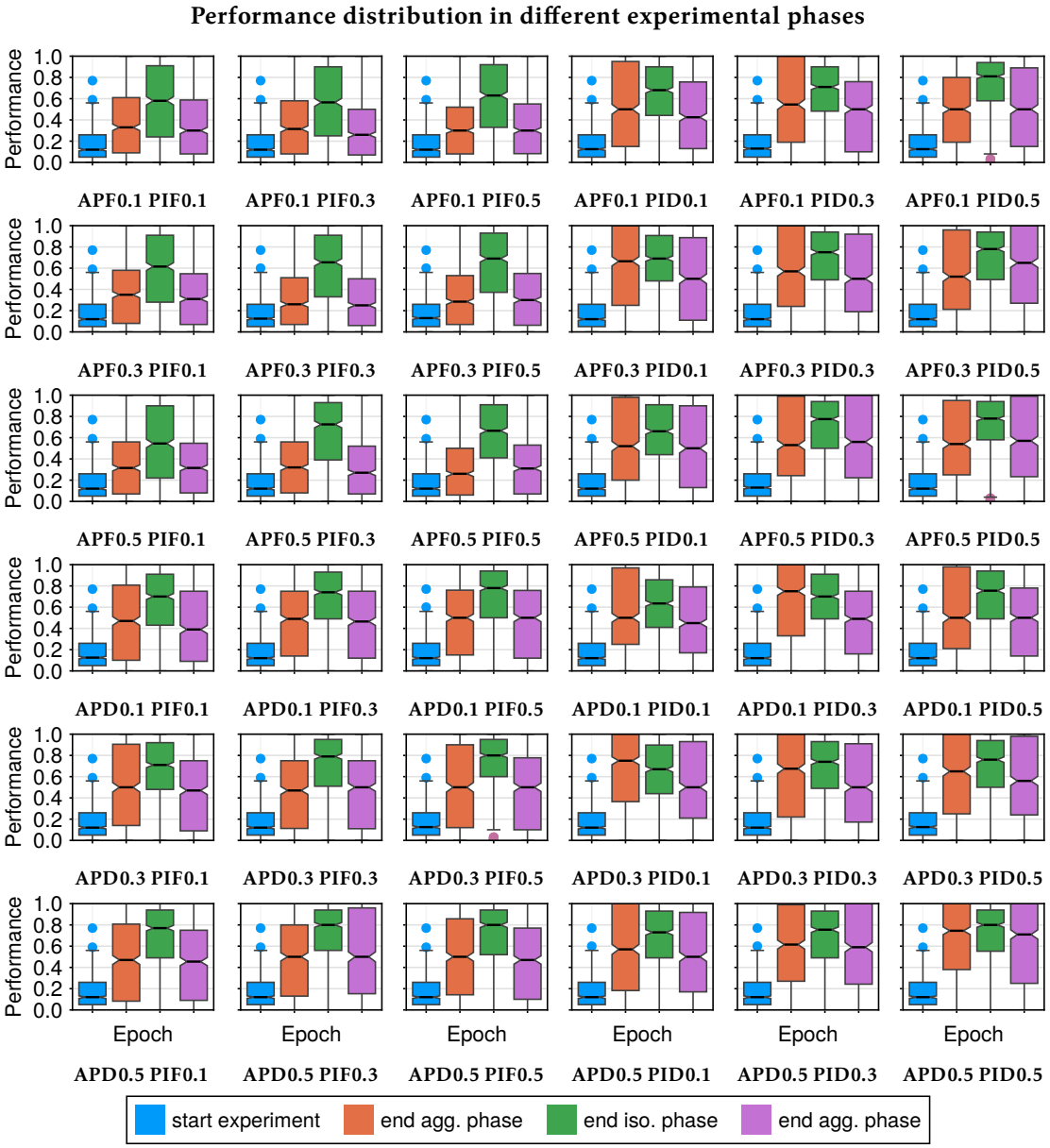


Figure 5.6: The distribution of robots performance at the beginning and end of the task phases, according to the various parameter settings. The columns contain the type and intensity of the perturbation: *fixed 0.1, 0.3, 0.5* and *dynamic 0.1, 0.3, 0.5*, respectively. The rows contain the type and probability of the adaptation: *fixed 0.1, 0.3, 0.5* and *dynamic 0.1, 0.3, 0.5*, respectively. All the plots have the same axes range and meaning, indicated on the left and bottom plots.

$p_{max} = 0.5$ corresponds to a probability to adapt $p_D = 0.25$. The former may end up being too low for not yet good behaviors, slowing down the adaptation and preventing the discovery

of better behaviors.

Analyzing the performance permits comparing different parametrizations of the experiment. Nevertheless, it does not provide a clear indication of whether the adaptation produces the expected results at the group level: *aggregation* and *isolation*. Therefore, we propose analyzing the size of the largest cluster at each epoch of the experiments. This is computed by finding groups of connected robots, whereby connected indicates robots that are in the perception range, $+10\%$ to face noise and accept temporary oscillations[5]. The results clearly confirm the previous results on the performance (see Figure 5.7). Additionally, they highlight a faster and more effective recovery in cluster size with perturbation intensity $\sigma = 0.5$ compared to the results obtained by using a value $\sigma = 0.3$. This supports the idea that a higher value of the perturbation intensity upper bound produces better results.

One claim of this investigation is that the robots composing the swarm are heterogeneous. This is undeniable at the start of the experiment, as each robot has a different, randomly created controller. Nevertheless, it is interesting to understand if the adaptation tends to homogenize the controllers or if it instead enhances their differences. We therefore analyze the difference among the controllers in each swarm at the beginning and at the end of the experiment (see Figure 5.8). We calculate the difference as the Euclidean distance between the weight vectors of each couple of neural networks in a swarm. Finally, we calculate the difference between the final distance and the initial distance of each couple. The distribution of the distance variation clearly shows that the heterogeneity within the same swarm increases during the experiment. Indeed, just a few deltas are negative, indicating two controllers that became more similar during the experiment. Interestingly, all those cases belong to experiments involving a dynamic perturbation intensity equal to 0.1, again suggesting that this value is too low to produce any interesting outcome. Finally, these results indicate that the adaptation tends to discover a unique controller for each robot, still permitting the emergence of complex collective behaviors.

[5]The cluster measure is more permissive than the performance measure.

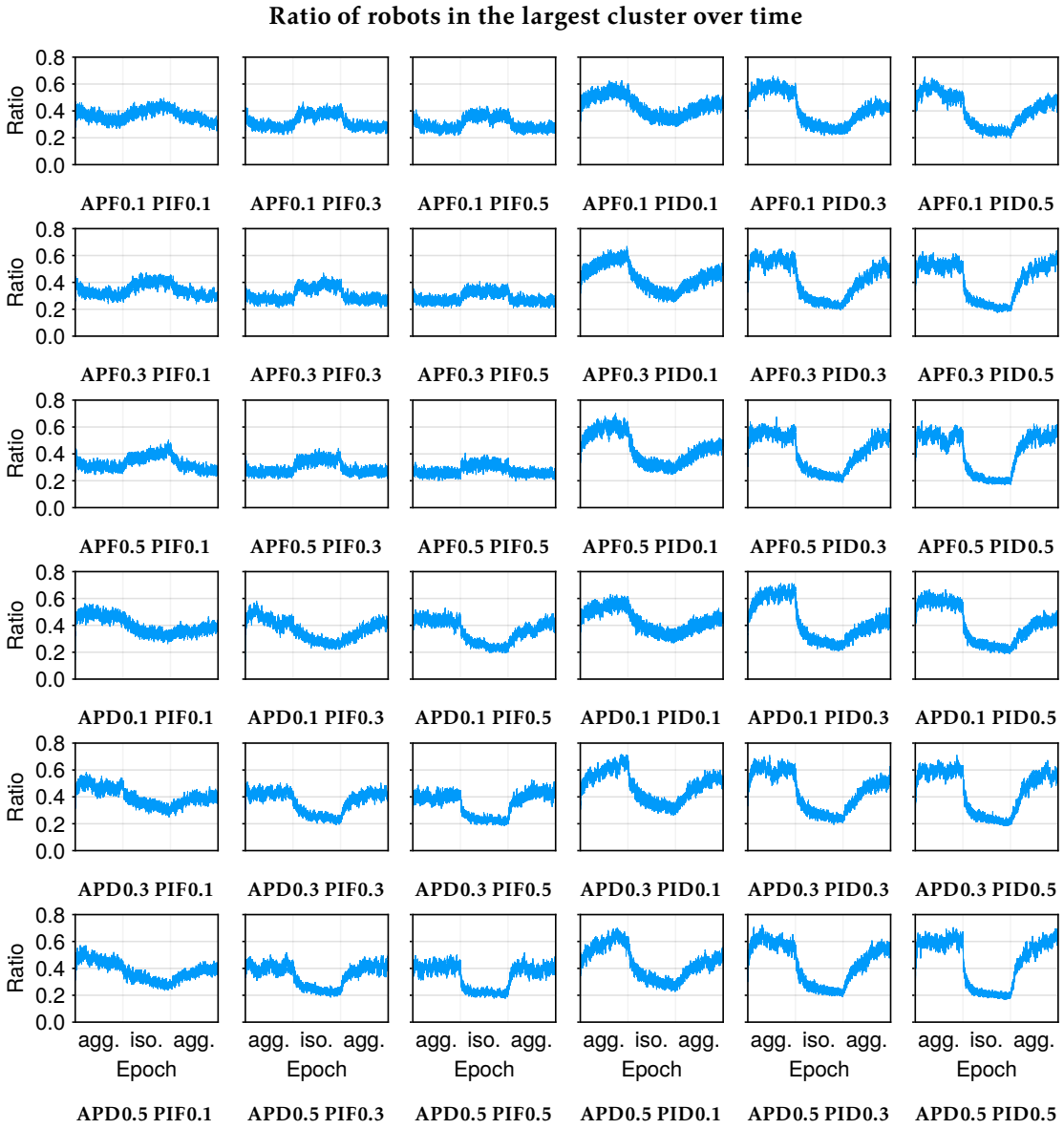


Figure 5.7: Average ratio of robots belonging to the largest cluster at each epoch, according to the various parameter settings. The columns contain the type and intensity of the perturbation: *fixed 0.1, 0.3, 0.5* and *dynamic 0.1, 0.3, 0.5*, respectively. The rows contain the type and probability of the adaptation: *fixed 0.1, 0.3, 0.5* and *dynamic 0.1, 0.3, 0.5*, respectively. All the plots have the same axes range and meaning, indicated on the left and bottom plots.

5.4 Discussion

The goal of this investigation is to devise an adaptive mechanism for the co-adaptation of robots belonging to a heterogeneous swarm. Reaching this goal required identifying and solving some

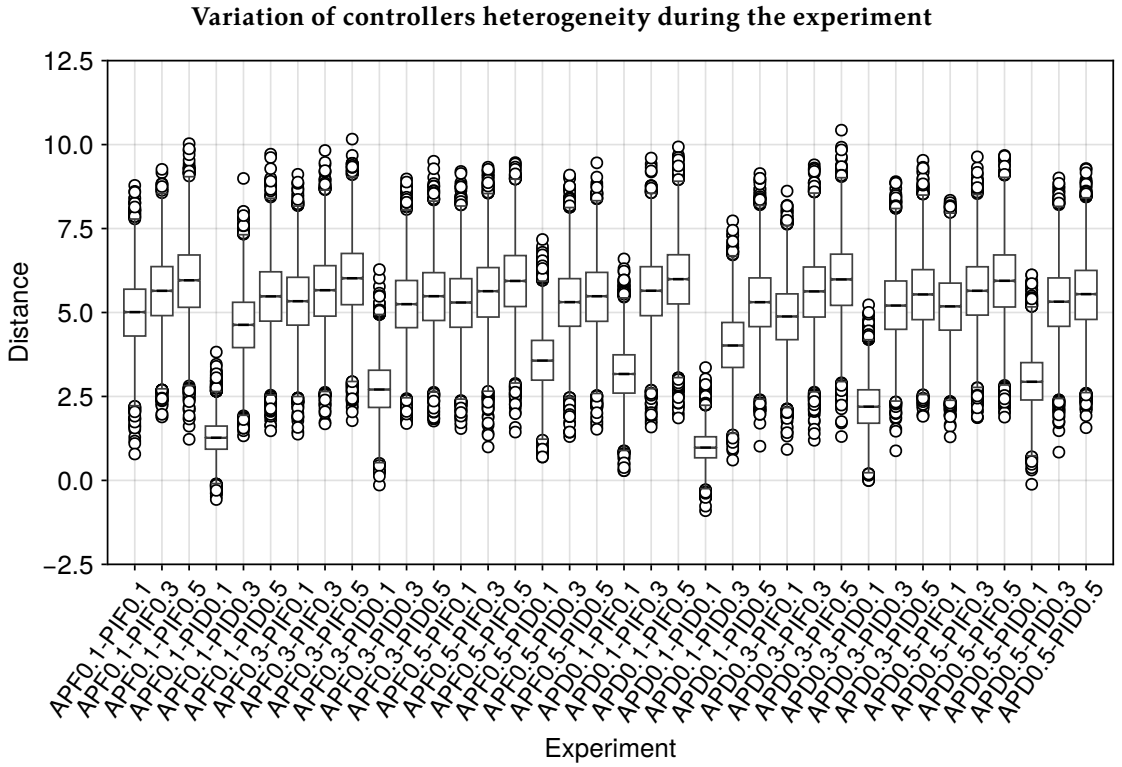


Figure 5.8: Difference in the heterogeneity of the robots controllers between the end and the start of the experiment. Given two robots r_a and r_b, we take their respective controllers c_a and c_b at the start of the experiment, and the controllers c'_a and c'_b at the end. We then calculate the euclidean distance d between c_a and c_b, and d' between c'_a and c'_b. We compute a value $\Delta = d' - d$ to evaluate the change in the degree of heterogeneity between the two controllers. Finally, we plot the distribution of the Δs for each of the experiment configurations.

general issues and difficulties. We immediately noticed that a constant adaptation is not an indicated approach. This is undesirable already for single robots, as it does not balance exploration and exploitation of the discovered behavior, focusing only on the former. In a swarm of concurrently adapting robots, it also prevents understanding what the collective performance is. Different individuals might have discovered a suitable behavior in different epochs, preventing to observe the collective one[6]. This complicates even the adaptation process itself, as adapting an individual behavior to a continuously changing collective behavior becomes more complex. To tackle these problems, we propose to systematically reuse the best behavior found by each robot, permitting to identify the emerging collective behavior. However, reusing the best be-

[6]During a constant adaptation, the best behavior found is never used again.

havior does not necessarily imply an effective adaptation. This also needs to be asynchronous to permit evaluating the new behaviors in a swarm mostly using its best ones. The goal is to mitigate the effect of *situationality*, that complicates the evaluation of the individual robot performance when neighbors interfere, especially in a disruptive way. We tackle this problem by introducing a probability to adapt, inducing just some individuals to adapt in a specific instant. The desired case is that of most of the swarm using its best behavior, and a few weaker individuals trying to adapt to it. This also suggests that well-performing robots should reuse their best behavior more often than their poorly behaving counterparts, which should instead focus on improving themselves. Intuitively, this links the probability to adapt to the performance, with the former decreasing when the latter increases. Our experiments confirm that this dependency improves the developed collective behavior with respect to a fixed value. Additionally, we argue that effective controllers should be modified mildly and weak controllers aggressively. The idea is that the formers are already close to a good solution and thus require smaller adjustments to improve, while the latter are potentially far from it and might benefit from a wider exploration. Also, this approach might permit maintaining the stability of the system even when essential individuals adapt, potentially further reducing the effect of disruptive situationality. We implement this by modulating the perturbation intensity of the controller of the robot according to its performance. The results support our thesis, showing some improvement in the collective performance.

We argued that *situationality* can slow down adaptation by affecting the performance of effective behaviors. We, therefore, proposed to introduce an asynchronous performance-dependent adaptation to mitigate the problem. Nevertheless, *situationality* can have the same detrimental effect also by enhancing the performance of poor behaviors. To mitigate this problem, we propose to continuously discount the performance of the best-known behavior if it is not reconfirmed. The idea is to penalize behaviors that are good only in specific and rare situations. Dually, the discount also permits forgetting obsolete behaviors when the task, the environmental conditions, or the robot itself change. Additionally, the discount factor avoids a possible problem related to averaging old and new performances as proposed in Chapter 3 and Chapter 4, that is, the risk to over-consider unlucky epochs attaining a low performance. Preliminary experiments shown that discounting is essential for the emergence of the collective behavior of

the swarm. The success of the mechanism in the proposed dual-task also seems to confirm this need.

In our experiments, we assessed the need for all the aforementioned factors in order to use online adaptation with swarm robotics. We confirm that all contribute to the emergence of the desired swarm behaviors, albeit to varying degrees. Additionally, we argue that our considerations and strategies are applicable to a broad spectrum of missions, and thus, they are not limited to a specific condition. Still, some difficulties in combining online adaptation and swarm robotics remain. Specifically, we expect the evaluation function of the robots to play an important role. In our task, improving each robot performance has a positive impact on the performance of the swarm. Nevertheless, we can identify a class of missions for which enhancing the collective performance requires hindering the individual one. An example is that of task partitioning, in which some robots do not have any direct or immediate reward but are essential for the group to operate effectively. This type of relationship is difficult to capture with local evaluation functions and currently limits the applicability of online adaptation in some swarm robotics tasks (Birattari, 2024). Even considering addressable tasks, we still might encounter some difficulties. Most of the mechanisms we propose assume a performance in $[0, 1]$, but this assumption is not valid in every situation. Some tasks might not have any known upper or lower bound of performance, complicating the choice of when to adapt and with what intensity. A possible solution might be using communication to share information about upper and lower bounds encountered by different robots. Currently, however, this is a problem yet to be solved.

An additional difficulty of using online adaptation in swarm robotics is connected to the intrinsic emergence of heterogeneity among robots. Indeed, as each robot modifies its own controller according to its own performance, the difference among the resulting behaviors potentially increases with time. This introduces additional complexity in the swarm interactions, complicating the emergence of a desired collective behavior with respect to homogeneous or partially heterogeneous systems. Yet, this might be extremely important to solve complex tasks with simple robots. Additionally, recent works suggest that heterogeneity enhances the performance, stability, and robustness of the whole system (Kengyel et al., 2015). Our investigation does not explicitly focus on those aspects, but instead demonstrates that obtaining a collective behavior online with heterogeneous controllers is possible.

An interesting aspect of our investigation is that it can be generalized to a variety of different situations. Specifically, we envision the evaluation function of the robots to consider the internal variables needed for surviving (Ashby, 1962; Braccini et al., 2024b; see Chapter 4). The robots would then change their behavior so as to maximize the chances of survival. Let's consider, for instance, a group of robots in a harsh environment with great thermal excursion, such as the moon. There, the variation between daytime and nighttime temperature ranges from $\sim$ 150°C at the poles to $\sim$ 300°C at the equator (J.-P. Williams et al., 2017). Additionally, let's imagine that each robot recharges its battery with solar panels. We could formulate our task in function of the battery level and the survival temperature. This pushes the robots to search for a strategy that makes it possible to maintain a good battery level and to avoid freezing. During the night, the robots would aggregate, minimizing the thermal dispersion and thus the battery consumption, trying to resist until daytime. During the day, the robots would isolate to capture as much thermal radiation as possible while carrying on their duties. Similar results can also arise from different conditions, such as environments with great thermal excursion in space rather than in time. It follows that the two collective behaviors that we present in this investigation can also result from different types of evaluation functions, and can be used to solve practical issues that cannot always be foreseen.

As a final point of this discussion, we want to highlight the simplicity and technical minimalism of the mechanism that we propose. Our robots use only local perception and cannot communicate. The controller is a minimal artificial neural network, and the evaluator simply averages instantaneous performances. The adaptation module consists of a stochastic weights modifier and a memory to store the best performance recorded and the best weights. We claim that this setup can be realistically implemented even in cheap miniaturized robots, keeping the option for mass production of these adaptive systems open.

In this chapter, we investigate the use of online adaptation in swarm robotics. Specifically, each robot adapts so that a collective behavior at the swarm level can emerge. Additionally, the task we investigate permits to assess the adaptation to different environmental conditions throughout the experiment. The results are promising, but the experiment itself is long. This is because finding multiple individual behaviors composing a collective one is not trivial. Therefore, one of the next investigations should consider increasing the speed of adaptation, even

at the expense of flexibility. In the next chapter, we will propose a system combining offline design and online adaptation of robot controllers with the aim of reducing the adaptation time when some information about the environment is known or can be assumed.

5.5 Chapter highlights

- With some precautions and enough time, online adaptation enables co-adaptation.
- Asynchronous adaptation is essential to enable each robot to adapt to the best behavior of the group.
- Well-behaving robots should adapt less frequently and aggressively than poorly-behaving ones.
- We coin the term *situationality* to indicate factors that affect the self-evaluation of robots, leading to a wrong estimate of controller effectiveness.
- Independently adapting while favoring the group performance requires being able to estimate the impact of one's own actions on the group.

Chapter bibliography

Ashby, W.R. (1962). "Principles of the self-organizing system." In: *Principles of Self-Organization: Transactions of the University of Illinois Symposium*. Vol. 7. Pergamon Press, pp. 255–278.

Baldini, P., M. Braccini, and A. Roli (2023a). "Online Adaptation of Robots Controlled by Nanowire Networks: A Preliminary Study." In: *Artificial Life and Evolutionary Computation. 16th Italian Workshop, WIVACE 2022, Gaeta, Italy, September 14–16, 2022, Revised Selected Papers*. Vol. 1780. Springer Nature Switzerland, pp. 171–182.

Baldini, P., A. Roli, and M. Braccini (2026b). "Online adaptation of heterogeneous robot swarms to a dynamic task." In: *Swarm Intelligence*. Submitted.

Birattari, M. (2024). *Personal communication*.

Bonani, M. et al. (2010). "The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research." In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Institute of Electrical and Electronics Engineers, pp. 4187–4193.

Bonani, M. et al. (2013). "Physical Interactions in Swarm Robotics: The Hand-Bot Case Study." In: *Distributed Autonomous Robotic Systems: The 10th International Symposium*. Vol. 83. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 585–595.

Braccini, M., A. Roli, and S. Kauffman (2022a). "A Novel Online Adaptation Mechanism in Artificial Systems Provides Phenotypic Plasticity." In: *Artificial Life and Evolutionary Computation. 15th Italian Workshop, WIVACE 2021, Winterthur, Switzerland, September 15–17, 2021, Revised Selected Papers*. Vol. 1722. Springer Nature Switzerland, pp. 121–132.

Braccini, M. et al. (2024b). "Sensory–Motor Loop Adaptation in Boolean Network Robots." In: *Sensors* 24.11, p. 3393.

Fukshansky, L. (2011). "Revisiting the hexagonal lattice: on optimal lattice circle packing." In: *Elemente der Mathematik* 66.1, pp. 1–9.

Kengyel, D. et al. (2015). "Potential of Heterogeneity in Collective Behaviors: A Case Study on Heterogeneous Swarms." In: *PRIMA 2015: Principles and Practice of Multi-Agent Systems*. Vol. 9387. Springer International Publishing, pp. 201–217.

Pinciroli, C. et al. (2012). "ARGoS: a Modular, Parallel, Multi-Engine Simulator for Multi-Robot Systems." In: *Swarm Intelligence* 6.4, pp. 271–295.

Williams, J.-P. et al. (2017). "The global surface temperatures of the Moon as measured by the Diviner Lunar Radiometer Experiment." In: *Icarus* 283, pp. 300–325.

6 Enhancing offline design

In the previous chapters, we proposed situations in which online adaptation is of primary importance. The characteristic that all share is the unavailability of information. A system able to autonomously change its own behavior to fulfill a goal is therefore very effective in this case. Nevertheless, there are situations in which we have a very good knowledge and expect it to be stable over time. For these specific instances, it would be desirable to have a reliable system able to adapt only to the little information that we do not know.

In this chapter, we investigate the combination of offline design with online adaptation (Baldini et al., 2026a). The aim is to produce a reliable and predictable system able to perform minimal behavioral adjustments in a partially known environment or mission. We begin by describing the system used to control the robot, the adaptation mechanism, and the task. Then, we present and discuss the results.

6.1 Controller, adaptive mechanism, and design system

In this section, we discuss the tools used in our experiments. We first describe the characteristics of the robot employed. Following, we describe the tool used for the offline design of the robot control software. This step also dictates how the control software adapts at runtime. Finally, we describe the strategy used to drive the adaptation online.

6.1.1 Robotic platform

The robot we use in the experiments is the e-puck (Mondada et al., 2009). It possesses several sensors, a communication system, and two motor actuators. However, some of its capabilities are superfluous for the class of tasks that we aim to solve, while others cannot technically be used concurrently[1]. Therefore, we decide to work on a subset of functionalities that are useful for our goals and that can be properly used together. We define this set of functionalities in a new *reference model* (Hasselmann et al., 2018b). A reference model is a formal specification of the robot platform for which an automatic design method can produce control software. It defines the inputs and outputs of the control architecture and how they map to the hardware. This means that the reference model can even re-elaborate different types of sensorial input to produce the same type of information (e.g., interpreting LIDAR or proximity data as recognizable obstacles). In the same way, it can use output commands to control different types of actuators (e.g., direction to motor or leg actuation). It follows that different types of robots having different hardware capabilities can share the same abstract inputs and outputs thanks to their reference model mapping (Kegeleirs et al., 2024).

The reference model we propose for this investigation is RM3.S, a variation of RM3 used in Garzón Ramos et al. (2020). Both use the omnidirectional camera and the RGB-LEDs to share and capture information. This disables the use of light sensors, leaving proximity and ground as the only remaining types of perception. Also, RM3.S uses the Range and Bearing (RAB) to communicate and receive some basic information, such as the will to perform an action and its resulting feedback. Finally, RM3 pre-processes some sensory cues to produce abstract, summarized inputs, while RM3.S uses less processed signals. For instance, while RM3.S returns the instantaneous ground color, RM3 returns the average ground color in the previous 5 computation steps. Conceptually, the main difference between the two reference models is that RM3 is intended to work with swarms of robots, while we design RM3.S to work with a single robot[2].

Going into details, Table 6.1 describes RM3.S. The data $prox_{i\in[1,8]}$ from the proximity sensors consists of the angle α of the sensor direction with respect to the front of the robot, and the

[1] For instance, the light sensors, the LEDs, and the camera all use the same type of physical signal: light, and thus can interfere with each other if not used consciously.

[2] The "S" in "RM3.S" stands indeed for "Single".

Input	Value	Description
$prox_{i\in[1,8]}$	$v \in [0, 1]$	The value v perceived by proximity sensor i and
	$\alpha \in [-\pi, \pi]$	its position α w.r.t. the front of the robot.
$gnd_{i\in[1,3]}$	$v \in [0, 1]$	The value v perceived by ground sensor i.
cam	$c \in \{R, G, B, C, M, Y\}$	The color c of the blob perceived by the camera,
	$\alpha \in [-\pi, \pi]$	its position α w.r.t. the front of the robot, and
	$\delta \in [0, 35]$	its distance δ in cm. Finally, the blob area s in
	$s \in (0, 95]$	pixels.
msg	$\alpha \in [-\pi, \pi]$	The information D received as a message from
	$D \in [0, 2^{32}]$	other robots and / or entities at distance δ in cm
	$\delta \in [0, 10]$	and angle α w.r.t. the front of the robot.
Output	Value	Description
$vel_{d\in\{l,r\}}$	$v \in [-12, 12]$	The linear velocity v in $\mathrm{cm/s}$ of the wheel d.
$led_{i\in[1,8]}$	$c \in \{\varnothing, R, G, B, C, M, Y\}$	The color c of LED i.
msg	$D \in [0, 2^{24}]$	The message D to broadcast.

Table 6.1: Inputs and outputs of the Reference Model 3.S.

normalized distance v of the perception. Perceiving an object at distance greater or equal than $\sim 5\,\mathrm{cm}$ results in a value $v = 0$, while an object at distance $0\,\mathrm{cm}$ results in a value $v = 1$[3]. The ground data $gnd_{i\in[1,3]}$ contains the value perceived below the robot, in shades of gray normalized in $[0, 1]$ where 1 indicates white and 0 indicates black. Whenever the camera perceives a color, the reference model produces an input containing the color of the blob, its angle with respect to the front of the robot, its distance, and its area. We limit the perception range of the camera to $35\,\mathrm{cm}$. The maximum area of the perceived blob depends on its distance, from 1 for extremely far colors to $\sim$95 for near ones[4]. The robot receives 4-byte messages through the RAB from a maximum distance of $10\,\mathrm{cm}$. Those contain a payload, plus the distance and position of the sender. We reserve the first byte of the payload to identify the sender, and use the 3 remaining bytes for the message data. The output of the reference model is the wheels linear velocity, the LEDs color, and a potential message to broadcast.

[3] https://github.com/demiurge-project/argos3-epuck/blob/v48/src/plugins/robots/e-puck/simulator/epuck_proximity_default_sensor.cpp#L137

[4] https://github.com/demiurge-project/argos3-epuck/blob/v48/src/plugins/robots/e-puck/simulator/epuck_omnidirectional_camera_sensor.cpp#L77

6.1.2 AutoMoDe-Lahmacun

AutoMoDe is a framework for the automatic design of control software for robots. It uses an automatic configuration algorithm to compose and parametrize simple modules so as to produce complex behaviors. It is known for its use in swarm robotics, where it managed to overtake human developers in the design of effective control software. Here we present AutoMoDe-Lahmacun, an implementation that permits the creation of control software able to adapt to different situations. Indeed, in addition to designing the control software of the robot, it also selects some parameters to be adapted online. The selected parameters are those that permit, according to offline simulations, the adaptation to the variety of missions that the robot will encounter during operation.

AutoMoDe-Lahmacun operates by composing a set of software modules into a probabilistic finite state machine (FSM). Two types of modules exist: behaviors and transitions. A behavior controls the actuation of the robot according to its perception, following a predefined logic decided by the programmer. The control logic of a behavior is minimal and simply not sufficient to complete complex tasks or missions. Only one behavior is active at any time. A transition controls, instead, the change in the active behavior according to some conditions. It does not affect the actuation of the robot, but only dictates the change in the active behavior according to perception. AutoMoDe-Lahmacun uses 3 behaviors and 6 transitions to compose the finite state machine, for a total of 9 software modules (see Table 6.2). This version of AutoMoDe uses fewer modules than some of its predecessors. The motivation is that we remove all the behaviors and transitions intended for a swarm and combine others together.

The three low-level behaviors used in Lahmacun are (i) `EXPLORATION`, (ii) `REACT-TO-COLOR`, and (iii) `STOP`. In `EXPLORATION`, the robot moves randomly according to a pre-defined strategy. The first strategy consists in always going straight and turning whenever it detects an obstacle. The second strategy makes the robot move straight for a number of steps sampled from a Lévy distribution (Codling et al., 2008; Feola et al., 2022), performing a turn at the end or when it encounters an obstacle. We select the strategy to use, the maximum number of turning steps, and the parameters controlling the Lévy distribution, respectively with t, θ, μ, and σ. In `REACT-TO-COLOR`, the robot reacts to blobs of color it perceives. The parameter t decides

Low-level behavior	Parameter	Description
EXPLORATION	t, θ, μ, σ	Movement with different random walk strategies.
REACT-TO-COLOR	t, i, c	Flee or approach a perceived color.
STOP		No actuation.
Transition	**Parameter**	**Description**
FIXED-PROBABILITY	p	Probabilistic transition.
FLOOR-COLOR	p, t, v	Probabilistic transition enabled by a specific floor color beneath the robot.
BLACK-FLOOR	p	Probabilistic transition enabled by a black floor color beneath the robot.
GRAY-FLOOR	p	Probabilistic transition enabled by a gray floor color beneath the robot.
WHITE-FLOOR	p	Probabilistic transition enabled by a white floor color beneath the robot.
COLOR-PERCEPTION	p, t, c	Probabilistic transition enabled by a specific color perceived by the camera.

Table 6.2: Behavior and transition modules used in AutoMoDe-Lahmacun.

whether to flee or approach a specific color c or *any* color among those defined in RM3.S. When reacting, the robot still has to avoid obstacles. The parameter i decides how intensely the robot wants to reach or flee the color, and thus how near it can get to an obstacle before significantly changing trajectory. In STOP, the robot simply stands still without moving. This behavior has no parameters.

AutoMoDe-Lahmacun uses six different transition modules: (i) FIXED-PROBABILITY, (ii) FLOOR-COLOR, (iii) BLACK-FLOOR, (iv) GRAY-FLOOR, (v) WHITE-FLOOR, and (vi) COLOR-PERCEPTION. FIXED-PROBABILITY permits a transition between behaviors with a fixed probability p. This transition does not depend on the robot state or perception, being simply stochastic. In BLACK-FLOOR, GRAY-FLOOR, WHITE-FLOOR, and FLOOR-COLOR the transition depends on the floor color. The first three respectively depend on black, gray, and white colors. The latter combines the formers in a single behavior that reacts to the color v specified as a parameter. The parameter t determines whether perceiving or not the color enables the transition. When the color condition holds, the transition becomes active according to the probability p specified as a parameter. Finally, COLOR-PERCEPTION enables the transition if the camera (i) does not perceive any color, (ii) perceives any color, (iii) perceives a specific color c, (iv) does not perceive a specific color c.

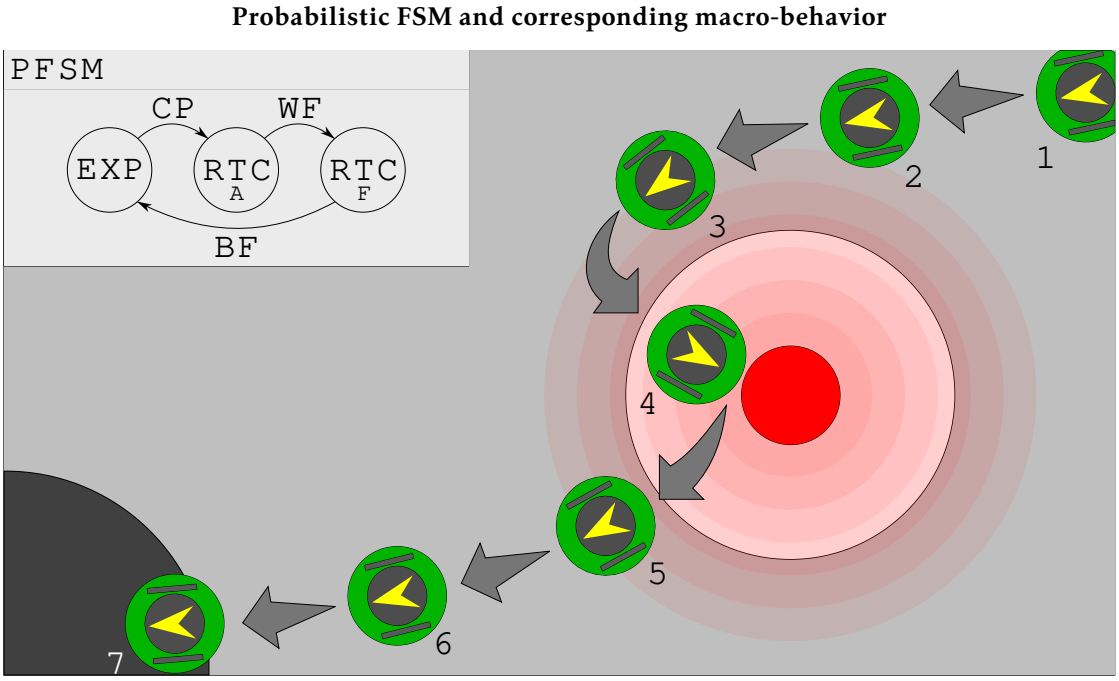


Figure 6.1: Example of a probabilistic finite state machine and corresponding macro-behavior when controlling an e-puck. The robot starts in position 1 with the `EXPLORATION` behavior. When it perceives a color in position 3, the transition enables the switch to behavior `REACT-TO-COLOR`. The parameter t of the behavior makes the robot approach the color source. At position 4, the white ground enables the transition to the second `REACT-TO-COLOR` behavior. The parameter t of this behavior makes the robot flee the color. The robot continues distancing the color source until perceiving the black floor at position 7, which enables the transition to the `EXPLORATION` behavior.

Even in this case, t decides which of the aforementioned strategies to employ, and p controls the probability of transitioning when the condition is verified.

AutoMoDe-Lahmacun internally uses IRace for the design of the finite state machines (Birattari et al., 2002). This selects and combines the modules to use, as well as their parameters (Birattari et al., 2021). The result is a probabilistic finite state machine designed for a specific environment and mission (see Figure 6.1). However, with AutoMoDe-Lahmacun, we want to produce finite state machines for just partially known environments. We approach this goal by presenting the robot with multiple instances of the environment and requiring the design of a finite state machine that performs well in all of them (Garzón Ramos et al., 2025). For this, we use a new setting of IRace that permits to evaluate the controller on multiple conditions[5].

[5]See `blockSize` parameter at `https://mlopez-ibanez.github.io/irace/news/index.html#irace-40`.

Specifically, it produces a ranking in each of the different instances and selects the solution with the best overall rank. This approach is very robust and permits presenting instances of the environment that grant different maximum performances.

AutoMoDe-Lahmacun presents a novel approach to the offline design of control software for robots. Aware of the flexibility that online adaptation provides, we allow the design process to select some parameters to be tuned online. We call these parameters *dynamic*, as their value changes at runtime. Currently, the offline design even selects the set of values that each dynamic parameter can assume at runtime. This is done in order to simplify and stabilize the adaptation, but it can be easily extended to use ranges of values. Online adaptation selects the optimal value for a parameter according to its performance in a specific instance of the environment. Assuming that the best parameter does not change over time, we can use a Multi-Armed Bandit algorithm for its selection (Slivkins, 2019). Specifically, we select the algorithm UCB1, proposed by Auer et al. (2002).

6.2 Experimental setting

We test our proposed approach on a task only partially known. Specifically, we clarify the goal of the mission, but we present the robot with conflicting training instances (i.e., in which supportive signals differ). The design method should produce a finite state machine that performs the best in the given cases. The desired outcome is a robot that adapts to the specific instance of the environment.

In order to provide a clear example, we devise a task illustrating the idea at its core. The task requires the robot to navigate an environment and to interact with some colored entities (see Figure 6.2). Some entities provide the robot with a prize, others waste its time. The color of the prizing entities is decided at the beginning of the experiment. The difficulty resides in deciding which entities to approach and which to ignore. This task represents a type of choice common in many contexts: doing or not doing. For instance, a robot acting in a wild and unknown environment might learn to avoid possibly dangerous situations. Differently, a robot supervising a factory might have to understand when to try solving a new problem by itself or instead call for help based on its experience.

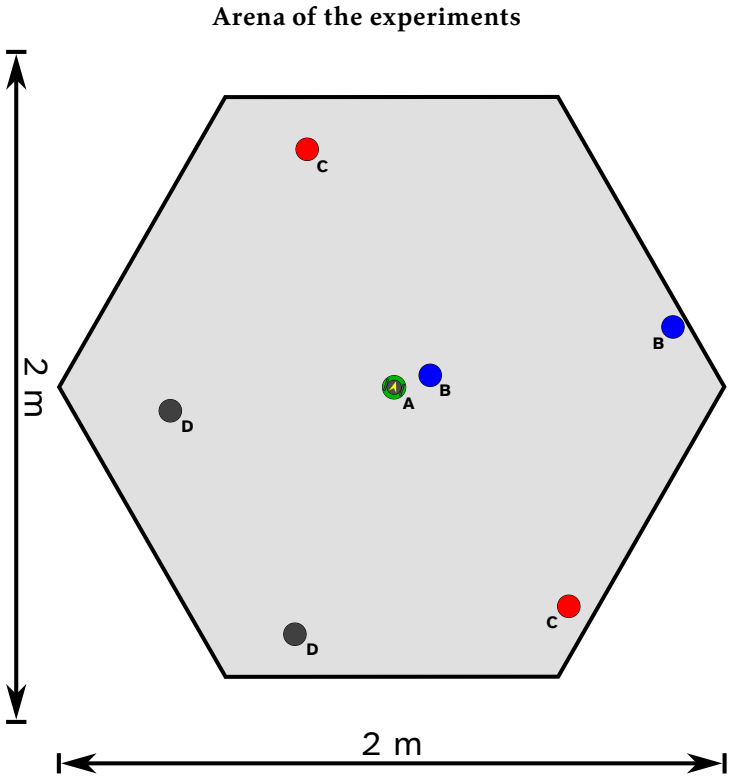


Figure 6.2: Representation of one instance of the arena of the experiment. In addition to the robot (A), it contains three different types of colored entities: blue (B), red (C), and black (D). Just one among the three colored entities prizes the robot upon interaction.

As previously stated, online adaptation requires the robot to know its performance at runtime. For the selected task, the robot considers the prizes bestowed by the colored entities. The communication of a prize happens by means of the RAB. The more prizes received in a specific amount of time, decided during design, the better the behavior. As we do not consider penalties, this makes the performance of the robot monotonically increasing in time. Mathematically, we can describe the performance function as follows:

$$P_{epoch} = \sum_{i=1}^{k} P^{i}_{step} \tag{6.1}$$

$$P_{step} = \begin{cases} 1 & if\ prize \in M \\ 0 & otherwise \end{cases} \tag{6.2}$$

where:

- k is the number of steps in an epoch;
- i is the step under consideration;
- P_{epoch} is the performance at a given epoch;
- P_{step} is the performance at a given step;
- M is the set of unread messages received through the RAB.

To statistically evaluate our approach, we start multiple design processes and test the produced controllers. Each design process presents the candidate controllers with multiple arenas characterized by the color of the prizing entities: in these experiments, *blue* or *red*.

In order to assess the utility of online adaptation, we run the design process for both *static* (i.e., non-adaptable) and *dynamic* (i.e., adaptable) controllers. Specifically, for the latter, we permit the selection of just one dynamic parameter with two possible values. This limitation is both to assess the impact of this minimum change and to reduce the search space of possible solutions. Indeed, permitting multiple dynamic parameters and values increases the search space of possible controllers.

In order to assess the range of achievable performances, we additionally design *static* controllers for each of the two arenas. This means that we produce controllers that do not aim to generalize, but just try to overfit to their presented environment. This gives us an indication of what a controller could achieve in a specific known situation. Conversely, testing these controllers on the arena they were not designed for permits to obtain an indication of a sort of lower bound of performance. We refer to controllers generated for the blue arena as *blue oracles*, and to those generated for the red arena as *red oracles*.

Each controller can be composed of a maximum of 4 behaviors, each with a maximum of 4 output transitions. For each experiment configuration (i.e., *static*, *dynamic*, *blue oracles*, and *red oracles*), we run 30 design processes. We select (randomly, one of) the best controller(s) produced by each process, and run it in each of the two considered arenas for 2500 s. This way, we obtain a distribution of 30 performance trends that permit to compare the four different experiment configurations in a given arena.

6.3 Results

As described in the previous section, we obtain 30 performance trends for *static*, *dynamic*, *blue oracle*, and *red oracle* controllers in each of the two arenas. We compare the results in order to assess whether combining offline design and online adaptation has any advantage with respect to designing controllers only offline.

The first analysis we perform aims at comparing the distribution of the *final* performance of *static* and *dynamic* controllers, and how they perform with respect to the lower and upper bounds attainable by using AutoMoDe. To do so, we first test the *blue oracles* and the *red oracles* in the arena they have been designed for, so as to obtain the upper bound of performance. We refer to this distribution as *oracle*. Then, we test them again on the arena they have not been designed for, so as to calculate the lower bound of performance. We refer to this distribution as *anti-oracle*. Finally, we obtain the *static* and *dynamic* distributions by testing their controllers in the two arenas. The results show that both *static* and *dynamic* final performances are higher than those of *anti-oracles*, and lower than those of *oracles* (see Figure 6.3). Moreover, between the *static* and *dynamic* distributions, the former contains the highest score(s), while the latter contains averagely better performances. Indeed, only about the best 25% *static* controllers can compare with the best 75% *dynamic* ones.

Nevertheless, this distinction in the shape of the distributions hides the performance in single arenas, that might explain the data differently. Therefore, we additionally consider the performance of the controllers in each of the two arenas separately (see Figure 6.4). The results show that the relative similarity of the *static* and *dynamic* distributions depends only on the specialization of *static* controllers. Indeed, most of them succeed in obtaining good performance in one arena at the expense of their effectiveness in the other. Controllers trying to balance their performance on both perform instead statistically worse than their *dynamic* counterpart. Finally, as expected, the *blue* and *red oracles* fully specialize in the arena they have been trained on, obtaining poor results in the other.

The result in the two arenas clearly shows that the initial similarity in the distributions is due to "specialization" of *static* controllers. In order to analyze the performance distribution more fairly, we compare the *average* performance of *static*, *dynamic*, *blue oracle*, and *red oracle*

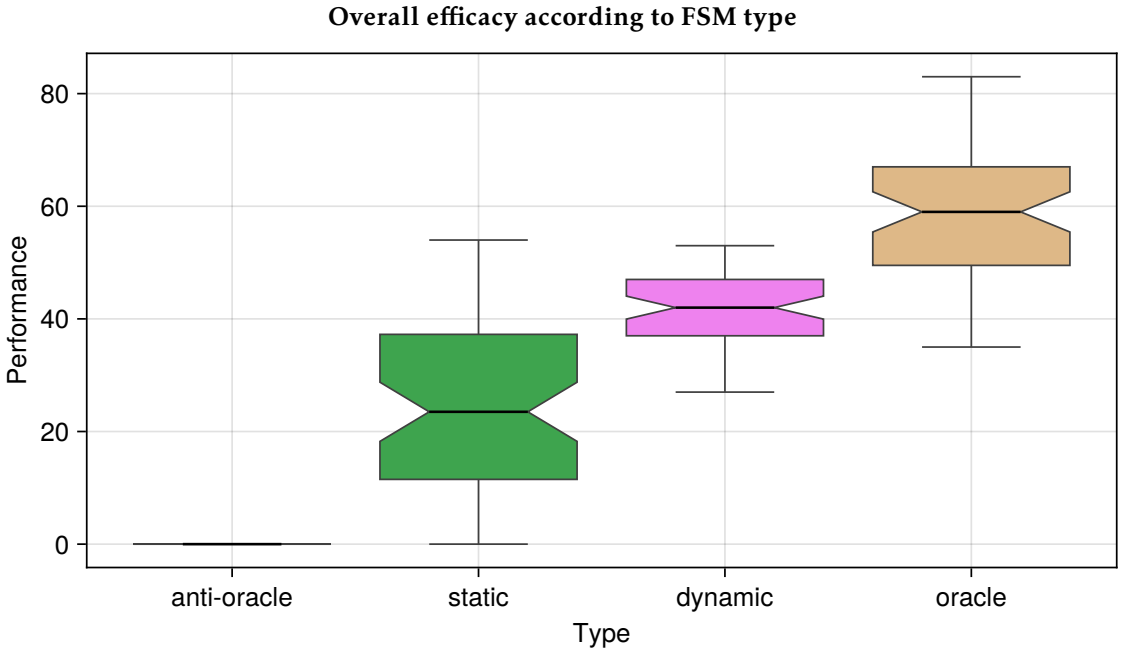


Figure 6.3: Overall performance of different types of finite state machines. Static and dynamic finite state machines are evaluated in both arenas of the experiment. Oracles represent the maximum performance we can expect to achieve using AutoMoDe in a specific arena. Finally, anti-oracles represent the minimum performance we can expect to attain.

controllers on the two arenas (see Figure 6.5). This means testing every controller in both arenas and then performing the average of the two resulting performances. The results show that the average performance attained by *dynamic* controllers is much higher than all the others. Notably, it completely dominates the *static* one. Interestingly, the *static* distribution is even lower than the *blue oracle* and *red oracle* ones, indicating that *on average* designing a *static* controller for two arenas is worse than specializing.

Another interesting aspect to analyze is the performance over time. Due to the intrinsic characteristics of online adaptation, we expect the performance to improve slightly as the experiment advances. This is because the robot initially wastes time trying to understand the best state in which to operate, exploiting this gained advantage only later on. The results confirm this expectation, with the average performance trend of *dynamic* controllers being similar to that of *static* ones in the first 10% of the experiment (i.e., about 4 minutes), and increasing in slope from there on (see Figure 6.6). We can clearly see that, as the time advances, the gain of *dynamic* controllers with respect to *static* ones increases. Nevertheless, the slope of *dynamic*

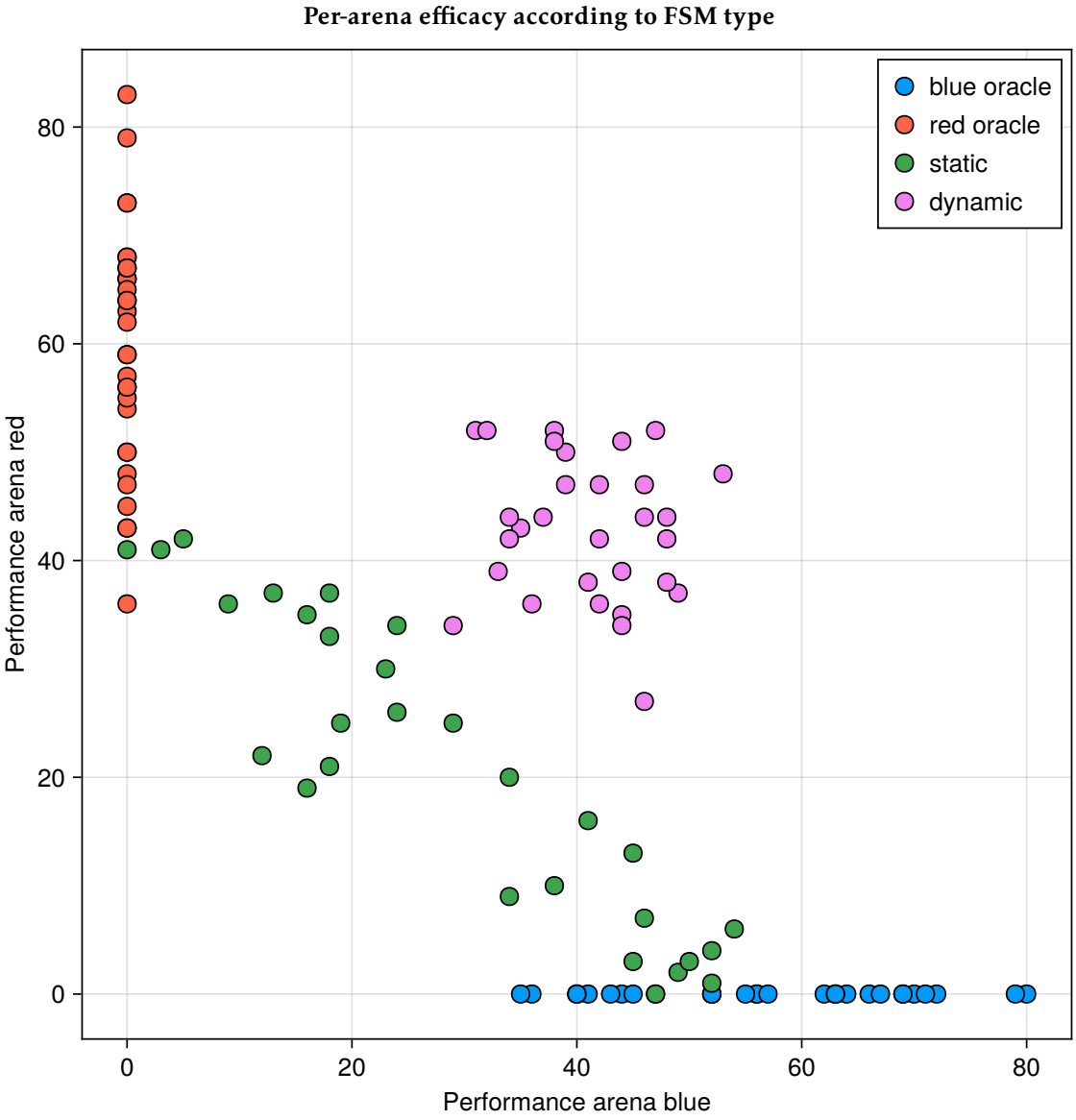


Figure 6.4: Efficacy of different types of finite state machines in a specific arena. Since oracles are normally evaluated on the arena they have been trained for, when tested on the other arena, they attain the performance of an anti-oracle, usually around 0.

controllers seems still less than that of the *oracles*. This might be due to the need for the online adaptation to periodically assess the dynamic values in search of a change in their effectiveness.

Finally, we explore the complexity of the produced solutions. We do so by analyzing the number of behaviors and transitions in each controller (see Table 6.3). We find that, on average, *dynamic* finite state machines are simpler than *static* and *oracle* ones. This is because adapting

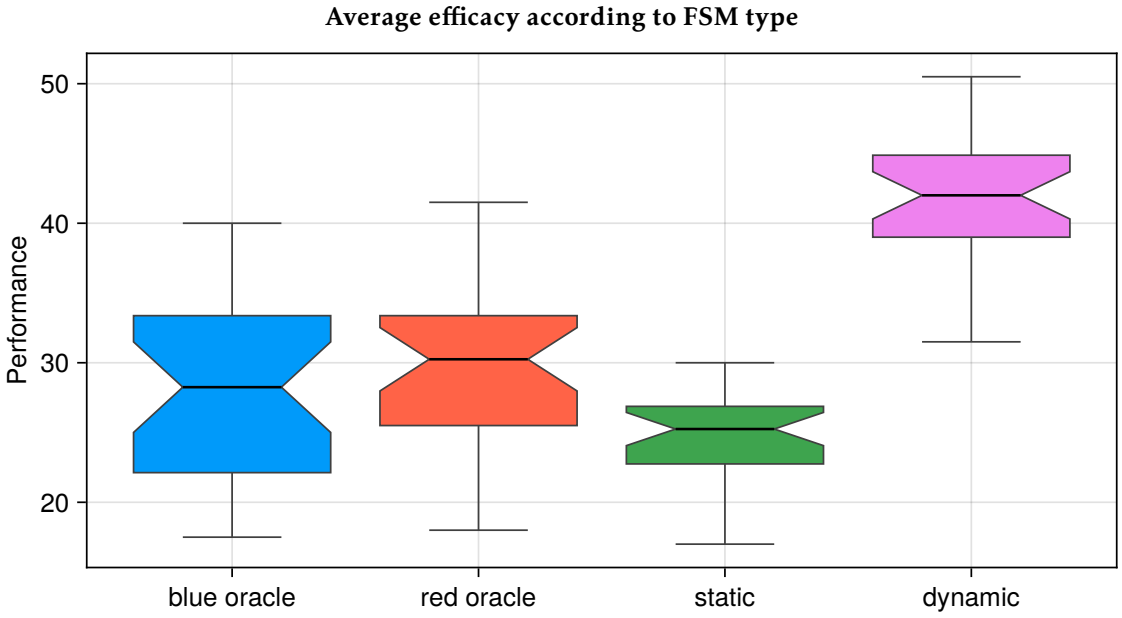


Figure 6.5: Average performance of different types of finite state machines on the two arenas. The distribution of *blue oracle*, *red oracle*, and *static* is low due to specialization of the controllers on a single arena. Differently, the *dynamic* distribution is high because its controllers adapt to the specific scenario.

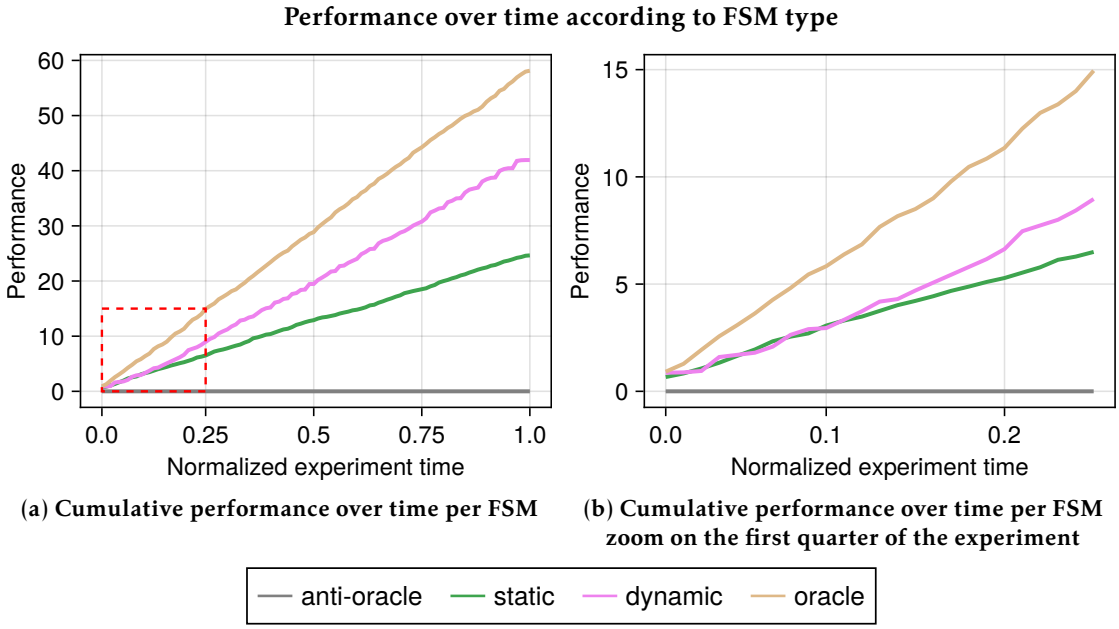
Controller	Average behaviors count	Average transitions count
dynamic	2.27	4.00
static	3.97	9.77
blue oracle	3.87	9.50
red oracle	3.94	10.13

Table 6.3: Average number of behaviors and transitions per finite state machine, according to the type of controller.

a key parameter permits exploiting similar strategies in the two arenas, varying only what is needed. Obviously, part of this simplification is due to the transfer of complexity from the finite state machine to the adaptive algorithm, but this still remains an interesting result.

6.4 Discussion

In this chapter, we investigate whether combining offline design and online adaptation is a valid strategy to improve the capabilities of robots. We find that the produced controllers are



(a) Cumulative performance over time per FSM

(b) Cumulative performance over time per FSM zoom on the first quarter of the experiment

Figure 6.6: Cumulative performance over time for different types of finite state machines. On the left, the trend throughout the whole experiment. On the right, a zoom of the trend in the first 25% of the experiment. The performance of *dynamic* finite state machines starts with a soft slope that increases with time, gaining advantage over *static* finite state machines after about 10% of the experiment length (i.e., about 4 minutes). This is due to the time required to discover the best dynamic value.

indeed more flexible, permitting to obtain more stable performance in different environments. The improvement is due to the ability to modify the key aspects of the controller automatically, adapting to the peculiarities of the surroundings. Specifically, we notice that even a single parameter can be enough to start behaving effectively in simple contexts. We expect, however, that more complex situations will require more dynamic parameters, so as to increase the variability of the behavior. By itself, this is not a problem, as the system can easily select more dynamic parameters. Nevertheless, we need to consider the size of the search space we will have to explore. Increasing the number of parameters enlarges the search space notably, possibly becoming a problem quickly. On the other hand, *dynamic* controllers are averagely simpler, possibly permitting to force searching in a reduced solution space. We will perform further investigations in this regard in the future.

The system we propose is effective for the class of tasks we consider. Indeed, it permits the robot to perform well in each of the conditions it might encounter, while maintaining a rel-

atively simple control logic. Nevertheless, we believe it is important to discuss its application scenarios more clearly. We consider a user desiring the automatic generation of control software for a partially known environment. Specifically, it wants the robot to display good performance in each possible situation. Let's imagine now a different user not interested in obtaining a good performance in any situation, but instead in obtaining the highest *average* performance. According to the probability of occurrence of each instance of the environment, this user might still opt for a static controller. Indeed, it could decide to maximize the performance for the most frequent instance of the environment, in order to increase the average one. In other words, it could decide to use one of the static control software that we call *oracles*, which are specialized for a specific environmental instance. The system we propose is obviously not designed to take these types of decisions, which instead depend on the user and on their knowledge of the possible environment the robot can face. Instead, it aims to generate a solution that performs well in most of the possible cases, without assumptions on their frequencies.

Another consideration is that the tool we propose currently considers only partially-known static environments. This means that some characteristics of the working scenario are unknown at design time, but remain fixed during the entire run. This choice comes from the goal of this investigation: exploring the controlled combination of offline design and online adaptation of robot controllers. Indeed, our aim is mainly to present the potential of this approach rather than a production-ready system. This led us to select a clear parameter-selection algorithm based on multi-armed bandits. Alternatives considering time-varying environments are not only possible, but also very easy to implement. An example is to keep using a multi-armed bandit with a buffer, such that old scores are continuously removed in favor of new ones. This would implicitly modify the distribution of the rewards according to changes in the environment, potentially leading to a shift in the preferred parameter and thus making the robot adaptable to time variations.

In Chapter 4, we discussed issues related to the time to adapt; mainly, the robot should be able to adapt its control system faster than the frequency of changes in the environment. The results of this chapter present an interesting example supporting this previous consideration; in our investigation, the adaptive finite state machine starts to gain an advantage against *static* controllers about 4 minutes after the start of the experiment. Let's imagine that a robot able to

adapt to an environment changing in time requires more or less the same time. If the environment changes periodically in less than 4 minutes, the robot might have trouble achieving a good average performance, potentially performing worse than *static* controllers. Therefore, the frequency of changes in an environment remains a factor to consider when designing an adaptive robot.

In Chapter 3, we proposed online adaptation as a strategy to overcome faults. Combining it with offline design could be a way to increase the effectiveness of the robot in these situations. The design tool could present different training instances in which the robot suffers different types of damage, selecting the most tolerant adaptive controller so as to overcome them.

In this chapter, we investigate the combination of offline design and online adaptation. Specifically, we build a tool for the automatic generation of adaptable control software. This produces controllers for a specific task, for which, however, some information is unknown at design time. The results show that the produced controllers are much more effective than controllers not able to adapt, supporting the inclusion of this type of adaptability in offline-designed control software. Additionally, this approach mitigates the adaptation time issue identified in the previous chapters, permitting the robot to adapt in a few minutes only.

6.5 Chapter highlights

- Combining offline design and online adaptation makes the robot behavior more reliable, flexible, and fast to adapt at the expense of variability.
- Combining offline design and online adaptation is reasonable when we have partial information about the environment.
- Combining offline design and online adaptation might not always be desirable, depending on what we want to obtain (e.g., averagely-good versus overall-good performance).
- The search space size of adaptable controllers increases with the number of dynamic parameters, and thus with the unknowns of the environment.
- The combination of offline design and online adaptation is applicable also to time-varying environments, supposing that the initially assumed conditions do not change.

Chapter bibliography

Auer, P., N. Cesa-Bianchi, and P. Fischer (2002). "Finite-time Analysis of the Multiarmed Bandit Problem." In: *Machine Learning* 47.2, pp. 235–256.

Baldini, P., A. Roli, and M. Birattari (2026a). "Compensating Design-Time Uncertainty Through Combined Offline Design And Online Adaptation of Robot Control Software." In: *Autonomous Robots*. To be submitted.

Birattari, M., A. Ligot, and G. Francesca (2021). "AutoMoDe: A Modular Approach to the Automatic Off-Line Design and Fine-Tuning of Control Software for Robot Swarms." In: *Automated Design of Machine Learning and Search Algorithms*. Springer International Publishing, pp. 73–90.

Birattari, M. et al. (2002). "A Racing Algorithm for Configuring Metaheuristics." In: *GECCO-2002: Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*. Vol. 2. Morgan Kaufmann Publishers, pp. 11–18.

Codling, E.A., M.J. Plank, and S. Benhamou (2008). "Random walk models in biology." In: *Journal of The Royal Society Interface* 5.25, pp. 813–834.

Feola, L. and V. Trianni (2022). "Adaptive Strategies for Team Formation in Minimalist Robot Swarms." In: *IEEE Robotics and Automation Letters* 7.2, pp. 4079–4085.

Garzón Ramos, D. and M. Birattari (2020). "Automatic Design of Collective Behaviors for Robots that Can Display and Perceive Colors." In: *Applied Sciences* 10.13, p. 4654.

Garzón Ramos, D. et al. (2025). "Automatic Design of Robot Swarms under Concurrent Design Criteria: A Study Based on Iterated F-Race." In: *Advanced Intelligent Systems* 7.1, p. 2400332.

Hasselmann, K. et al. (2018b). *Reference models for AutoMoDe*. Tech. rep. TR/IRIDIA/2018-002. IRIDIA, Université Libre de Bruxelles, Belgium.

Kegeleirs, M. et al. (2024). "Transferability in the Automatic Off-Line Design of Robot Swarms: From Sim-to-Real to Embodiment and Design-Method Transfer Across Different Platforms." In: *IEEE Robotics and Automation Letters* 9.3, pp. 2758–2765.

Mondada, F. et al. (2009). "The e-puck, a Robot Designed for Education in Engineering." In: *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*. Vol. 1. 1. IPCB: Instituto Politécnico de Castelo Branco, pp. 59–65.

Slivkins, A. (2019). "Introduction to Multi-Armed Bandits." In: *Foundations and Trends® in Machine Learning* 12.1–2, pp. 1–286.

Part III

Final remarks

7 Synopsis of adaptive controllers

In this chapter, we aim to summarize the characteristics of the adaptive controllers proposed and employed in this dissertation. We begin by describing once more the control architectures employed, which leverage: network systems, artificial neural networks[1], and finite state machines. We also detail the adaptive mechanisms operating on these architectures, which include recoupling and reweighting of sensors and actuators, random weights modification, and multi-armed bandit based parameter selection. Finally, we conclude with a discussion on general adaptation parameters that govern the process in every situation: re-evaluation and discounting, adaptation intensity, adaptation probability, evaluation time, and evaluation functions; their ability to operate with different control architectures is evidence of their generality.

7.1 Network controllers

The first types of controller that we employed leverage Boolean and nanowire networks for computation (see Chapters 2 and 3). These two dynamical systems are notably different in their functioning, but they also present some similarities. Notably, they can be conceptualized as sets of interconnected nodes whose state updates according to that of other nodes and to potential external influence. Consequently, these systems do not possess predefined inputs or outputs;

[1] Obviously, also artificial neural networks are "network systems". Here, we use these two terms for simplicity, so as to distinguish Boolean and nanowire networks from artificial neural networks.

instead, we can connect to any point of the network to either perturb or read its local state. In order to exploit these media for computation, we therefore employed a simple strategy consisting in coupling sensorial inputs and actuation outputs to network nodes. The inputs perturb the state of the network, modifying its dynamics and causing changes in other nodes, some of which will be used as outputs. Nevertheless, since we do not modify the network structure, the control challenge is restricted to identifying the optimal connection points to elicit the desired behavioral output. A consequence of this constraint is that, if the inherent complexity of a given system is insufficient, the search for a viable control solution may be unsuccessful. Conversely, this approach enables using even unmodifiable control systems for computation, similarly to what happens in reservoir computing or evolution in materio.

The use of unconventional and unmodifiable control systems may require the conversion of inputs and outputs between the sensor/actuator and the network operational domains. For instance, Boolean networks operate in the binary domain, necessitating the conversion of analog and digital values. This could involve feeding the network with the digitalized value representation on multiple nodes, over time, or by applying a binarization threshold. Furthermore, some systems sensitivity to input perturbations might be limited; it is the case, for instance, of nanowire networks, which might not produce output if the input is either too weak or incorrectly polarized. For this sort of situation, we propose the possible negation and weighting of the inputs (and potentially of the outputs), consisting in their multiplication by a custom weight.

Adaptation in network systems is a rather simple process: it selects some connections, and it reweights or recouples them to other nodes of the network. Nevertheless, in order to avoid overriding values and trivializing the control problem, we advise recoupling onto free nodes that are not in proximity of output nodes; the goal is to force the system to process the inputs without outputting them directly or with negligible changes. Furthermore, during reweighting, it is important to keep in mind the operational domain of the network system, so as to avoid saturating it with excessively multiplied values.

7.2 Neural controllers

In our investigations, we also employed artificial neural networks to control robots (see Chapters 4 and 5). These systems are classical models in artificial intelligence that can approximate, in principle, any continuous function. This means that, given some inputs, they will compute the corresponding output according to their model. In our case, we use two specific instances of networks: a simple recurrent neural network and a perceptron. We do not perform traditional training, but rather modify their values randomly in a process akin to the evolution of neural networks in evolutionary computation. Also in this case, we suggest limiting the range of possible values that the weights can assume; the rationale for this constraint is that, in the event of notable environmental changes, the system must be able to return to a neutral state in order to enable further adaptations. Without this precaution, the risk is that the network will require a long time before converging to an effective state, if it succeeds at all.

7.3 Finite state machines controllers

The last type of controller that we employed in our investigations is finite state machines (see Chapter 6). These are coherent organizations of basic behaviors that produce the desired, more complex, robot behavior. Despite the possibility of adapting finite state machines in their entirety, we limited the scope of adaptation to the modification of their parameters. Indeed, we design the controller offline, leaving only selected parameters free to adapt; those are the ones producing the most effective range of behaviors for the target goal. The result is that the adaptation process is very rapid and safe, but less flexible.

The adaptive process employed for finite state machines leverages a multi-armed bandit algorithm, which selects the parameter values that produce the behavior more statistically effective. The only disadvantage of our implementation is that the performance distribution resulting from using a specific parameter value must be fixed. This is usually possible only when considering static environments. Nevertheless, we propose a variation of this algorithm simply employing a limited buffer to enable forgetting old and possibly no longer valid evaluations. The expected result is that the system will become capable of adapting also in dynamic environ-

ments, and not only in partially known, static ones.

7.4 Adaptation parameters

All adaptive processes discussed in the previous sections share some common high-level approaches and parameters. Among the most important in tackling dynamic scenarios is the need to update the evaluation of already tested controllers. Indeed, this permits forgetting old solutions once they stop being effective, or if their performance depended on a lucky evaluation only (we call this problem "situationality"). The absence of this mechanism would compel the system to persist in utilizing a temporary effective behavior (or its variations) across all subsequent, potentially diverse scenarios, without the possibility of forgetting it. Therefore, updating the evaluations is of primary importance. Here, we propose to perform a weighted average of the new and old performances, or to discount the latter by multiplication with a value lower than 1. We avoid directly overwriting the performance in order to make the system more resistant to unlucky evaluations.

The adaptation intensity is also among the most impactful parameters: it controls how different a new controller can be with respect to its original version. Since we imagine similar controllers to behave similarly, this parameter effectively controls how innovative a new behavior can be. We find that the best use of adaptation intensity involves its modulation according to the performance, so as to produce more varied controllers when the performance is low, and more uniform ones when it is high. The goal is to modulate the difference in the resulting behavior according to the performance. From a practical perspective, the adaptation intensity can modulate various aspects according to the control mechanism employed; for instance, it can control the number of reconnected couplings or the standard deviation of a parameter change when performing reweighting.

Another important parameter is the adaptation probability, that controls the probability of performing an adaptation step rather than using the best behavior known. Practically, it balances exploration and exploitation of the controllers. Also in this case, we find that the best use of this parameter involves its modulation according to the performance. The obvious motivation is to maintain the performance of the robot high, employing well-behaving controllers

when they are known and otherwise exploring new solutions. Additionally, this permits co-adaptation, enabling weaker individuals to adapt more often so as to learn to behave in synergy with stronger ones.

Another aspect to select carefully when employing online adaptation is the duration of the controller evaluation. On the one hand, this can lead to erroneous evaluations if time is insufficient. On the other hand, excessively long evaluations delay the test of new controllers. In this dissertation, we proposed to select this parameter offline according to simulated scenarios (see Chapter 6). However, there is a margin for improvement through the investigation of strategies that can better balance reliability and speed.

Finally, we highlight the importance of selecting the correct evaluation functions to drive the adaptive process (see Chapter 4). We believe that those should be as general as possible, so as to avoid biases induced by partial knowledge of the environment. In this regard, it could be useful to employ functions abstracting away from the task and using some general forms of evaluation. Nevertheless, this is an issue still to be solved, and for which we propose some possible solutions in Chapter 9.

8 Ancillary explorations

This chapter provides a concise summary of all the investigations conducted alongside the main research line of this dissertation. This includes works related to, but not aligned with, the focus of the present dissertation.

My first investigation in the field of robotics explored the use of reservoir computing, a framework that leverages the innate input separation and transformation properties of complex dynamical systems, referred to as *reservoir* (Baldini, 2022). This approach permits training a simple linear readout layer that merely discriminates different states of the reservoir, thereby substantially simplifying the learning process. Furthermore, the input processing of the reservoir can be exceptionally rapid, approaching the speed of light in optical implementations (Larger et al., 2017); this enables, in addition to the reduced training time, even to compute at speeds surpassing those achievable with conventional electronic hardware circuitry. Over the years, reservoir computing has consistently demonstrated significant computational capabilities and has been successfully employed across numerous tasks and fields. Consequently, the work investigates its specific utility within robotics, providing a comprehensive theoretical and empirical review. Furthermore, the paper proposes several application ideas and describes possible future developments.

Recently, we had the opportunity to investigate self-assembly capabilities in modular soft-robots named voxel (Baldini et al., 2025c). Self-assembly refers to the ability of individual units

to aggregate in suitable structures in order to tackle complex tasks that could not be solved individually. To achieve this capability, we evolved the artificial neural network controller of the robots offline with CMA-ES (N. Hansen et al., 2001), so as to induce the formation of a single, functional structure. This aggregated system must complete the task of traversing a hole in the terrain, which is too wide for the individual robots. Experiments conducted in increasingly complex simulated arenas demonstrate that the evolved robots are often capable of forming a connected chain and of completing their mission. Finally, we also discuss limitations and possible improvements of the proposed approach.

As part of robotics research, we believe it is also important to investigate the integration and the effects of novel technologies in the field. Recently, the advent of Large Language Models (LLMs) has revolutionized many aspects of software development, yet their specific impact on robotics has been largely overlooked. We claim that the peculiar characteristics of robotics programming, mainly the need to operate reliably in a noisy and unpredictable world, necessitate special attention. Therefore, in Baldini et al. (2025b) we begin investigating the consequences of using LLMs as assistants during the programming of robot control software. We analyze how they affect the performance of the robot in simulation and across a pseudo-reality gap; additionally, we consider the similarity of the produced controllers, the coding experience reported by the programmers, and the coding time. The results suggest that control software produced with the assistance of LLMs is less robust to unseen conditions and is, overall, more similar across different developers. Additionally, participants using LLMs reported feelings of frustration during development, but they produced functional code faster.

On the topic of Large Language Models, we performed further investigations involving their integration within cyberphysical systems. Specifically, in Braccini et al. (2026b), we explore the use of generative artificial intelligence in extended reality, with a focus on the effects that can emerge in case of errors or disturbances in the communication channel between the physical and virtual worlds. Indeed, in systems where LLMs mediate or generate responses, these processes can trigger novel and unexpected outputs, thereby representing affordances for creative interactions. We start to explore this scenario by analyzing the robustness of LLMs to perturbations in the form of scrambled character blocks of prompts, thus simulating a typical extended reality chat scenario where an LLM controls a virtual avatar. As a preliminary study, we analyze the re-

sponse behavior of two small LLMs using a limited sample size of 30 questions. We observe that the temperature parameter and the size of the LLM jointly influence the robustness to perturbations. Interestingly, in perturbed scenarios, a higher temperature increased the correctness of the LLMs rather than their creativity in the sense of producing unexpected content.

The work on the extended reality scenario led us to imagine a robotic teacher employing Large Language Models to output educational information; despite some concerns, we believe that it represents a likely application of this technology in the near future. Therefore, we decided to investigate the suitability of LLMs as teachers through the proxy of creativity (Braccini et al., 2026a). The question is whether these systems can replicate the creativity and nuances of human languages, providing students with the variability and richness of expression necessary for effective learning and for nurturing critical and divergent thinking. This is particularly important in the pre-school or primary education context, where novel and insightful content is crucial for effective engagement. Framing this problem from an abstract linguistic perspective, it translates into wondering how effectively the "creativity" of LLM-generated text can be quantified and how it compares to human one. So, recognizing the inherent ambiguity in defining creativity, we investigate the variability of LLM outputs as a potential proxy and compare it with observed text variability in a large human-authored dataset. Specifically focusing on early childhood education, we compare the semantic and syntactic diversity of responses from LLMs and humans prompted to answer like 5-year-olds. Our empirical results show that LLMs have lower variability than humans, particularly on repeated questions, though the gap narrows for different questions. This limitation raises concerns for educational suitability, where diverse explanations of concepts are crucial, thereby suggesting a need to enhance LLMs expressive range and diversity.

In Chapter 3, we employed robots controlled by Boolean networks. In a later work, we considered another use of these systems: the simulation of living tissues (Braccini et al., 2025). The aim of the study is to understand how the interaction between biological cells influences the emerging dynamic of a tissue; we perform such an investigation through the proxy of coupled Boolean networks. Specifically, we focus on attractor properties and responses to perturbations. We show that, irrespective of the dynamical regime of individual Boolean networks (i.e., ordered, critical, or chaotic), their collective behavior tends toward criticality. This result suggests that

tissues, commonly hypothesized to exist in a critical state, may in fact be composed of cells in different dynamic regimes, and that the collective dynamics may be modulated by cellular interaction. Currently, we imagine that similar results could be observed also in robot swarms, whereby self-organizing properties could emerge even from ordered (e.g., purely reactive) or chaotic (e.g., pseudo-random) individual behaviors.

Also in Shrestha et al. (2025) we investigate complex dynamics, but we do so in a modified "Game of Life" (i.e., a 2D cellular automaton). The key difference with the classical Game of Life definition resides in the added possibility of local mutations of core rules, enabling a heterogeneous life-like cellular automaton guided by an evolutionary inner loop. Furthermore, we introduce the concept of cell aging, permitting to observe death and replacement of cells by means of age. We employ this model to conduct an experimental campaign so as to identify suitable parameters that produce long-term dynamics and favor genotypic innovation. To summarize, this work explores heterogeneity induced by local interaction and external context, and studies which factors enable innovation. We believe that this sort of investigation, despite involving simple simulated scenarios, could prove propaedeutic to further advancements even in the study of online adaptation in robotics.

Chapter bibliography

Baldini, P. (2022). *Reservoir Computing in robotics: a review*. arXiv: 2206.11222.

Baldini, P., M. Braccini, and A. Roli (2025b). "Impact of LLM-Assisted Coding in Creativity and Robustness of Robot Controllers." In: *HAIC 2025 – Workshop on Human-AI Collaborative Systems 2025. Proceedings of the 1st Workshop on Human-AI Collaborative Systems, co-located with 28th European Conference on Artificial Intelligence (ECAI 2025)*. Vol. 4072. CEUR Workshop Proceedings, pp. 1–12.

Baldini, P. et al. (2025c). "Cooperation in Evolved Modular Soft Robots." In: *Artificial Life and Evolutionary Computation. 18th Italian Workshop, WIVACE 2024, Namur, Belgium, September 11–13, 2024, Revised Selected Papers*. Vol. 2532. Springer Nature Switzerland, pp. 160–172.

Braccini, M., G. Aguzzi, and P. Baldini (2026a). "Unraveling Creativity Through Variability: A Comparison of LLMs and Humans in an Educational Q&A Scenario." In: *Technology, Knowledge and Learning*.

Braccini, M., P. Baldini, and A. Roli (2025). "Cell–Cell Interactions: How Coupled Boolean Networks Tend to Criticality." In: *Artificial Life* 31.1, pp. 68–80.

Braccini, M. et al. (2026b). "On the LLM Robustness in a Simulated Conversational XR Scenario: A Preliminary Semantic Analysis." In: *Image Analysis and Processing - ICIAP 2025 Workshops. 23rd International Conference, Rome, Italy, September 15–19, 2025, Proceedings, Part II*. Vol. 16170. Springer Nature Switzerland, pp. 637–648.

Hansen, N. and A. Ostermeier (2001). "Completely Derandomized Self-Adaptation in Evolution Strategies." In: *Evolutionary Computation* 9.2, pp. 159–195.

Larger, L. et al. (2017). "High-Speed Photonic Reservoir Computing Using a Time-Delay-Based Architecture: Million Words per Second Classification." In: *Phys. Rev. X* 7.1, p. 011015.

Shrestha, A. et al. (2025). "Emergent Dynamics in Heterogeneous Life-Like Cellular Automata." In: *Artificial Life and Evolutionary Computation. 18th Italian Workshop, WIVACE 2024, Namur, Belgium, September 11–13, 2024, Revised Selected Papers*. Vol. 2532. Springer Nature Switzerland, pp. 1–15.

9 Research directions

All the investigations presented in this dissertation act as a foundation for further explorations. This chapter presents an overview of ongoing and possible future works involving online adaptation of robots.

Current online adaptation mechanisms in robotics use evaluation functions to drive the changes. Nevertheless, devising evaluation functions is a duty still reserved to human designers, which could be biased by previous knowledge and thus implicitly constrain the formulation. Furthermore, it is sometimes difficult to express goals in function of the robot perception. We propose the automatic generation of evaluation functions as a strategy to overcome these issues. A similar approach has been recently used in inverse reinforcement learning, where the reward function is inferred from demonstrations (Arora et al., 2021); however, this reward function can be omniscient (i.e., it can have more information than that available to the robot), and it is thus useful only during training. What we propose is instead to generate a function employing the robot perception and action only, enabling it to be used online to drive the adaptation.

An alternative approach to the generation of evaluation functions could attempt to employ complex computational systems; here, we propose to assess Large Language Models, which could synthesize them based on natural language specifications. Undertaking this line of investigation would also require validation in complex missions and scenarios, so as to evaluate advanced reasoning capabilities and exclude that the generation relies solely on raw training

data (i.e., that it just "remembers" an already seen function). Nevertheless, we retain a degree of skepticism regarding the ultimate efficacy of generating evaluation functions from such systems.

The automatic generation of evaluation functions will likely necessitate a phase of design or test offline. During this process, it could also be possible to train the controller of the robot, thereby combining offline design and online adaptation as proposed in Chapter 6. Nevertheless, if the operational environment is largely unknown, it could be difficult to train a controller directly. A viable alternative is to draw inspiration from meta-learning to devise a controller able to learn online faster (Vettoruzzo et al., 2024). This approach would offer the advantage of pre-tuning the controller while generating the evaluation function. We hypothesize that it could speed up the adaptation (thanks to meta-learning) and make it less subject to human biases (thanks to the automatic design of the evaluation function).

One of the challenges in online adaptation is the substantial proportion of ineffective controllers tested during runtime. In this dissertation, we address this issue by modulating the variability of newly created controllers (i.e., the size of the search space centered on the current best controller) according to the robot performance (see Chapters 4 and 5). Nevertheless, this approach risks blocking if the sampling process operates in the surroundings of an isolated peak of the solution landscape. This scenario would prevent the discovery of potentially superior peaks except by random chance, thereby trapping the search process within a locally ineffective region of the search space. In such a situation, the capacity to evaluate highly dissimilar controllers (i.e., to search in a wider solution space) would be beneficial. Nevertheless, a frequent wide search is not sustainable online because of the limited test time and the need to maintain a reasonable performance; therefore, it should be permitted but reduced to the bare minimum. Here, we propose to train a network that, given perception and action, outputs a guess on the resulting perception at the next computational step, in a process similar to that proposed in Ha et al. (2018b) and Ha et al. (2018a). This could be used to iteratively simulate multiple steps so as to assess the controller effectiveness in the task. The idea of this abstract strategy is depicted in Algorithm 1. The risk is that this approach could not work in very dynamic contexts, but, in this case, it could be possible to keep training the simulation network runtime to make reliable guesses in the new conditions.

Algorithm 1 Pseudocode of controller simulation.

```
n ← param('n')          ▷ set a parameter n indicating the maximum number of iterations
ρ ← 0                   ▷ create a variable accumulating the performance
c ← new_controller      ▷ generate a new controller
p ← perception          ▷ get the current robot perception
repeat
    a ← c(p)            ▷ select the action according to perception
    ρ ← ρ + evaluate(p, a)   ▷ add the current evaluation to the cumulative performance
    p ← simulate(p, a)       ▷ simulate the next state according to perception and action
    n ← n − 1           ▷ update the iteration counter
until n ≤ 0             ▷ repeat for n iterations
```

Online adaptation permits the discovery of effective behaviors by trying them on the robot itself; nevertheless, this approach brings with it some difficulties. Mainly, while testing unknown control strategies, the robot is subject to risky situations. For instance, a poor controller could ignore a precipice and lead the robot to a catastrophic fall, potentially damaging or destroying itself. This type of situation is obviously to be avoided, but, to the best of the author's knowledge, the literature currently lacks ways to prevent it. The motivation is probably that the research on online adaptation only considered "what can be achieved" rather than the associated risks. Therefore, we propose to endow the robot with some safety modules representing innate reflexes that could prevent catastrophic damage. An example is indeed that of a robot moving towards a precipice and blocking just before falling. Alternatively, they could prevent crashing at high speed against obstacles, just permitting graceful pushes. This addition, despite its simplicity, should enable a broader applicability of online adaptation by making adaptive robots more reliable and safe.

Besides improving the safety of adaptation, we believe it is also important to enhance the adaptation process itself; specifically, the decision on whether to adapt or not. For instance, if the performance is already sufficiently good, it may not be useful to search for alternative controllers, or at least, it could not be useful in the short term. We anticipated this aspect in Chapter 5 by introducing a probability to adapt depending on the performance, but the plan is to improve the strategy by deciding to adapt when the performance deviates significantly from the desired or expected one. Also, it would be interesting to improve the adaptation phase itself, for instance by employing heuristics to decide which parameters to modify. An exam-

ple could be assigning priorities to the reconnection of sensorial perception in network-based control systems, or to modify specific weights in an artificial neural network.

Another possible investigation involves the use of online adaptation in swarms of robots when the performance cannot be easily inferred locally. Indeed, as introduced in Chapter 5, we could use online adaptation because minimizing and maximizing the number of neighbors automatically supported the emergence of the desired collective behavior. Nevertheless, in a mission such as, for instance, foraging, where task specialization can improve the work, this approach becomes more difficult. Therefore, we propose the creation of a communication system to permit each robot to infer the performance of the swarm (e.g., by sharing the number of delivered prey in foraging). This should enable using online adaptation also to address more complex swarm robotics tasks.

Chapter bibliography

Arora, S. and P. Doshi (2021). "A survey of inverse reinforcement learning: Challenges, methods and progress." In: *Artificial Intelligence* 297, p. 103500.

Ha, D. and J. Schmidhuber (2018a). "Recurrent World Models Facilitate Policy Evolution." In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., pp. 2451–2463.

— (2018b). *World Models*. arXiv: 1803.10122.

Vettoruzzo, A. et al. (2024). "Advances and Challenges in Meta-Learning: A Technical Review." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.7, pp. 4763–4779.

Conclusion

The primary objective of this dissertation is to enhance the understanding of the fundamental factors enabling online adaptation of robotic systems. This aims to fill a gap in the literature, whereby methodologies for the embedded modification of control software are evaluated experimentally with little attention paid to the elements crucial to their success. Furthermore, this aims to overcome an issue of classic investigations: the lack of evaluations in changing conditions; indeed, most investigations in the literature are limited to robots immersed in static environments, which do not permit a full verification of the systems capabilities. We address these problems by experimenting in changing environments with minimal control systems, which enable the investigation of these factors in isolation while avoiding more complicated controllers to obscure them due to pure raw computational power. Furthermore, this approach implicitly introduces solutions suitable for robots with limited computational capabilities, which might not be able to run state-of-the-art mechanisms.

The first experimental chapter of this dissertation (see Chapter 2) begins investigating the conditions under which online adaptation is possible (see **RQ1**) and discusses ways to assess the proposed mechanisms (see **RQ2**). There, we propose a control system and corresponding adaptive mechanism operating on nanowire networks, a novel miniaturized hardware dynamical system, cheap to produce and efficient to operate, suitable for implementing minimally cognitive robots. The investigation demonstrates that this system is suitable for learning advanced tasks in static conditions and that the nanowire network is a valid alternative to digital devices for controlling miniaturized robots. Alongside these results, this preliminary investigation enables us to understand how to properly analyze the performance of online adapting robots and

the associated challenges. We conclude that considering the trend of performance over time is the most reasonable way to assess them.

In the following chapter (see Chapter 3), we begin investigating a core application of online adaptation: overcoming faults, which represents a fundamental instance of changing contexts. This investigation permits to further expand our understanding of the operational requirements and limits of online adaptation (see **RQ1**) and to verify whether it permits recovering the performance in case of internal changes (i.e., occurring inside the robot itself; see **RQ4**). We test the adaptive mechanism in different tasks, identifying that re-evaluating the controller is a necessary characteristic to address changing conditions. Furthermore, we confirm that online adaptation enables the performance recovery after faults, up to a certain limit depending on the intensity of the damage and contingent on the robot ability to obtain a reliable indication of its performance. Specifically, this latter constraint represents an important limitation in adaptive robotics, for which we do not see many realistic solutions.

Chapter 4 investigates the dual problem: addressing changes in the external environment, expanding our knowledge on the effectiveness of online adaptation in changing conditions (see **RQ4**) and about contextual requirements and difficulties (see **RQ1**). This investigation confirms the previously identified need to re-evaluate the best solution, and it additionally shows that modulating the adaptation intensity is a viable strategy to control the search for novel solutions. Furthermore, this chapter discusses other important aspects of online adaptation, such as adaptation time, range of performance, homeostatic and task-oriented evaluation functions, and behavior switching.

The following chapter (see Chapter 5) addresses instead a very specific instance of dynamic environments: one in which the variability is given by other individuals. This permits enhancing our knowledge of the conditions under which online adaptation is possible and on corresponding difficulties (see **RQ1**), simultaneously investigating adaptation in the case of multi-robot systems and rapidly changing conditions (see **RQ5** and **RQ4**, respectively). Specifically, we consider the problem of co-adaptation, which encompasses previous complexities and even adds new ones; indeed, the robot needs to adapt to others behavior and at the same time permit others to adapt to its own. This highlights, for the first time, the need to perform asynchronous adaptation and strengthens the argument for a modulated adaptation intensity.

Finally, in the last chapter among the core exploratory ones (see Chapter 6), we discuss the combination of offline design and online adaptation, along with its corresponding advantages (see **RQ6**); indeed, these approaches possess characteristics that compensate for each other weaknesses. Specifically, we demonstrate that this combination can produce more reliable and fast-adapting robots, while, at the same time, add flexibility to otherwise static controllers. We show that this solution performs exceptionally well when we have partial knowledge of the environment but we can still predict a set of its possible instances in the real world.

After the core experimental part, we devote Chapter 7 to summarize the characteristics of the created adaptive controllers and the important parameters and strategies for their functioning. This permits to show that the parameters and strategies we propose can be generalized to operate on multiple control systems (see **RQ3**). Following, we describe ancillary investigations conducted in parallel with this dissertation experiments (see Chapter 8). Those are often connected with the topics of robotics and adaptation, but they do not find a place in the logical flow of the experimental investigations presented previously. Subsequently, Chapter 9 describes ongoing and future works, thereby articulating the plan of investigations to be pursued.

This dissertation contributes to the advancement of the field of robotics, particularly in the area of online adaptation. The hope is that the information collected herein will serve more ambitious developments of the discipline. The long-term hope is to reach the point at which humanity will be able to create artificial systems with a level of intellect comparable to that of humans.

List of Figures

List of Tables

List of Algorithms

Bibliography

Almansoori, A., M. Alkilabi, and E. Tuci (2024). "On the evolution of adaptable and scalable mechanisms for collective decision-making in a swarm of robots." In: *Swarm Intelligence* 18.1, pp. 79–99.

Annabi, L., A. Pitti, and M. Quoy (2020). "Autonomous learning and chaining of motor primitives using the Free Energy Principle." In: *2020 International Joint Conference on Neural Networks (IJCNN)*. Institute of Electrical and Electronics Engineers, pp. 1–8.

Antonelo, E.A., B. Schrauwen, and D. Stroobandt (2008). "Event detection and localization for small mobile robots using reservoir computing." In: *Neural Networks* 21.6, pp. 862–871.

Arora, S. and P. Doshi (2021). "A survey of inverse reinforcement learning: Challenges, methods and progress." In: *Artificial Intelligence* 297, p. 103500.

Ashby, W.R. (1962). "Principles of the self-organizing system." In: *Principles of Self-Organization: Transactions of the University of Illinois Symposium*. Vol. 7. Pergamon Press, pp. 255–278.

Auddy, S. et al. (2023). "Continual learning from demonstration of robotics skills." In: *Robotics and Autonomous Systems* 165, p. 104427.

Auer, P., N. Cesa-Bianchi, and P. Fischer (2002). "Finite-time Analysis of the Multiarmed Bandit Problem." In: *Machine Learning* 47.2, pp. 235–256.

Auerbach, J. et al. (2014). "RoboGen: Robot Generation through Artificial Evolution." In: *ALIFE 14. The Fourteenth International Conference on the Synthesis and Simulation of Living Systems*. The MIT Press, pp. 136–137.

Ayub, A. and C. Fendley (2022). "Few-Shot Continual Active Learning by a Robot." In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., pp. 30612–30624.

Baele, G. et al. (2009). "Open-ended on-board Evolutionary Robotics for robot swarms." In: *2009 IEEE Congress on Evolutionary Computation*. Institute of Electrical and Electronics Engineers, pp. 1123–1130.

Bakker, B. et al. (2006). "Quasi-online reinforcement learning for robots." In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* Institute of Electrical and Electronics Engineers, pp. 2997–3002.

Balaprakash, P., M. Birattari, and T. Stützle (2007). "Improvement Strategies for the F-Race Algorithm: Sampling Design and Iterative Refinement." In: *Hybrid Metaheuristics*. Vol. 4771. Springer Berlin Heidelberg, pp. 108–122.

Balaska, V. et al. (2020). "Unsupervised semantic clustering and localization for mobile robotics tasks." In: *Robotics and Autonomous Systems* 131, p. 103567.

Baldassarre, G., S. Nolfi, and D. Parisi (2003a). "Evolution of Collective Behavior in a Team of Physically Linked Robots." In: *Applications of Evolutionary Computing. EvoWorkshop 2003: EvoBIO, EvoCOP, EvoIASP, EvoMUSART, EvoROB, and EvoSTIM, Essex, UK, April 14-16, 2003, Proceedings*. Vol. 2611. Springer Berlin Heidelberg, pp. 581–592.

— (2003b). "Evolving Mobile Robots Able to Display Collective Behaviors." In: *Artificial Life* 9.3, pp. 255–267.

Baldini, P. (2022). *Reservoir Computing in robotics: a review*. arXiv: 2206.11222.

Baldini, P., M. Braccini, and A. Roli (2023a). "Online Adaptation of Robots Controlled by Nanowire Networks: A Preliminary Study." In: *Artificial Life and Evolutionary Computation. 16th Italian Workshop, WIVACE 2022, Gaeta, Italy, September 14–16, 2022, Revised Selected Papers*. Vol. 1780. Springer Nature Switzerland, pp. 171–182.

— (2025a). "Fault Recovery Through Online Adaptation of Boolean Network Robots." In: *Sensors* 25.18, p. 5849.

— (2025b). "Impact of LLM-Assisted Coding in Creativity and Robustness of Robot Controllers." In: *HAIC 2025 – Workshop on Human-AI Collaborative Systems 2025. Proceedings of the 1st Workshop on Human-AI Collaborative Systems, co-located with 28th European Conference on Artificial Intelligence (ECAI 2025)*. Vol. 4072. CEUR Workshop Proceedings, pp. 1–12.

Baldini, P., A. Roli, and M. Birattari (2026a). "Compensating Design-Time Uncertainty Through Combined Offline Design And Online Adaptation of Robot Control Software." In: *Autonomous Robots*. To be submitted.

Baldini, P., A. Roli, and M. Braccini (2023b). "On the Performance of Online Adaptation of Robots Controlled by Nanowire Networks." In: *IEEE Access* 11, pp. 144408–144420.

— (2026b). "Online adaptation of heterogeneous robot swarms to a dynamic task." In: *Swarm Intelligence*. Submitted.

Baldini, P. et al. (2025c). "Cooperation in Evolved Modular Soft Robots." In: *Artificial Life and Evolutionary Computation. 18th Italian Workshop, WIVACE 2024, Namur, Belgium, September 11–13, 2024, Revised Selected Papers*. Vol. 2532. Springer Nature Switzerland, pp. 160–172.

Barker, D.J.P. (1995). "Fetal origins of coronary heart disease." In: *British Medical Journal* 311.6998, pp. 171–174.

Bell, G. (2007). *Selection: The Mechanism of Evolution*. 2nd ed. Oxford, United Kingdom: Oxford University Press.

Benedettini, S. et al. (2013). "Dynamical regimes and learning properties of evolved Boolean networks." In: *Neurocomputing* 99, pp. 111–123.

Benzinger, T.H. (1969). "Heat regulation: homeostasis of central temperature in man." In: *Physiological Reviews* 49.4, pp. 671–759.

Bergonti, F. et al. (2024). "Co-Design Optimisation of Morphing Topology and Control of Winged Drones." In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. Institute of Electrical and Electronics Engineers, pp. 8679–8685.

Berseth, G. et al. (2021). "SMiRL: Surprise Minimizing Reinforcement Learning in Unstable Environments." In: *International Conference on Learning Representations (ICLR 2021)*.

Bhovad, P. and S. Li (2021). "Physical reservoir computing with origami and its application to robotic crawling." In: *Scientific Reports* 11.1, p. 13002.

Bianco, R. and S. Nolfi (2004a). "Evolving the Neural Controller for a Robotic Arm Able to Grasp Objects on the Basis of Tactile Sensors." In: *Adaptive Behavior* 12.1, pp. 37–45.

— (2004b). "Toward open-ended evolutionary robotics: evolving elementary robotic units able to self-assemble and self-reproduce." In: *Connection Science* 16.4, pp. 227–248.

Birattari, M. (2024). *Personal communication*.

Birattari, M., A. Ligot, and G. Francesca (2021). "AutoMoDe: A Modular Approach to the Automatic Off-Line Design and Fine-Tuning of Control Software for Robot Swarms." In: *Automated Design of Machine Learning and Search Algorithms*. Springer International Publishing, pp. 73–90.

Birattari, M. et al. (2002). "A Racing Algorithm for Configuring Metaheuristics." In: *GECCO-2002: Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*. Vol. 2. Morgan Kaufmann Publishers, pp. 11–18.

Blix, A.S. (2016). "Adaptations to polar life in mammals and birds." In: *Journal of Experimental Biology* 219.8, pp. 1093–1105.

Bonani, M. et al. (2010). "The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research." In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Institute of Electrical and Electronics Engineers, pp. 4187–4193.

Bonani, M. et al. (2013). "Physical Interactions in Swarm Robotics: The Hand-Bot Case Study." In: *Distributed Autonomous Robotic Systems: The 10th International Symposium*. Vol. 83. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 585–595.

Bongard, J., V. Zykov, and H. Lipson (2006). "Resilient machines through continuous self-modeling." In: *Science* 314.5802, pp. 1118–1121.

Bongard, J.C. and H. Lipson (2004). "Automated damage diagnosis and recovery for remote robotics." In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*. Vol. 4. Institute of Electrical and Electronics Engineers, pp. 3545–3550.

Bongard, J.C. and R. Pfeifer (2003). "Evolving Complete Agents using Artificial Ontogeny." In: *Morpho-functional Machines: The New Species*. Tokyo: Springer Japan, pp. 237–258.

Bos, F. et al. (2022). "Free Energy Principle for State and Input Estimation of a Quadcopter Flying in Wind." In: *2022 International Conference on Robotics and Automation (ICRA)*. Institute of Electrical and Electronics Engineers, pp. 5389–5395.

Braccini, M., G. Aguzzi, and P. Baldini (2026a). "Unraveling Creativity Through Variability: A Comparison of LLMs and Humans in an Educational Q&A Scenario." In: *Technology, Knowledge and Learning*.

Braccini, M., P. Baldini, and A. Roli (2024a). "An Investigation of Graceful Degradation in Boolean Network Robots Subject to Online Adaptation." In: *Artificial Life and Evolutionary*

Computation. 17th Italian Workshop, WIVACE 2023, Venice, Italy, September 6–8, 2023, Revised Selected Papers. Vol. 1977. Springer Nature Switzerland, pp. 202–213.

— (2025). "Cell–Cell Interactions: How Coupled Boolean Networks Tend to Criticality." In: *Artificial Life* 31.1, pp. 68–80.

Braccini, M., E. Barbieri, and A. Roli (2023). "The Role of Dynamical Regimes of Online Adaptive BN-Robots in Noisy Environments." In: *Artificial Life and Evolutionary Computation*. Vol. 1780. Springer Nature Switzerland, pp. 183–194.

Braccini, M., A. Roli, and S. Kauffman (2022a). "A Novel Online Adaptation Mechanism in Artificial Systems Provides Phenotypic Plasticity." In: *Artificial Life and Evolutionary Computation. 15th Italian Workshop, WIVACE 2021, Winterthur, Switzerland, September 15–17, 2021, Revised Selected Papers*. Vol. 1722. Springer Nature Switzerland, pp. 121–132.

Braccini, M., A. Roli, and S.A. Kauffman (2020). *Online adaptation in robots as biological development provides phenotypic plasticity*. arXiv: 2006.02367.

— (2022b). "A Novel Online Adaptation Mechanism in Artificial Systems Provides Phenotypic Plasticity." In: *Artificial Life and Evolutionary Computation*. Vol. 1722. Springer Nature Switzerland, pp. 121–132.

Braccini, M. et al. (2019). "A simplified model of chromatin dynamics drives differentiation process in Boolean models of GRN." In: *2019 Conference on Artificial Life, ALIFE 2019*. The MIT Press, pp. 211–217.

Braccini, M. et al. (2021). "Dynamical properties and path dependence in a gene-network model of cell differentiation." In: *Soft Computing* 25.9, pp. 6775–6787.

Braccini, M. et al. (2022c). "On the Criticality of Adaptive Boolean Network Robots." In: *Entropy* 24.10, 1368:1–1368:21.

Braccini, M. et al. (2024b). "Sensory–Motor Loop Adaptation in Boolean Network Robots." In: *Sensors* 24.11, p. 3393.

Braccini, M. et al. (2026b). "On the LLM Robustness in a Simulated Conversational XR Scenario: A Preliminary Semantic Analysis." In: *Image Analysis and Processing - ICIAP 2025 Workshops. 23rd International Conference, Rome, Italy, September 15–19, 2025, Proceedings, Part II*. Vol. 16170. Springer Nature Switzerland, pp. 637–648.

Bradshaw, A.D. (1965). "Evolutionary Significance of Phenotypic Plasticity in Plants." In: vol. 13. Academic Press, pp. 115–155.

Braitenberg, V. (1986). Cambridge, Massachusetts – London, England: The MIT Press.

Bredeche, N., E. Haasdijk, and A.E. Eiben (2010). "On-Line, On-Board Evolution of Robot Controllers." In: *Artifical Evolution*. Vol. 5975. Springer Berlin Heidelberg, pp. 110–121.

Brooks, R.A. (1992). "Artificial Life and Real Robots." In: *Toward a Practice of Autonomous Systems. Proceedings of the First European Conference on Artificial Life*. Cambridge, Massachusetts, USA: The MIT Press, pp. 3–10.

Cannon, W.B. (1929). "Organization for physiological homeostasis." In: *Physiological Reviews* 9.3, pp. 399–431.

Caravelli, F. et al. (2023). "Mean Field Theory of Self-Organizing Memristive Connectomes." In: *Annalen der Physik* 535.8, p. 2300090.

Carroll, J.L. (2003). "Invited Review: Developmental plasticity in respiratory control." In: *Journal of Applied Physiology* 94.1, pp. 375–389.

Chechik, G., I. Meilijson, and E. Ruppin (1999). "Neuronal Regulation: A Mechanism for Synaptic Pruning During Brain Maturation." In: *Neural Computation* 11.8, pp. 2061–2080.

Chen, C. et al. (2019). "Crowd-Robot Interaction: Crowd-Aware Robot Navigation With Attention-Based Deep Reinforcement Learning." In: *2019 International Conference on Robotics and Automation (ICRA)*. Institute of Electrical and Electronics Engineers, pp. 6015–6022.

Chen, T. et al. (2020). "Unsupervised Anomaly Detection of Industrial Robots Using Sliding-Window Convolutional Variational Autoencoder." In: *IEEE Access* 8, pp. 47072–47081.

Chen, Z. et al. (2025). "Data-Driven Methods Applied to Soft Robot Modeling and Control: A Review." In: *IEEE Transactions on Automation Science and Engineering* 22, pp. 2241–2256.

Cheney, N. et al. (2016). "On the Difficulty of Co-Optimizing Morphology and Control in Evolved Virtual Creatures." In: *ALIFE 2016. The Fifteenth International Conference on the Synthesis and Simulation of Living Systems*. The MIT Press, pp. 226–233.

Cheng, Y. and W. Zhang (2018). "Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels." In: *Neurocomputing* 272, pp. 63–73.

Christensen, D.V. et al. (2022). "2022 roadmap on neuromorphic computing and engineering." In: *Neuromorphic Computing and Engineering* 2.2, p. 22501.

Churamani, N. et al. (2022). "Continual Learning for Affective Robotics: A Proof of Concept for Wellbeing." In: *2022 10th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*. Institute of Electrical and Electronics Engineers, pp. 1–8.

Cliff, D., P. Husbands, and I. Harvey (1993). "Explorations in Evolutionary Robotics." In: *Adaptive Behavior* 2.1, pp. 73–110.

Codling, E.A., M.J. Plank, and S. Benhamou (2008). "Random walk models in biology." In: *Journal of The Royal Society Interface* 5.25, pp. 813–834.

Corucci, F. et al. (2018). "Evolving Soft Locomotion in Aquatic and Terrestrial Environments: Effects of Material Properties and Environmental Transitions." In: *Soft Robotics* 5.4, pp. 475–495.

Crespi, E.J. and R.J. Denver (2005). "Ancient origins of human developmental plasticity." In: *American Journal of Human Biology* 17.1, pp. 44–54.

Cully, A. et al. (2015). "Robots that can adapt like animals." In: *Nature* 521.7553, pp. 503–507.

Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. London, United Kingdom: John Murray.

Demis, E.C. et al. (2016). "Nanoarchitectonic atomic switch networks for unconventional computing." In: *Japanese Journal of Applied Physics* 55.11, 1102B2.

DeWitt, T.J., A. Sih, and D.S. Wilson (1998). "Costs and limits of phenotypic plasticity." In: *Trends in Ecology & Evolution* 13.2, pp. 77–81.

Doncieux, S. et al. (2015). "Evolutionary Robotics: What, Why, and Where to." In: *Frontiers in Robotics and AI* 2.

Dorigo, M. and U. Schnepf (1991). "Organisation of robot behaviour through genetic learning processes." In: *Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments*. Vol. 2. Institute of Electrical and Electronics Engineers, pp. 1456–1460.

Dorigo, M. et al. (2004). "Evolving Self-Organizing Behaviors for a Swarm-Bot." In: *Autonomous Robots* 17.2, pp. 223–245.

Duarte, M. et al. (2016). "Evolution of Collective Behaviors for a Real Swarm of Aquatic Surface Robots." In: *PLOS ONE* 11.3, pp. 1–25.

Dürr, P. et al. (2008). "Evolvability of Neuromodulated Learning for Robots." In: *2008 ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems (LAB-RS)*. Institute of Electrical and Electronics Engineers, pp. 41–46.

Eiben, A.E., E. Haasdijk, and N. Bredeche (2010a). "Embodied, On-line, On-board Evolution for Autonomous Robotics." In: *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*. Vol. 7. Springer, pp. 361–382.

Eiben, A.E., G. Karafotias, and E. Haasdijk (2010b). "Self-Adaptive Mutation in On-line, On-board Evolutionary Robotics." In: *2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop*. Institute of Electrical and Electronics Engineers, pp. 147–152.

Eiben, A.E., S. Kernbach, and E. Haasdijk (2012). "Embodied artificial evolution: artificial evolutionary systems in the 21st Century." In: *Evolutionary Intelligence* 5.4, pp. 261–272.

Elman, J.L. (1990). "Finding Structure in Time." In: *Cognitive Science* 14.2, pp. 179–211.

Feola, L. and V. Trianni (2022). "Adaptive Strategies for Team Formation in Minimalist Robot Swarms." In: *IEEE Robotics and Automation Letters* 7.2, pp. 4079–4085.

Ferigo, A. et al. (2025). "Totipotent neural controllers for modular soft robots: Achieving specialization in body-brain co-evolution through Hebbian learning." In: *Neurocomputing* 614, p. 128811.

Ferrante, E. et al. (2015). "Evolution of Self-Organized Task Specialization in Robot Swarms." In: *PLOS Computational Biology* 11.8, pp. 1–21.

Fisher, R.A. (1930). *The Genetical Theory of Natural Selection*. Oxford: The Clarendon Press.

Floreano, D. and F. Mondada (1994). "Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot." In: *Animals to Animats 3. Proceedings of the Third International Conference on Simulation of Adaptive Behavior*. The MIT Press, pp. 421–430.

— (1996). "Evolution of homing navigation in a real mobile robot." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.3, pp. 396–407.

Floreano, D. and S. Nolfi (1997). "God Save the Red Queen! Competition in Co-Evolutionary Robotics." In: *Genetic Programming 1997: Proceedings of the Second Annual Conference*. Morgan Kaufman Publishers.

Floreano, D., S. Nolfi, and F. Mondada (1998). "Co-Evolution and Ontogenetic Change in Competing Robots." In: *Advances in the Evolutionary Synthesis of Intelligent Agents*. The MIT Press, pp. 273–305.

Floreano, D. and J. Urzelai (1999). "Evolution of Neural Controllers with Adaptive Synapses and Compact Genetic Encoding." In: *Advances in Artificial Life*. Vol. 1674. Springer Berlin Heidelberg, pp. 183–194.

— (2000a). "Evolutionary on-line selforganisation of autonomous robots." In: *Proceedings of the 5th International Conference on Artificial Life and Robotics*.

— (2000b). "Evolutionary robots with on-line self-organization and behavioral fitness." In: *Neural Networks* 13.4, pp. 431–443.

— (2001a). "Evolution of Plastic Control Networks." In: *Autonomous Robots* 11.3, pp. 311–317.

— (2001b). "Neural morphogenesis, synaptic plasticity, and evolution." In: *Theory in Biosciences* 120.3, pp. 225–240.

Francesca, G. et al. (2014). "An Experiment in Automatic Design of Robot Swarms." In: *Swarm Intelligence*. Vol. 8667. Springer International Publishing, pp. 25–37.

Francesca, G. et al. (2015). "AutoMoDe-Chocolate: automatic design of control software for robot swarms." In: *Swarm Intelligence* 9.2, pp. 125–152.

Friston, K. (2009). "The free-energy principle: a rough guide to the brain?" In: *Trends in Cognitive Sciences* 13.7, pp. 293–301.

Friston, K., J. Kilner, and L. Harrison (2006). "A free energy principle for the brain." In: *Journal of Physiology-Paris* 100.1, pp. 70–87.

Fu, K. et al. (2020). "Reservoir Computing with Neuromemristive Nanowire Networks." In: *2020 International Joint Conference on Neural Networks (IJCNN)*. Institute of Electrical and Electronics Engineers, pp. 1–8.

Fukshansky, L. (2011). "Revisiting the hexagonal lattice: on optimal lattice circle packing." In: *Elemente der Mathematik* 66.1, pp. 1–9.

Galván, A. (2010). "Neural plasticity of development and learning." In: *Human Brain Mapping* 31.6, pp. 879–890.

Garzón Ramos, D. and M. Birattari (2020). "Automatic Design of Collective Behaviors for Robots that Can Display and Perceive Colors." In: *Applied Sciences* 10.13, p. 4654.

Garzón Ramos, D. et al. (2025). "Automatic Design of Robot Swarms under Concurrent Design Criteria: A Study Based on Iterated F-Race." In: *Advanced Intelligent Systems* 7.1, p. 2400332.

Ghadirzadeh, A. et al. (2021). "Human-Centered Collaborative Robots With Deep Reinforcement Learning." In: *IEEE Robotics and Automation Letters* 6.2, pp. 566–571.

Ghalambor, C.K., L.M. Angeloni, and S.P. Carroll (2010). "Behavior as Phenotypic Plasticity." In: *Evolutionary Behavioral Ecology*. Oxford University Press, pp. 90–107.

Ghalambor, C.K. et al. (2007). "Adaptive versus non-adaptive phenotypic plasticity and the potential for contemporary adaptation in new environments." In: *Functional Ecology* 21.3, pp. 394–407.

Gharbi, I. et al. (2023). *Show me what you want: Inverse reinforcement learning to automatically design robot swarms by demonstration*. arXiv: 2301.06864.

Gomes, J. and A.L. Christensen (2018). "Task-Agnostic Evolution of Diverse Repertoires of Swarm Behaviours." In: *Swarm Intelligence*. Vol. 11172. Springer International Publishing, pp. 225–238.

Gomes, J., P. Mariano, and A.L. Christensen (2019). "Challenges in cooperative coevolution of physically heterogeneous robot teams." In: *Natural Computing* 18.1, pp. 29–46.

Gonzalez, R. and K. Iagnemma (2018). *DeepTerramechanics: Terrain Classification and Slip Estimation for Ground Robots via Deep Learning*. arXiv: 1806.07379.

Gould, S.J. and E.S. Vrba (1982). "Exaptation—a Missing Term in the Science of Form." In: *Paleobiology* 8.1, pp. 4–15.

Groß, R. and M. Dorigo (2004). "Evolving a Cooperative Transport Behavior for Two Simple Robots." In: *Artificial Evolution. 6th International Conference, Evolution Artificielle, EA 2003, Marseilles, France, October 27-30, 2003, Revised Selected Papers*. Vol. 2936. Springer Berlin Heidelberg, pp. 305–316.

Ha, D. and J. Schmidhuber (2018a). "Recurrent World Models Facilitate Policy Evolution." In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., pp. 2451–2463.

— (2018b). *World Models*. arXiv: 1803.10122.

Haasdijk, E., A. Atta-ul-Qayyum, and A.E. Eiben (2011). "Racing to improve on-line, on-board evolutionary robotics." In: *GECCO '11. Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*. Association for Computing Machinery, pp. 187–194.

Haasdijk, E., A.E. Eiben, and G. Karafotias (2010). "On-line evolution of robot controllers by an encapsulated evolution strategy." In: *IEEE Congress on Evolutionary Computation*. Institute of Electrical and Electronics Engineers, pp. 1–7.

Haddon-Hill, G.W. and S. Murata (2024). "Active Vision for Physical Robots Using the Free Energy Principle." In: *Artificial Neural Networks and Machine Learning – ICANN 2024. 33rd International Conference on Artificial Neural Networks, Lugano, Switzerland, September 17–20, 2024, Proceedings, Part X*. Vol. 15025. Springer Nature Switzerland, pp. 270–284.

Hajizada, E., B. Swaminathan, and Y. Sandamirskaya (2024). "Continual Learning for Autonomous Robots: A Prototype-based Approach." In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Institute of Electrical and Electronics Engineers, pp. 13988–13995.

Haller, B. von, A. Ijspeert, and D. Floreano (2005). "Co-evolution of Structures and Controllers for Neubot Underwater Modular Robots." In: *Advances in Artificial Life*. Vol. 3630. Springer Berlin Heidelberg, pp. 189–199.

Hamann, H. (2014). "Evolution of Collective Behaviors by Minimizing Surprise." In: *ALIFE 14. The Fourteenth International Conference on the Synthesis and Simulation of Living Systems*. The MIT Press, pp. 344–351.

Hansen, E.A. et al. (2018). "Achieving Connectivity Between Wide Areas Through Self-Organising Robot Swarms Using Embodied Evolution." In: *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. Institute of Electrical and Electronics Engineers, pp. 875–883.

Hansen, N. and A. Ostermeier (2001). "Completely Derandomized Self-Adaptation in Evolution Strategies." In: *Evolutionary Computation* 9.2, pp. 159–195.

Hansen, P. and N. Mladenović (2001). "Variable neighborhood search: Principles and applications." In: *European Journal of Operational Research* 130.3, pp. 449–467.

Harding, S. and J.F. Miller (2005). "Evolution in materio: a real-time robot controller in liquid crystal." In: *2005 NASA/DoD Conference on Evolvable Hardware (EH'05)*. Institute of Electrical and Electronics Engineers, pp. 229–238.

Hartfelder, K. and W. Engels (1998). "2 Social Insect Polymorphism: Hormonal Regulation of Plasticity in Development and Reproduction in the Honeybee." In: ed. by R.A. Pedersen and G.P. Schatten. Vol. 40. Current Topics in Developmental Biology. Academic Press, pp. 45–77.

Hasselmann, K., F. Robert, and M. Birattari (2018a). "Automatic Design of Communication-Based Behaviors for Robot Swarms." In: *Swarm Intelligence*. Vol. 11172. Springer International Publishing, pp. 16–29.

Hasselmann, K. et al. (2018b). *Reference models for AutoMoDe*. Tech. rep. TR/IRIDIA/2018-002. IRIDIA, Université Libre de Bruxelles, Belgium.

Hasselmann, K. et al. (2021). "Empirical assessment and comparison of neuro-evolutionary methods for the automatic off-line design of robot swarms." In: *Nature Communications* 12.1, p. 4345.

Häussermann, K., O. Zweigle, and P. Levi (2015). "A Novel Framework for Anomaly Detection of Robot Behaviors." In: *Journal of Intelligent & Robotic Systems* 77.2, pp. 361–375.

Hebb, D.O. (1949). *The organization of behavior: a neuropsychological theory*. New York, NY: John Wiley & Sons, Inc.

Heinerman, J. et al. (2016). "On-line Evolution of Foraging Behaviour in a Population of Real Robots." In: *Applications of Evolutionary Computation*. Vol. 9598. Springer International Publishing, pp. 198–212.

Hiller, J. and H. Lipson (2012). "Automatic Design and Manufacture of Soft Robots." In: *IEEE Transactions on Robotics* 28.2, pp. 457–466.

Holland, J.H. (1992). "Genetic Algorithms." In: *Scientific American* 267.1, pp. 66–73.

Hu, Y. et al. (2020). "Learning Control for Robotic Manipulator with Free Energy." In: *2020 39th Chinese Control Conference (CCC)*. Institute of Electrical and Electronics Engineers, pp. 1903–1908.

Huang, S., I. Ernberg, and S. Kauffman (2009). "Cancer attractors: A systems view of tumors from a gene network dynamics and developmental perspective." In: *Seminars in Cell & Developmental Biology* 20.7, pp. 869–876.

Huang, S. et al. (2005). "Cell Fates as High-Dimensional Attractor States of a Complex Gene Regulatory Network." In: *Physical Review Letters* 94.12, p. 128701.

Husbands, P. et al. (2021). "Recent advances in evolutionary and bio-inspired adaptive robotics: Exploiting embodied dynamics." In: *Applied Intelligence* 51.9, pp. 6467–6496.

Jakobi, N., P. Husbands, and I. Harvey (1995). "Noise and the reality gap: The use of simulation in evolutionary robotics." In: *Advances in Artificial Life*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 704–720.

Jäncke, L. (2009). "The plastic human brain." In: *Restorative Neurology and Neuroscience* 27.5, pp. 521–538.

Jang, B. et al. (2019). "Q-Learning Algorithms: A Comprehensive Classification and Applications." In: *IEEE Access* 7, pp. 133653–133667.

Jo, S.H. et al. (2010). "Nanoscale memristor device as synapse in neuromorphic systems." In: *Nano letters* 10.4, pp. 1297–1301.

Jones, S. et al. (2018). "Evolving Behaviour Trees for Swarm Robotics." In: *Distributed Autonomous Robotic Systems: The 13th International Symposium*. Vol. 6. Cham: Springer International Publishing, pp. 487–501.

Jones, S. et al. (2019). "Onboard Evolution of Understandable Swarm Behaviors." In: *Advanced Intelligent Systems* 1, p. 1900031.

Joshi, S., S. Kumra, and F. Sahin (2020). "Robotic Grasping using Deep Reinforcement Learning." In: *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. Institute of Electrical and Electronics Engineers, pp. 1461–1466.

Kalashnikov, D. et al. (2018). "Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation." In: *Proceedings of The 2nd Conference on Robot Learning*. Vol. 87. Proceedings of Machine Learning Research (PMLR), pp. 651–673.

Karlebach, G. and R. Shamir (2008). "Modelling and analysis of gene regulatory networks." In: *Nature Reviews Molecular Cell Biology* 9.10, pp. 770–780.

Katju, V. and U. Bergthorsson (2018). "Old Trade, New Tricks: Insights into the Spontaneous Mutation Process from the Partnering of Classical Mutation Accumulation Experiments with High-Throughput Genomic Approaches." In: *Genome Biology and Evolution* 11.1, pp. 136–165.

Kauffman, S.A. (1969). "Metabolic stability and epigenesis in randomly constructed genetic nets." In: *Journal of theoretical biology* 22.3, pp. 437–467.

Kegeleirs, M. et al. (2024). "Transferability in the Automatic Off-Line Design of Robot Swarms: From Sim-to-Real to Embodiment and Design-Method Transfer Across Different Platforms." In: *IEEE Robotics and Automation Letters* 9.3, pp. 2758–2765.

Kengyel, D. et al. (2015). "Potential of Heterogeneity in Collective Behaviors: A Case Study on Heterogeneous Swarms." In: *PRIMA 2015: Principles and Practice of Multi-Agent Systems*. Vol. 9387. Springer International Publishing, pp. 201–217.

Khatib, A. et al. (2024). "Enhancing Multi-Robot Systems Cooperation through Machine Learning-based Anomaly Detection in Target Pursuit." In: *Journal of Robotics and Control (JRC)* 5.3, pp. 893–901.

Kolb, B. and R. Gibb (2011). "Brain plasticity and behaviour in the developing brain." In: *Journal of the Canadian Academy of Child and Adolescent Psychiatry* 20.4, pp. 265–276.

Kriegman, S. et al. (2020). "A scalable pipeline for designing reconfigurable organisms." In: *Proceedings of the National Academy of Sciences* 117.4, pp. 1853–1859.

Kuckling, J. et al. (2018). "Behavior Trees as a Control Architecture in the Automatic Modular Design of Robot Swarms." In: *Swarm Intelligence*. Vol. 11172. Springer International Publishing, pp. 30–43.

Kuzawa, C.W. and Z.M. Thayer (2011). "Timescales of human adaptation: the role of epigenetic processes." In: *Epigenomics* 3.2, pp. 221–234.

Langerhans, R.B. (2008). "Predictability of phenotypic differentiation across flow regimes in fishes." In: *Integrative and Comparative Biology* 48.6, pp. 750–768.

Larger, L. et al. (2017). "High-Speed Photonic Reservoir Computing Using a Time-Delay-Based Architecture: Million Words per Second Classification." In: *Phys. Rev. X* 7.1, p. 011015.

Lasker, G.W. (1969). "Human Biological Adaptability." In: *Science* 166.3912, pp. 1480–1486.

Lema, S.C. (2008). "The Phenotypic Plasticity of Death Valley's Pupfish: Desert fish are revealing how the environment alters development to modify body shape and behavior." In: *American Scientist* 96.1, pp. 28–36.

Lesort, T. et al. (2020). "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges." In: *Information Fusion* 58, pp. 52–68.

Levis, N.A., A.J. Isdaner, and D.W. Pfennig (2018). "Morphological novelty emerges from pre-existing phenotypic plasticity." In: *Nature Ecology & Evolution* 2.8, pp. 1289–1297.

Lewis, M.A., A.H. Fagg, and A. Solidum (1992). "Genetic programming approach to the construction of a neural network for control of a walking robot." In: *Proceedings 1992 IEEE International Conference on Robotics and Automation*. Vol. 3. Institute of Electrical and Electronics Engineers, pp. 2618–2623.

Li, D. and O. Okhrin (2024). "A platform-agnostic deep reinforcement learning framework for effective Sim2Real transfer towards autonomous driving." In: *Communications Engineering* 3.1, p. 147.

Li, H., H. Jiao, and Z. Yang (2023). "Ship trajectory prediction based on machine learning and deep learning: A systematic review and methods analysis." In: *Engineering Applications of Artificial Intelligence* 126, p. 107062.

Liang, H. et al. (2022). "Multifingered Grasping Based on Multimodal Reinforcement Learning." In: *IEEE Robotics and Automation Letters* 7.2, pp. 1174–1181.

Ligot, A., K. Hasselmann, and M. Birattari (2020). "AutoMoDe-Arlequin: Neural Networks as Behavioral Modules for the Automatic Design of Probabilistic Finite-State Machines." In: *Swarm Intelligence*. Vol. 12421. Springer International Publishing, pp. 271–281.

Lipson, H. and J.B. Pollack (2000). "Automatic design and manufacture of robotic lifeforms." In: *Nature* 406.6799, pp. 974–978.

Lipson, H. et al. (2006). "Evolutionary Robotics for Legged Machines: From Simulation to Physical Reality." In: *Intelligent Autonomous Systems* 9, pp. 11–18.

Liu, J. et al. (2024). "Continual Learning for Robotic Grasping Detection With Knowledge Transferring." In: *IEEE Transactions on Industrial Electronics* 71.9, pp. 11019–11027.

Löwel, S. and W. Singer (1992). "Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity." In: *Science* 255.5041, pp. 209–212.

Luque, B. and R.V. Solé (1997). "Phase transitions in random networks: Simple analytic determination of critical points." In: *Phys. Rev. E* 55.1, pp. 257–260.

Mahdavi, S.H. and P.J. Bentley (2006). "Innately adaptive robotics through embodied evolution." In: *Autonomous Robots* 20.2, pp. 149–163.

Martin, S.J., P.D. Grimwood, and R.G.M. Morris (2000). "Synaptic Plasticity and Memory: An Evaluation of the Hypothesis." In: *Annual Review of Neuroscience* 23, pp. 649–711.

Meeûs, T. de, F. Prugnolle, and P. Agnew (2007). "Asexual reproduction: Genetics and evolutionary aspects." In: *Cellular and Molecular Life Sciences* 64.11, pp. 1355–1372.

Michel, O. (1998). "Webots: Symbiosis Between Virtual and Real Mobile Robots." In: *Virtual Worlds. First International Conference, VW'98 Paris, France, July 1–3, 1998 Proceedings*. Vol. 1434. Springer Berlin Heidelberg, pp. 254–263.

— (2004). "Cyberbotics Ltd. Webots™: Professional Mobile Robot Simulation." In: *International Journal of Advanced Robotic Systems* 1.1, p. 5.

Miglino, O., K. Nafasi, and C.E. Taylor (1994). "Selection for Wandering Behavior in a Small Robot." In: *Artificial Life* 2.1, pp. 101–116.

Milano, G., E. Miranda, and C. Ricciardi (2022). "Connectome of memristive nanowire networks through graph theory." In: *Neural Networks* 150, pp. 137–148.

Milano, G. and C. Ricciardi (2023). "Nanowire memristor as artificial synapse in random networks." In: *Intelligent Nanotechnology*. Elsevier, pp. 219–246.

Milano, G. et al. (2019). "Recent Developments and Perspectives for Memristive Devices Based on Metal Oxide Nanowires." In: *Advanced Electronic Materials* 5.9, p. 1800909.

Milano, G. et al. (2020). "Brain-Inspired Structural Plasticity through Reweighting and Rewiring in Multi-Terminal Self-Organizing Memristive Nanowire Networks." In: *Advanced Intelligent Systems* 2.8, p. 2080071.

Miller, J., S.L. Harding, and G. Tufte (2014). "Evolution-in-materio: evolving computation in materials." In: *Evolutionary Intelligence* 7.1, pp. 49–67.

Miller, J.F. and K. Downing (2002). "Evolution in materio: looking beyond the silicon box." In: *Proceedings 2002 NASA/DoD Conference on Evolvable Hardware*. Institute of Electrical and Electronics Engineers, pp. 167–176.

Mondada, F. et al. (2009). "The e-puck, a Robot Designed for Education in Engineering." In: *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*. Vol. 1. 1. IPCB: Instituto Politécnico de Castelo Branco, pp. 59–65.

Montagna, S., M. Braccini, and A. Roli (2021). "The impact of self-loops on Boolean networks attractor landscape and implications for cell differentiation modelling." In: *IEEE/ACM transactions on computational biology and bioinformatics* 18.6, pp. 2702–2713.

Morlino, G., V. Trianni, and E. Tuci (2012). "Evolution of Collective Perception in a Group of Autonomous Robots." In: *Computational Intelligence*. Vol. 399. Springer Berlin Heidelberg, pp. 67–80.

Nadizar, G. et al. (2021). "On the effects of pruning on evolved neural controllers for soft robots." In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '21. Lille, France: Association for Computing Machinery, pp. 1744–1752.

Nadizar, G. et al. (2022). "Merging pruning and neuroevolution: towards robust and efficient controllers for modular soft robots." In: *The Knowledge Engineering Review* 37, e3.

Nadizar, G. et al. (2023). "An experimental comparison of evolved neural network models for controlling simulated modular soft robots." In: *Applied Soft Computing* 145, p. 110610.

Nair, A. et al. (2021). "AWAC: Accelerating Online Reinforcement Learning with Offline Datasets." In: *International Conference on Learning Representations (ICLR 2021)*.

Nakajima, K. and I. Fischer (2021). Springer Singapore.

Nakajima, K. et al. (2013). "A soft body as a reservoir: case studies in a dynamic model of octopus-inspired soft robotic arm." In: *Frontiers in Computational Neuroscience* 7.

Nei, M. (2013). *Mutation-Driven Evolution*. 1st ed. Oxford, United Kingdom: Oxford University Press.

Nie, X. et al. (2024). "Online Active Continual Learning for Robotic Lifelong Object Recognition." In: *IEEE Transactions on Neural Networks and Learning Systems* 35.12, pp. 17790–17804.

Niv, Y. et al. (2002). "Evolution of Reinforcement Learning in Uncertain Environments: A Simple Explanation for Complex Foraging Behaviors." In: *Adaptive Behavior* 10.1, pp. 5–24.

Nolfi, S. (1997). "Evolving non-trivial behaviors on real robots: A garbage collecting robot." In: *Robotics and Autonomous Systems* 22.3, pp. 187–198.

Nolfi, S. and D. Floreano (1998a). "Coevolving Predator and Prey Robots: Do "Arms Races" Arise in Artificial Evolution?" In: *Artificial Life* 4.4, pp. 311–335.

— (1998b). "How co-evolution can enhance the adaptive power of artificial evolution: Implications for evolutionary robotics." In: *Evolutionary Robotics*. Vol. 1468. Springer Berlin Heidelberg, pp. 22–38.

Nolfi, S. and D. Marocco (2002). "Evolving robots able to visually discriminate between objects with different size." In: *International Journal of Robotics and Automation* 17.4, pp. 163–170.

Nolfi, S. and D. Parisi (1995). "Evolving non-trivial behaviors on real robots: An autonomous robot that picks up objects." In: *Topics in Artificial Intelligence*. Vol. 992. Springer Berlin Heidelberg, pp. 243–254.

Nygaard, T.F. et al. (2018). "Real-world evolution adapts robot morphology and control to hardware limitations." In: *GECCO '18. Proceedings of the Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, pp. 125–132.

Oliver, G., P. Lanillos, and G. Cheng (2022). "An Empirical Study of Active Inference on a Humanoid Robot." In: *IEEE Transactions on Cognitive and Developmental Systems* 14.2, pp. 462–471.

Ormrod, J.E. (2018). *Human Learning*. 8th ed. New York, NY: Pearson plc.

Parker, L.E., D. Rus, and G.S. Sukhatme (2016). "Multiple Mobile Robot Systems." In: *Springer Handbook of Robotics*. Cham: Springer International Publishing, pp. 1335–1384.

Pinciroli, C. et al. (2012). "ARGoS: a Modular, Parallel, Multi-Engine Simulator for Multi-Robot Systems." In: *Swarm Intelligence* 6.4, pp. 271–295.

Pini, G. and E. Tuci (2008). "On the design of neuro-controllers for individual and social learning behaviour in autonomous robots: an evolutionary approach." In: *Connection Science* 20.2–3, pp. 211–230.

Piqué, F. et al. (2022). "Controlling Soft Robotic Arms Using Continual Learning." In: *IEEE Robotics and Automation Letters* 7.2, pp. 5469–5476.

Purves, D. et al. (2012). *Neuroscience*. 5th ed. Sunderland, Massachusetts: Sinauer Associates Inc.

Quick, T. et al. (2003). "Evolving Embodied Genetic Regulatory Network-Driven Control Systems." In: *Advances in Artificial Life. 7th European Conference, ECAL 2003, Dortmund, Germany, September 14-17, 2003, Proceedings*. Vol. 2801. Springer Berlin Heidelberg, pp. 266–277.

Quinn, M. (2000). "Evolving co-operative homogeneous multi-robot teams." In: *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*. Vol. 3. Institute of Electrical and Electronics Engineers, pp. 1798–1803.

— (2001). "A comparison of approaches to the evolution of homogeneous multi-robot teams." In: *Proceedings of the 2001 Congress on Evolutionary Computation*. Vol. 1. Institute of Electrical and Electronics Engineers, pp. 128–135.

Quinn, M. et al. (2003). "Evolving controllers for a homogeneous system of physical robots: structured cooperation with minimal sensors." In: *Philosophical Transactions of the Royal Society A* 361, pp. 2321–2343.

Ram, A. et al. (1997). "Case-based reactive navigation: a method for on-line selection and adaptation of reactive robotic control parameters." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 27.3, pp. 376–394.

Ricciardi, C. and G. Milano (2022). "*In Materia* Should Be Used Instead of *In Materio*." In: *Frontiers in Nanotechnology* 4.

Roli, A. et al. (2011). "On the design of Boolean network robots." In: *Applications of Evolutionary Computation. EvoApplications 2011: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC, Torino, Italy, April 27-29, 2011, Proceedings, Part I*. Vol. 6624. Springer Berlin Heidelberg, pp. 43–52.

Roli, A. et al. (2018). "Dynamical Criticality: Overview and Open Questions." In: *Journal of Systems Science and Complexity* 31.3, pp. 647–663.

Rothman, D.L., S.C. Stearns, and R.G. Shulman (2021). "Gene expression regulates metabolite homeostasis during the Crabtree effect: Implications for the adaptation and evolution of Metabolism." In: *Proceedings of the National Academy of Sciences* 118.2, e2014013118.

Rybczak, M., N. Popowniak, and A. Lazarowska (2024). "A Survey of Machine Learning Approaches for Mobile Robot Control." In: *Robotics* 13.1, p. 12.

Samuel, D.J. and F. Cuzzolin (2021). "Unsupervised Anomaly Detection for a Smart Autonomous Robotic Assistant Surgeon (SARAS) Using a Deep Residual Autoencoder." In: *IEEE Robotics and Automation Letters* 6.4, pp. 7256–7261.

Sanjuán, R. et al. (2010). "Viral Mutation Rates." In: *Journal of Virology* 84.19, pp. 9733–9748.

Schaal, S. (2002). "Learning Robot Control." In: *The handbook of brain theory and neural networks*. Cambridge, Massachusetts: The MIT Press, pp. 983–987.

Schranz, M. et al. (2020). "Swarm Robotic Behaviors and Current Applications." In: *Frontiers in Robotics and AI* 7, p. 36.

Serra, R. et al. (2007). "Why a simple model of genetic regulatory networks describes the distribution of avalanches in gene expression data." In: *Journal of Theoretical Biology* 246.3, pp. 449–460.

Shaheen, K. et al. (2022). "Continual Learning for Real-World Autonomous Systems: Algorithms, Challenges and Frameworks." In: *Journal of Intelligent & Robotic Systems* 105.1, p. 9.

Shrestha, A. et al. (2025). "Emergent Dynamics in Heterogeneous Life-Like Cellular Automata." In: *Artificial Life and Evolutionary Computation. 18th Italian Workshop, WIVACE 2024, Namur, Belgium, September 11–13, 2024, Revised Selected Papers*. Vol. 2532. Springer Nature Switzerland, pp. 1–15.

Silva, F., L. Correia, and A.L. Christensen (2014a). "Speeding Up Online Evolution of Robotic Controllers with Macro-neurons." In: *Applications of Evolutionary Computation*. Vol. 8602. Springer Berlin Heidelberg, pp. 765–776.

— (2015a). "A Case Study on the Scalability of Online Evolution of Robotic Controllers." In: *Progress in Artificial Intelligence*. Vol. 9273. Springer International Publishing, pp. 189–200.

— (2016). "Leveraging Online Racing and Population Cloning in Evolutionary Multirobot Systems." In: *Applications of Evolutionary Computation*. Vol. 9598. Springer International Publishing, pp. 165–180.

— (2017). "Evolutionary online behaviour learning and adaptation in real robots." In: *Royal Society Open Science* 4, p. 160938.

Silva, F., P. Urbano, and A.L. Christensen (2012a). "Adaptation of Robot Behaviour through Online Evolution and Neuromodulated Learning." In: *Advances in Artificial Intelligence – IBERAMIA 2012. 13th Ibero-American Conference on AI, Cartagena de Indias, Colombia, November 13-16, 2012, Proceedings*. Vol. 7636. Springer Berlin Heidelberg, pp. 300–309.

— (2014b). "Online Evolution of Adaptive Robot Behaviour." In: *International Journal of Natural Computing Research* 4.2, pp. 59–77.

Silva, F. et al. (2012b). "odNEAT: An Algorithm for Distributed Online, Onboard Evolution of Robot Behaviours." In: *ALIFE 2012. The Thirteenth International Conference on the Synthesis and Simulation of Living Systems*. The MIT Press, pp. 251–258.

Silva, F. et al. (2015b). "odNEAT: An Algorithm for Decentralised Online Evolution of Robotic Controllers." In: *Evolutionary Computation* 23.3, pp. 421–449.

Silver, D. et al. (2016). "Mastering the game of Go with deep neural networks and tree search." In: *Nature* 529.7587, pp. 484–489.

Sims, K. (1994). "Evolving 3D Morphology and Behavior by Competition." In: *Artificial Life* 1.4, pp. 353–372.

Singh, B., R. Kumar, and V.P. Singh (2022). "Reinforcement learning in robotic applications: a comprehensive survey." In: *Artificial Intelligence Review* 55.2, pp. 945–990.

Slivkins, A. (2019). "Introduction to Multi-Armed Bandits." In: *Foundations and Trends® in Machine Learning* 12.1–2, pp. 1–286.

Soltoggio, A. et al. (2008). "Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios." In: *Artificial Life XI. Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*. The MIT Press, pp. 569–576.

Sommer, R.J. (2020). "Phenotypic Plasticity: From Theory and Genetics to Current and Future Challenges." In: *Genetics* 215.1, pp. 1–13.

Sutton, R.S. and A.G. Barto (2018). Cambridge, Massachusetts – London, England: The MIT Press.

Szpirer, J., D.G. Ramos, and M. Birattari (2024). "Automatic design of robot swarms that perform composite missions: an approach based on inverse reinforcement learning." In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Institute of Electrical and Electronics Engineers, pp. 5791–5798.

Trianni, V. et al. (2003). "Evolving Aggregation Behaviors in a Swarm of Robots." In: *Advances in Artificial Life. 7th European Conference, ECAL 2003, Dortmund, Germany, September 14-17, 2003, Proceedings*. Vol. 2801. Springer Berlin Heidelberg, pp. 865–874.

Tuci, E. et al. (2008). "Evolving Homogeneous Neurocontrollers for a Group of Heterogeneous Robots: Coordinated Motion, Cooperation, and Acoustic Communication." In: *Artificial Life* 14.2, pp. 157–178.

Uriot, T. et al. (2022). "Spacecraft collision avoidance challenge: Design and results of a machine learning competition." In: *Astrodynamics* 6.2, pp. 121–140.

Urzelai, J. and D. Floreano (1999). "Incremental Evolution with Minimal Resources." In: *First International Khepera Workshop (IKW'1999)*.

— (2001). "Evolution of Adaptive Synapses: Robots with Fast Adaptive Behavior in New Environments." In: *Evolutionary Computation* 9.4, pp. 495–524.

Urzelai, J. et al. (1998). "Incremental Robot Shaping." In: *Connection Science* 10.3-4, pp. 341–360.

Van de Maele, T. et al. (2021). "Active Vision for Robot Manipulators Using the Free Energy Principle." In: *Frontiers in Neurorobotics* 15.

Ven, G.M. van de, T. Tuytelaars, and A.S. Tolias (2022). "Three types of incremental learning." In: *Nature Machine Intelligence* 4.12, pp. 1185–1197.

Vettoruzzo, A. et al. (2024). "Advances and Challenges in Meta-Learning: A Technical Review." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.7, pp. 4763–4779.

Villani, M., A. Barbieri, and R. Serra (2011). "A dynamical model of genetic networks for cell differentiation." In: *PLoS ONE* 6.3, e17703.

Wang, D., H. Deng, and Z. Pan (2020). "MRCDRL: Multi-robot coordination with deep reinforcement learning." In: *Neurocomputing* 406, pp. 68–76.

Wang, L. et al. (2024). "A Comprehensive Survey of Continual Learning: Theory, Method and Application." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.8, pp. 5362–5383.

Watkins, C.J.C.H. and P. Dayan (1992). "Q-learning." In: *Machine Learning* 8.3, pp. 279–292.

Wen, Y. et al. (2020). "Online Reinforcement Learning Control for the Personalization of a Robotic Knee Prosthesis." In: *IEEE Transactions on Cybernetics* 50.6, pp. 2346–2356.

Williams, G.C. (2019). *Adaptation and Natural Selection: A Critique of Some Current Evolutionary Thought*. 2nd ed. Princeton, New Jersey, United States: Princeton University Press.

Williams, J.-P. et al. (2017). "The global surface temperatures of the Moon as measured by the Diviner Lunar Radiometer Experiment." In: *Icarus* 283, pp. 300–325.

Ye, H. et al. (2024). "Person Re-Identification for Robot Person Following With Online Continual Learning." In: *IEEE Robotics and Automation Letters* 9.11, pp. 9151–9158.

Zhang, K. et al. (2020). "Continuous reinforcement learning to adapt multi-objective optimization online for robot motion." In: *International Journal of Advanced Robotic Systems* 17.2, p. 1729881420911491.

Zheng, K. et al. (2024). "Adaptive collision avoidance decisions in autonomous ship encounter scenarios through rule-guided vision supervised learning." In: *Ocean Engineering* 297, p. 117096.