The Dissertation Committee for Federico Pigozzi
certifies that this is the approved version of the following dissertation:

Harnessing the Power of Collective Intelligence: the Case Study of Voxel-based Soft Robots

Committee:

Dr. Eric Medvet, Supervisor

Raffaele Pesenti

Erica Salvato

Luca Valcarenghi

Harnessing the Power of Collective Intelligence: the Case Study of Voxel-based Soft Robots

by
Federico Pigozzi

Dissertation

Presented to the Faculty of

The University of Trieste

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Trieste
December 2023

Epigraph

It's like in the great stories, Mr. Frodo. The ones that really mattered. Full of darkness and danger they were. And sometimes you didn't want to know the end. Because how could the end be happy? How could the world go back to the way it was when so much bad had happened? But in the end, it's only a passing thing, this shadow. Even darkness must pass. A new day will come. And when the sun shines it will shine out the clearer. Those were the stories that stayed with you. That meant something, even if you were too small to understand why. But I think, Mr. Frodo, I do understand. I know now. Folk in those stories had lots of chances of turning back, only they didn't. They kept going, because they were holding on to something. That there is some good in this world, and it's worth fighting for.

—J.R.R. Tolkien

Acknowledgments

Looking back to my Ph.D. journey, I feel a tempest of emotions. It was the most intense experience of my life, blending some of the greatest joys and some of the greatest disappointments. All in all, the Ph.D. experience shaped me as an academic and a person of culture in general, teaching me skills that I believe will prove invaluable in any aspect of life. Amid this tumult, there are several people I would like to thank. In case you are reading this and I omitted your name, I deeply apologize for that, but I wrote these acknowledgments at 5 AM while listening to a medieval version of 50 cent ft. Eminem and drinking orange juice, so I hope you will forgive me.

I thank my supervisor Eric Medvet, for the patience and dedication he showed while mentoring me as the head of the Evolutionary Robotics and Artificial Life Lab. Alongside him, I thank the many collaborators—both junior and senior—I had the opportunity to work with over these years. Our names will be forever side-by-side in our respective bibliographies.

I thank the Ministry of Education (as well as Eric Medvet) for funding my Ph.D. scholarship and travel for international conferences.

I carried out some of the research leading to this thesis while a visiting graduate fellow at the University of Vermont, located in Burlington, USA. I was a host of the Morphology, Evolution, and Cognition Lab led by Josh Bongard, whose work on the "xenobots" back in 2020 first excited me about pursuing a Ph.D. in the field of evolutionary robotics. This visit was a great learning experience and I am grateful for the hospitality that the lab—and the Burlington community at large—collectively showed toward me, a stranger with an unusual accent.

I also would like to thank all the individuals who believed in my research and motivated me to continue it. These people may not only belong to the research community: as a matter of example, I am thankful to Matt who hosted me in Cambridge, USA when a conference was over and my flight had been canceled.

Finally, I thank my family for always being very supportive, providing me with warmth when I most needed it, and shelter at no rent.

Abstract

Harnessing the Power of Collective Intelligence: the Case Study of Voxel-based Soft Robots

Federico Pigozzi, PhD
The University of Trieste, 2023

SUPERVISOR: Dr. Eric Medvet

The field of Evolutionary Robotics (ER) is concerned with the evolution of artificial agents—robots. Albeit groundbreaking, progress in the field has stagnated to the point that a paradigm change has become necessary. In particular, current approaches lack adaptability. A solution has emerged from ideas from Collective Intelligence (CI). In CI—which has relevant examples in nature—behavior emerges from the interaction between several components. In the absence of central intelligence, collective systems are usually more adaptable.

In this thesis, we set out to harness the power of CI, focusing on the case study of simulated Voxel-based Soft Robots (VSRs): they are aggregations of homogeneous and soft cubic blocks that actuate by altering their volume. The morphologies of VSRs are intrinsically modular and thus an ideal substrate for CI; however, controllers employed until now do not take advantage of such modularity. Our results prove that we can foster and control the degree of modularity within a VSR controller, but also, by embedding one neural network inside each module with no inter-module communication, have VSRs truly driven by the CI of their modules. Furthermore, no work to date considers the case of modules that, by their local information only, must reason about the global properties of the collective. We evolve a robot to

detect its global body properties given only local information processing. Finally, we investigate different levels of adaptation in a CI system by considering how evolution and learning interact in VSRs, and, in a different study, how Hebbian learning allows VSRs to generalize better to unseen environmental conditions. Looking beyond VSRs, we propose a novel soft robot formalism that more closely resembles natural tissues and blends local with global actuation.

We believe this thesis sets the stage for further developments at the intersection between ER and CI, in the hope that collective systems will address even more of the field, thus paving the way for a new spring in the fascinating world of ER.

Table of Contents

Chapter 1: Introduction

The field of Evolutionary Robotics (ER) (Bongard, 2013) is concerned with the optimization of virtual creatures (in the following, referred also as artificial agents, or robots) while relying on Evolutionary Algorithms (EAs) (De Jong, 2006) as a search method. Starting from Sims (1994), the field has blossomed, thanks to the tendency of EAs to avoid local optima and optimize intractable spaces, like robot morphologies (Floreano and Urzelai, 2000). ER has achieved remarkable feats, like squeezing through tight spaces (Cheney et al., 2015), underwater locomotion (Corucci et al., 2018), adapting to real-world changing environments (Nygaard et al., 2018), and crossing the sim-to-real gap by designing living organisms (Kriegman et al., 2020a,c). Despite all these efforts, there is a feeling in the research community that the field has stagnated and grown weary (Eiben, 2021b), or even dead (Matthews et al., 2023); after all, most works in ER have not departed significantly from the "stick-robots" of Karl Sims, who could evolve his virtual creatures to walk, swim, and compete. There is a need for a paradigm change to disentangle ER.

On the other side of the fence, progress in deep learning has spiraled out of control: chatbots that eerily resemble humans (Thoppilan et al., 2022), world champions in several games (Schrittwieser et al., 2020), and even producers of realistic Minecraft artifacts (Sudhakaran et al., 2021). However, the amazing progress of deep learning has come to the cost of ballooning complexity in architectures and optimization algorithms (just consider the case of transformer models (Lin et al., 2022)). Many members of the public have compared AI to alchemy, starting from the late philosopher Hubert Dreyfus (1929-2017) (Dreyfus, 1965). In general, successful AI works often consist of tweaks to architectures and algorithms that have no foundation: bells-and-whistles can put technically flawed algorithms to work, while other algorithms work better with those stripped away (Hutson, 2018). Rahimi (2017) described this aspect of AI research as "alchemical": just like medieval alchemists, the field proceeds by trial-and-error, without grasping its scientific underpinnings. On top of that—or, maybe, because of that complexity—deep learning systems are often not robust when it comes to generalization. For example, Qu et al. (2020) showed that a super-human agent trained on unmodified game screens could fail by simply modifying some pixels on the screen.

Ideas from Collective Intelligence (CI) have emerged as an alternative to deep learning (Ha and Tang, 2022), blending the Artificial Life (ALife), robotics, and complex systems fields in the doing. CI emerges from the complex interaction of (relatively simple) individual components, not under a "central" intelligence. Examples abound in nature, including the foraging of ants and swarming in avian species, but we believe the metazoan *Trichoplax adhaerens* to be epitomatic (Prakash and Bull, 2022). It is a brainless animal that forages for food by self-organization of the cilia laying on its downside: they self-organize only through mechanical interactions between them, the organism's cells, and the environment. By not relying on a central intelligence, CI systems promise to achieve adaptability, including self-healing (Mordvintsev et al., 2020), and place less rigid assumptions to the environment (Tang and Ha, 2021).

In this thesis, we set out to harness the power of CI and focus on the case study of simulated Voxel-based Soft Robots (VSRs). VSRs are aggregations of mechanically identical elastic blocks: as such, they have emerged as a relevant formalism to model state-of-the-art robotic systems, e.g., soft robotics (Rus and Tolley, 2015). VSRs are appealing for investigating questions related to evolutionary biology (Kriegman et al., 2018), ALife (Cheney et al., 2014a), and designing living organisms that evolve *in vivo* (Kriegman et al., 2020a), thus bridging the sim-to-real gap (Blackiston et al., 2021).

Being aggregations of homogeneous blocks, VSRs are an ideal testbed for CI. Yet, there exist several unanswered questions in the literature that, if answered, would benefit the fields of robotics, ALife, and arguably artificial intelligence at large. We next outline the particular questions that we aim to answer.

1.1 Research questions

VSRs, being intrinsically modular, are ideal testbeds for CI. Modularity is a very desirable property on the road toward fully autonomous robotic ecosystems: in these ecosystems, robots would participate in a pipeline of automatic reconfiguration: as the task and environment change, new robots are "born" and reconfigured from past robots (Hale et al., 2019b; Buchanan et al., 2020). But, despite the tremendous achievements of VSRs, the full potential for modularity remains unexploited. Albeit VSR morphologies are intrinsically modular, controllers used until now act as abstract, disembodied processing units: disassembling such VSRs to reassemble differently, perhaps by combining modules from different VSRs, is a challenging problem. We tap into the world of *distributed* controllers (Beer et al., 1992; Yim et al., 2001; Butler and Rus, 2003), whose building blocks (e.g., neurons for a neural network) have a physical location in the robot body (e.g., a specific voxel). Still, distributed controllers employed for robots often rely on inter-module communication, limiting their flexibility. To address this dilemma, we ask two questions:

1 Can we evolve modularity in a distributed VSR controller?

2 Can we design a distributed controller that dispenses with inter-module communication?

Furthermore, few works to date consider the case of modules that, by their local information only, must reason about the global properties of the collective (Walker and Hauser, 2021; Thandiackal et al., 2021); in other words, the spatial scale of CI. We aim to fill this gap and consider the problem of *affordance detection*: understanding what actions the environment and the body afford a robot when operating with a distributed controller that relies only on local information processing. We focus on a case study of passing through an aperture in a wall and detecting whether the aperture, given the robot body, is passable or not. Hence the research question:

3 Can we evolve robots that, by local information processing only, can reason about their affordances?

If we were to deploy robots in real-world settings, we would want them to behave according to correct predictions of affordances projected by their bodies in their environment, hence the relevance of this case study.

The time scale of CI is also important; indeed, natural adaptation takes place across different time scales: evolution, development, and learning (Sipper et al., 1997). Intuitively, nature shaped animal life through an "innate" evolved component and a learned component. We embed a learning loop inside an outer evolutionary loop and ask ourselves:

4 Given the importance that the representation plays for EAs, how does it impact the speed and degree of learning?

5 Does learning—while operating at a faster time scale than evolution—help in generalizing to unseen environmental conditions?

Albeit groundbreaking, VSRs have one limitation: voxels are simulated as a set of rigid particles interconnected by soft beams (see Chapter 2). The resulting structure is soft, but cannot shape-change because of the rigid components. The question then becomes:

6 Can we simulate a soft robot capable of morphing in arbitrary shapes?

Finally, to set the stage for any subsequent analysis, we first investigate:

7 What factors impact diversity and effectiveness, two key measures of evolved populations of robots (Doncieux et al., 2015)?

As factors, we identify the three ingredients of any ER system: the representation, the EA, and the environment.

1.2 Achievements

We here present a concise overview of the achievements of this thesis:

1 We carry on an extensive study of what factors (among the representation, the EA, and the environment) impact diversity and effectiveness in VSRs, and discover that, in general, the EA and the environment matter more than the representation. While diversity has been extensively studied in ER (Cully and Demiris, 2017), we believe that our work is the first to consider multiple factors affecting the diversity of both morphology and behavior.

2 We propose a novel self-organizing, embodied neural controller for VSRs that, while evolving, spreads across the VSR body in a way that permits the emergence of modularity. In addition to being effective on a locomotion task, different levels of modularity can be achieved as well as the automatic discovery of modules.

3 We use the same neural controller inside each voxel, but with voxels communicating through implicit mechanical interactions rather than explicit inter-voxel communication.

4 We evolve robots that can detect whether their body affords them to solve a task—given an environment—on a case study of passing through an aperture of variable size; we also show that different robot morphologies can facilitate or obstruct a robot's ability to perceive such affordances.

5 We study how the learning of the robot controller inside the evolution of the robot body interplay with each other, in particular, how different ways to represent a body impact the speed and degree of learning.

6 We show how Hebbian learning in a VSR controller evolves robots that can generalize better to unseen environmental conditions (in the form of damage) than with a non-Hebbian setting.

7 We conceive and validate a novel soft robot formalism that consists of a pressurized medium enveloped by a chain of particles and beams. These robots actuate by changing the internal pressure—global actuation—and by changing the resting length of the beams—local actuation. These robots are effective at shape-changing for various tasks.

1.3 Organization of the thesis

The remainder of the thesis is organized as follows.

In Chapter 2 we present a concise historical background on the field of ER, narrowing it down to the case study of VSRs and EC as an optimization algorithm. For VSRs, we formalize their morphology and controller in detail.

In Chapter 3, we investigate what factors influence the diversity (of shape and behavior) and the effectiveness (on a locomotion task) in the joint evolution of VSR bodies and brains. This chapter—the most observatory in character—is the starting point of our journey since it provides insights that will be relevant to the other chapters.

In Chapter 4 we answer the two research questions concerning modularity in CI. We address **2** in Section 4.1 with the proposal of a self-organizing, embodied neural controller for VSRs that, in addition to evolving different degrees of modularity across the robot body, also supports the automatic discovery of modules. We address **3** in Section 4.2 with a self-attention mechanism that dispenses with any inter-module communication, allowing the VSR modules to be truly driven by their CI.

In Chapter 5 we answer the four research questions concerning the spatial and time scales of CI. We address **4** in Section 5.1 by showcasing the evolution of VSRs that can not only squeeze through the aperture of a wall but also detect when their body affords them such an action considering the size of the aperture. Our robots solve such a challenge by relying only on local tactile feedback. CI systems display adaptation not only at different spatial scales (i.e., local or global) but also at different time scales: we study how different ways of evolving (the robot body, at a slower time scale) impact the speed and degree of learning (the robot brain, at a faster time scale) in Section 5.2 (**5**). Moreover, empowering VSR brains with Hebbian learning—a specific instance of learning—evolves robots that, but virtue of the self-organization of the Hebbian synapses, generalize better to unseen environmental conditions (**6**).

In Chapter 6, we propose and validate a novel formalism for modeling and simulating soft robots, consisting of a pressurized medium enveloped by a membrane. While VSRs still rely on an internal structure of rigid elements to simulate softness, our new robots can assume a large gamut of shapes by combining local and global actuation, thus answering 1.1.

Finally, in Chapter 7 we draw conclusions and reflections.

1.4 Publications

This thesis is based on a list of co-authored publications. All of these publications are the work of the author and the contributions of the co-authors are as follows. Nearly all the publications are co-authored under the supervision of Eric Medvet, who also provided feedback and conceptualized some of the works. Andrea Ferigo, Federico Julian Camerota Verdù, Stephanie Woodman, Stefano Furlan, Giorgia Nadizar, and Marco Rochelli contributed actual material, while other senior collaborators helped supervise, conceptualize, and provide feedback.

The introduction Chapter 1 consists of unpublished material and fragments of Pigozzi (2023b) The background Chapter 2 consists of fragments of all the author's oeuvre and unpublished material. Chapter 3 on factors impacting diversity and effectiveness of VSRs is based on Pigozzi et al. (2023b), which is itself the journal extension of the conference publication Medvet et al. (2021)[1], both of them co-authored with Alberto Bartoli and Marco Rochelli. Section 4.1 on a novel self-organizing neural controller for VSRs is based on Pigozzi and Medvet (2022). Section 4.2 on controlling VSRs without relying on inter-module communication is based on Pigozzi et al. (2022)[2] co-authored with Yujin Tang and David Ha. Section 5.1 about robots that can perceive their body affordances is based on Pigozzi et al. (2023c) also co-authored with Stephanie Woodman and Rebecca Kramer-Bottiglio. This work was carried out while a visiting graduate fellow in the lab of Josh Bongard (who conceptualized it) at the University of Vermont. Section 5.2 about studying how learning and evolution interplay in modular robots is based on Pigozzi et al. (2023a) co-authored with Federico Julian Camerota Verdù. Chapter 6 about a novel soft-bodied formalism is based on Pigozzi (2023a) (to appear at the time of this writing), itself a journal extension of the conference publication Pigozzi (2022b). Finally, Chapter 7 with the concluding remarks is unpublished material.

The author also co-authored other publications (Pigozzi et al., 2021; Furlan et al., 2022; Medvet et al., 2022), not treated in this thesis because they pertain to unrelated research lines, during his Ph.D.

[1]Winner of the best paper award in the complex systems track.

[2]Runner-up for the best paper award in the complex systems track.

Chapter 2: Background

In this section, we motivate the field of ER in Section 2.1, before delving into the case study of VSRs in Section 2.2 and motivating EAs as a search method in Section 2.3.

2.1 Evolutionary Robotics

For a robot, one can optimize either the morphology, the controller, or both. In general, the choice of the optimization algorithm (i.e., the search method) is free, but relying on Evolutionary Computation (EC) gives rise to ER (Nolfi and Floreano, 2000).

The *embodied cognition* paradigm (Pfeifer and Bongard, 2006) posits that the intelligence of an agent (e.g., a robot) emerges from the complex interaction between the body, the brain, and the environment. Intelligence is not only rational but it is embedded in the body also (Paul, 2006). Many organisms can perform complex computations through their bodies only. As a matter of example, some animals (e.g., salamanders and flatworms) are capable of regrowing amputated limbs, and the way they achieve such regeneration is by biological processes localized in the severed portion of their body (Joven et al., 2019). The environment plays a key role as well. For example, it is well-known how different environmental conditions can alter the development of an individual (Miras et al., 2020b).

Similarly, robots are usually modeled with the classic control loop: there are two entities, the agent and the environment. Most of the time, they interact using interfaces: the agent can collect information from the environment using sensors while applying modifications to the environment using actuators.

Embodied cognition applies to “natural” as well as “artificial” agents, namely, robots. When designing a robot, two key aspects must be planned: the controller and the morphology. The former can implement any function outputting actuation values for the actuators; the latter dictates how the body of the robot shall be built and assembled. Ideally, the two aspects are interlocked: a given controller might not be well-adapted to the target morphology, and vice versa (Eiben and Hart, 2020a).

Finally, to be endowed with agency, an agent must display some attachment to a specific goal, or task. Examples of tasks are locomoting on a surface, navigating a maze, or grasping a Rubik's cube (Bhatia et al., 2021). Tasks may not only be of an engineering nature: investigating questions related to evolutionary biology or embodied cognition are other examples (Kriegman, 2019). To find the robot that can best solve a task, designers have resorted to optimization algorithms. Optimization can be performed on the controller (for a fixed morphology), on the morphology (for a fixed controller), or both of them at the same time (joint optimization). Different optimization approaches have historically been employed, particularly EC (Floreano and Urzelai, 2000), due to its flexibility in the choice of representation (see Section 2.3).

To conclude, here is a very gentle history of ER:

1994 Sims (1994) evolved virtual creatures for terrestrial and underwater locomotion. It is the first published work on ER.

2000 Lipson and Pollack (2000) proved able to manufacture robots evolved with Sims' simulator. It received a lot of media coverage and introduced ER to the general public.

2011 Hiller and Lipson (2011) modeled and manufactured a new type of robotic agent, voxel-based and soft.

2013 Cheney et al. (2013) applied a neuroevolutionary algorithm to successfully evolve the morphology of multi-material voxel-based soft robots.

2020 Kriegman et al. (2020a) synthesized living organisms (the "xenobots") from cells of *Xenopus laevis* (a frog species), after having evolved their morphology *in silico*. It spurred media attention and may have catapulted robotics into a new age of interfacing with synthetic biology.

Albeit self-contained, this introduction to ER does not include an exhaustive treatment of the state-of-the-art relevant for this thesis. It instead appears in the introduction and related works sections of the respective chapters for the ease of consultation. This introduction to ER is thus a primer on the field.

2.2 Voxel-based Soft Robots

Voxel-based Soft Robots (VSRs) are a kind of modular robots composed as aggregations of elastic cubic blocks (*voxels*), made of soft material. Each voxel acts by contracting or expanding its volume and it is the overall symphony of volume contractions and expansions that allows for the emergence of behavior at the robot level. VSRs were first formalized in Hiller and Lipson (2012), together with a fabrication method. In this work, we consider a 2D variant of simulated (in discrete time and continuous space) VSRs (Medvet et al., 2020b). While disregarding one dimension makes these simulated VSRs less realistic, it also eases the optimization of VSR design, thanks to the smaller search space. We remark, however, that the representations and the algorithms adopted in this thesis are easily portable to the 3D setting.

In the following, we outline the characteristics of VSRs relevant to this thesis and refer the reader to (Medvet et al., 2020b) for more details. A VSR is defined by its *morphology* (i.e., the body) and its *controller* (i.e., the brain). The former is in turn built with a *shape*, dictating how many voxels the robot is constructed with and how they are arranged in a 2D grid, and a *sensory apparatus*, telling what are the sensors and how they are placed over the robot body. Sensors can provide information about the external environment and the robot itself, turning the controller into a closed-loop system. The controller is in charge of determining how the area of each voxel varies over time.

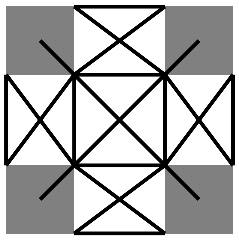


Figure 2.1: The mechanical model of a voxel in our simulation. Gray squares are rigid bodies and black wiggly strings are springs.

2.2.1 VSR morphology

The morphology of a VSR describes how the voxels, i.e., deformable squares of side length $l = 3\,\text{cm}$, are arranged in a grid topology of size $w \times h$. We model each voxel as the assembly of spring-damper systems, masses, and distance constraints (Medvet et al., 2020b) and is rigidly connected to its four adjacent voxels (if present). The springs, by oscillating, allow the voxel to alter its area and thus endow it with softness, while the bodies endow the voxel with a mass and a sense of an "embodiment", including the ability to react to forces and collide with other objects. We set the same parameters for the components of each voxel as the default ones; as a result, all the voxels share the same mechanical properties. We present a schematic view of our voxel model in Section 2.2.1.

Over time, the voxels change their area according to (a) external forces acting on the voxel (e.g., other voxels and bodies like the ground) and (b) a control signal dictated by the controller. The latter produces a contraction/expansion force that is modeled in the simulation as an instantaneous change in the resting length of the spring-damper systems of the voxel. The length change is linearly dependent on an *actuation value* residing in $[-1, 1]$, -1 being the greatest possible expansion and 1 being the greatest possible contraction. The controller sets the actuation value for each voxel, at every time step of the simulation.

A VSR can be equipped with sensors, and each voxel can have one or more sensors. A sensor outputs, for every time step, a *sensor reading* $\boldsymbol{s} \in \mathbb{R}^m$, where m is the dimensionality of the sensor type. In this study, we equip VSRs with four different types of sensors. Area sensors sense the ratio between the current area of the voxel and its resting area ($m = 1$). Touch sensors sense whether the voxel is currently touching another body different from the enclosing VSR (e.g., the ground) or not, and output a value of 1 or 0, respectively ($m = 1$). Velocity sensors sense the speed of the center of mass of the voxel along the x- and y-directions ($m = 2$). Lidar sensors sense the distance to the closest objects along a predefined set of directions. Precisely, for each direction, a lidar sensor measures the distance between the voxel center of mass and the closest object in that direction, clipping it to d. If no object is present at all, the sensor reading is set to d. We used $d = 10\,\mathrm{cm}$ and the following directions with respect to the positive x-axis: $-\frac{1}{4}\pi$, $-\frac{1}{8}\pi$, 0, $\frac{1}{8}\pi$, $\frac{1}{4}\pi$ (so $m = 5$). Sensor readings undergo a soft normalization, with tanh function and rescaling, to ensure the output is in $[0, 1]^m$. After normalization, every sensor reading $\boldsymbol{s}$ is perturbed into $\boldsymbol{s}' = \boldsymbol{s} + \boldsymbol{\nu}$, with $\boldsymbol{\nu} = \{\nu_i\}_i \in \mathbb{R}^m$ and $\nu_i \sim \mathcal{N}(0, 0.01)$ being additive Gaussian noise of mean 0 and variance 0.01. The purpose of this transformation is to simulate real-world sensor noise. Figure 2.2 shows two example VSRs simulated using our software.

2.2.2 VSR controller

Let n be the number of voxels of the VSR and let $\boldsymbol{r}^{(k)} = [s_1 \; s_2 \; \dots]$ be the concatenation of sensor readings for all the VSR sensors at time step k, i.e., at time $t = k\Delta t$, where Δt is the interval between two simulation time steps. The controller is closed-loop with input $\boldsymbol{r}^{(k)}$ and output $\boldsymbol{a}^{(k)} \in [-1, 1]^n$.

Previous works dealing with VSRs employed different approaches in instantiating this general definition, with different degrees of complexity. Some works, such as, e.g., (Cheney et al., 2013), employed an open-loop controller (a phase controller), whose output depends only on k, i.e., it does not exploit the information coming from the sensors. In other cases, as, e.g., (Talamini et al., 2019), the controller does not have a memory and its output depends only on the current input. In the latter scenario, the controller can be modeled as a function $\boldsymbol{a}^{(k)} = f\left(\boldsymbol{r}^{(k)}\right)$, or simply

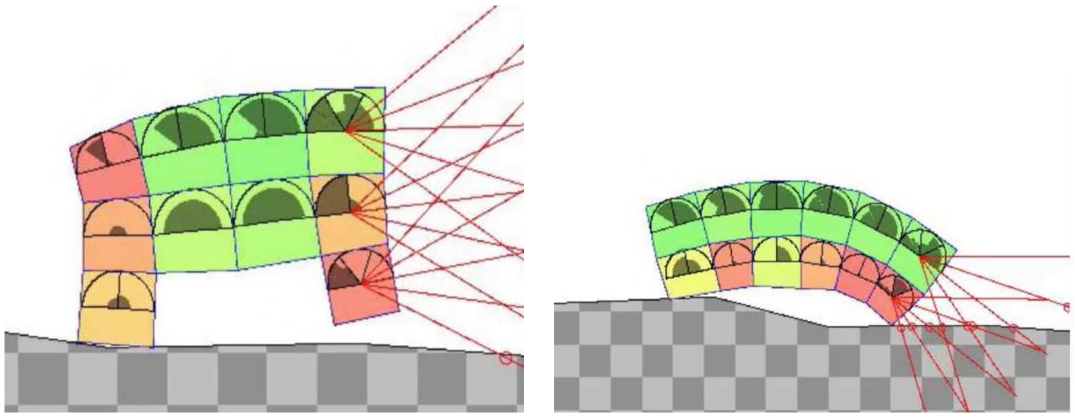

(a) Biped (b) Worm

Figure 2.2: A biped and a worm example morphologies, taken at a snapshot of the simulation. Each square is a voxel. The color represents the ratio between its current area and its rest area: red stands for contraction, green for expansion, and yellow for no change. The semi-circular sectors drawn at the center of each voxel (if present) encode the sensor readings, and are partitioned into subsectors according to the number of sensors; subsectors are further partitioned according to the sensor dimensionality m. The red lines depict the rays of the lidar system.

$\boldsymbol{a} = f(\boldsymbol{r})$. In this condition, and if the function $f : \mathbb{R}^p \to \mathbb{R}^n$ can be parametrized with a numerical vector $\boldsymbol{\theta} \in \mathbb{R}^q$, then the problem of finding a good controller given a morphology and a task can be cast as a numerical optimization problem.

2.2.2.1 Distributed VSR controller

For the sake of this study, we resort to the distributed neural controller proposed in Medvet et al. (2020a) that facilitates the study of CI. It consists of some fully connected, feed-forward Artificial Neural Networks (ANNs), one for every voxel, and operates as follows.

At time step $t = k\Delta t$, each ANN (i) receives as input the local sensor readings and the $4n_{\text{comm}}$ *communication values* generated by the four adjacent ANNs at the previous time step and (ii) outputs the local actuation value and $4n_{\text{comm}}$ communication values to be fed to the adjacent ANNs at the next time step. Formally, each

ANN works as follows:

$$\left[a^{(k)}\ \boldsymbol{o}_N^{(k)}\ \boldsymbol{o}_E^{(k)}\ \boldsymbol{o}_S^{(k)}\ \boldsymbol{o}_W^{(k)}\right] = \text{ANN}\left(\left[\boldsymbol{s}_i^{(k)}\ \boldsymbol{i}_N^{(k-1)}\ \boldsymbol{i}_E^{(k-1)}\ \boldsymbol{i}_S^{(k-1)}\ \boldsymbol{i}_W^{(k-1)}\right]\right), \qquad (2.1)$$

where $a^{(k)} \in \mathbb{R}$ is the local actuation value, $\boldsymbol{o}_N^{(k)} \in \mathbb{R}^{n_\text{comm}}$ is the vector of communication values directed to the voxel above (similarly for E, S, W), $\boldsymbol{s}_i^{(k)} \in [0,1]^4$ is the concatenation of the local sensor readings, and $\boldsymbol{i}_N^{(k-1)} \in \mathbb{R}^{n_\text{comm}}$ is the vector of communication values coming from the voxel above and been generated at the previous time step (similarly for E, S, W). If one of the neighbors is absent (e.g., if the voxel lies on the boundary of the morphology), we set $\boldsymbol{i}_N^{(k-1)}$ (or E, S, W) to a zero-vector $\mathbf{0}$ of the appropriate size. Similarly, we use a zero-vector as input communication values for all the voxels at the very first time step.

Despite its simplicity, this form of controller may result in interesting and variegate behaviors, since the interconnections between voxel ANNs make the overall architecture recurrent (Rumelhart et al., 1986), endowing the system with a form of "memory": therefore, there is a further dynamics introduced by the recurrent ANN that interacts with the dynamics induced by the mechanical model of the soft body (where the spring-and-hamper systems hold a form of "memory" too).

After some preliminary experiments and by taking into account the findings of Medvet et al. (2020a), we use $n_\text{comm} = 1$, no hidden layers, and tanh as the activation function for all the neurons—the latter guarantees that all communication values and the actuation value are in $[-1, 1]$.

2.3 Evolutionary Computation (EC)

EC is a family of population-based optimization algorithms (Luke, 2009); a population of candidate solutions is kept and evolved in iterations. At each iteration (or generation), the population is updated by applying the selection and genetic (or search) operators. An objective function, known in the jargon as fitness, drives the optimization process. Borrowing from the principles of Darwinian evolution (Darwin, 1859), selection operators decide how unfit solutions (or individuals) should be discarded from the population (on the contrary, fit solutions carry on to the next iteration). Genetic operators stochastically perturb (generally, through mutation or recombination) fit individuals to produce new candidate individuals. The combination of selection and genetic operators allows us to ascend in the fitness landscape. As a final ingredient, an initial population of individuals should be initialized at the beginning of evolution, usually randomly.

Given the random initialization and the stochastic search operators, EAs are usually regarded *stochastic* optimization algorithms. It is by their stochasticity that EAs can potentially avoid being trapped in local minima and uncover novel and interesting solutions. These are desirable properties in robotics, where:

(i) a large number of degrees of freedom makes robotic systems very hard to tune for a human engineer (Rus and Tolley, 2015);

(ii) fitness landscapes are usually very rugged and hard to explore (Smith et al., 2002);

(iii) diversity of solutions is also important (Cully et al., 2015).

Last but not least, by placing no assumptions on the fitness function to be optimized, EAs are effective when there is no analytical formulation for the objective (consider, e.g., the optimization of a robot's morphology).

2.3.1 Evolution of Things

Textbook EC performs search through the genetic operators in a space called the genotype space; genotypes are then mapped to the phenotype space as phenotypes, which, in turn, are mapped to the fitness space. In other words, the phenotypes are the solutions we are interested in, while genotypes make up the space of the actual search. Albeit having been valid for decades, this paradigm of textbook EC fails to capture all the facets of a complex system (like a robotic one), whose behavior cannot be inferred from the individual properties of the components alone, but especially from their interaction (Sayama, 2015). To address this shortcoming, Eiben and Smith (2015) have proposed a fourth space, the behavior space, sitting in the middle between phenotype and fitness space, and labeled the resulting framework Evolution of Things. In the Evolution of Things, phenotypes (like robots, of any kind of software or hardware) are *situated* entities that exist in a given environment. As such, by interacting with the environment, they produce behavior that cannot be predicted from the mere phenotype, and the fitness must computed onto the behavior. The Evolution of Things is the branch of EC where this thesis fits.

Chapter 3: Factors impacting diversity and effectiveness of evolved modular robots

In many natural environments, different forms of living organisms accomplish the same task while being diverse in shape and behavior. This biodiversity is what makes life capable of adapting to disrupting changes. Being able to reproduce biodiversity in artificial agents, while still optimizing them for a particular task, might increase their applicability to scenarios where human response to unexpected changes is not possible. As a first step in this thesis, we conduct an extensive study on what factors impact the diversity and effectiveness of evolved modular robots.

In this chapter, we answer to question **7**. We evolve, at the same time, the morphology and controller of VSRs for the task of locomotion. As mentioned in Section 2.2, VSRs grant great freedom in the design of both morphology and controller and are hence promising in terms of "biodiversity". We investigate experimentally whether three key factors—representation, EA, and environment—impact the emergence of biodiversity and if this occurs at the expense of effectiveness (by representation, we mean genetic encoding). We also devise an automatic machine learning pipeline for systematically characterizing the morphology and behavior of robots resulting from the evolution process. We classify the robots into species and then measure biodiversity in populations of robots evolved in many conditions resulting from the combination of different morphology representations, controller representations, EAs, and environments. The experimental results suggest that, in general, EA and the environment matter more than representation.

3.1 Introduction

The vast majority of works in robotics aim to find one robot design that fits well for a given task in a given environment. One intrinsic limitation of this approach is that, when the environmental conditions change, the found design might become less effective, and possibly ineffective, making all the deployed robots immediately useless. That design and the approach that produced it are intrinsically not adaptable. Conversely, in nature, several different designs exist at the same time that fit well for a given task (e.g., locomotion, food harvesting, and reproduction) with varying environmental conditions. Diversity of designs, i.e., *biodiversity*, is hence the way nature achieves adaptation. Through biodiversity, natural evolution made life robust to disruptive changes, by filling a variety of ecological niches with different species. Indeed, biodiversity is so valuable that it has to be preserved to protect life itself (Tilman et al., 2017), as well as to increase the stability of ecosystems (Arese Lucini et al., 2020). Diversity plays a fundamental role even in other settings, e.g., in geology (Schrodt et al., 2019), economics (O'Sullivan and Sheffrin, 2003), culture and politics (Young, 1979).

Obtaining diversity while evolving robots is not an easy endeavor, though. First, comprehension, measurement, and promotion of diversity are important challenges themselves in the broader field of EC (Squillero and Tonda, 2016; Črepinšek et al., 2013), from which ER borrows the optimization techniques. Second, the characterization of diversity in ER and a deep understanding of which factors favor or impede it are still open issues (Silva et al., 2016). The complexity of the robot-environment interplay makes it hard to obtain useful diversity, i.e., the diversity that does not affect effectiveness.

In this chapter, we study the impact of three factors on the diversity and effectiveness of populations of evolved VSRs: the solution representation, the EA, and the environment. We investigate whether the open-ended design freedom of VSRs can also foster diversity. To this end, we deal with the *joint* evolution of both morphology and controller of the robot, a problematic task in the ER community (Lipson et al., 2016). We consider two representations for the controller and two representations for the morphology that differ in their expressiveness and hence exhibit different potentials for diversity while allowing concurrent evolution of VSRs morphology and controller. We also consider four EAs that are radically different in how they deal with diversity and, finally, three environments with different degrees of difficulty. Among these, we propose and assess a novel EA that promotes diversity through speciation, where individuals are partitioned into species based on a few morphology and behavior descriptors that we designed for the case of VSRs doing locomotion.

A key contribution of this chapter is an automatic pipeline for *systematically* analyzing a very large number of VSRs (hundreds of thousands), while still looking at them with the human eye as we do when associating living organisms with a specific *species*. In doing so, we rely on Machine Learning (ML) to automatically assign species to VSRs. We build an ML pipeline for classifying VSRs into species according to morphology and behavior descriptors extracted from simulations of VSRs that perform the task of locomotion. Then, we use the relative abundance of predicted species as a measure of diversity for a population of VSRs, using the well-established Simpson index (Simpson, 1949).

While there have been several studies addressing diversity in EC (Mouret and Doncieux, 2012; Miras et al., 2020b; Nordmoen et al., 2021), with some of them considering the domain of robots, we believe that our work is the first to consider multiple factors affecting the diversity of both morphology and behavior. Moreover, we analyze those factors based on a notion of diversity that exploits humans' ability to discriminate between different approaches to locomotion and hence facilitates comprehension. We believe our study may help future designers of evolvable robotic ecosystems in prioritizing different factors in terms of their impact on the diversity and effectiveness of evolved robots. Namely, we found that the environment and the EA seem to have a greater impact on diversity than the representation. Hence, these two factors should likely be the ones on which a designer should focus more.

3.2 Related works

Earlier works examined diversity in ER (Mouret and Doncieux, 2012; Auerbach and Bongard, 2014; Samuelsen and Glette, 2014; Miras et al., 2020b; De Carlo et al., 2020; Nordmoen et al., 2021). Our work shares with them the general methodology, i.e., investigating the factors that may have an impact on evolution but differ in either the kind of robots considered or the factors themselves. Most of those works focused mainly on the impact of the environment (Auerbach and Bongard, 2014; Miras et al., 2020b; Gupta et al., 2021), while here we consider also the controller representation, the morphology representation, and the EA. The studies that are most similar to ours are (De Carlo et al., 2020) and (Mouret and Doncieux, 2012). The former investigates whether a mechanism of artificial speciation can favor morphological diversity. We also devise an EA that employs speciation and, in this respect, both the cited work and our approach were inspired by NEAT (Stanley and Miikkulainen, 2002). However, differently from (De Carlo et al., 2020), we (i) also consider the diversity of behavior, (ii) work with a more expressive kind of robots, VSRs, and (iii) analyze the joint impact of representation, EA, and environment. Mouret and Doncieux (2012) conducted an empirical study for comparing approaches for encouraging the diversity of the behavior of evolved robots. The authors focused mainly on the EA and on the measure of similarity on which to build diversity promotion. Different from the cited work, we (i) also consider the diversity of the robot morphologies and (ii) take the representation of solutions, i.e., how to map a genotype into a pair morphology–controller, as a factor potentially affecting diversity. We do not explicitly compare different similarity measures; yet, in the novel simple EA we propose for promoting diversity through speciation, we experiment with three different ways of measuring similarity between pairs of robots.

The pursuit of diversity has become more and more important in the EC community (Cully and Demiris, 2017). Proposals in this respect include novelty search (Lehman and Stanley, 2008), novelty search with local competition (Lehman and Stanley, 2011), and quality-diversity algorithms (Cully and Demiris, 2017). In a recent study, Nordmoen et al. (2021) showed that MAP-Elites, a form of quality-diversity optimization, is particularly suitable for exploiting the potential for diversity of modular (rigid) robots. The authors showed experimentally that populations of robots that evolved with MAP-Elites are more successful when transferred to new environments than those that evolved with other EAs. Moreover, they specifically spotted a strong correlation between the diversity of earlier populations and the effectiveness of locomotion in new environments. In a previous study, Tarapore et al. (2016) found that MAP-Elites is sensible to the representation of the controller in evolutionary robotics applications: apparently, the lower locality of indirect, generative representations makes this EA less effective than with direct representations. Interestingly, MAP-Elites has also been recently found particularly effective in the simultaneous evolution of morphology and controller of VSRs (Ferigo et al., 2022b), with both indirect and direct representations of the controller. Overall, these results further motivate our aim of investigating the factors, beyond the EA, that favor or disfavor diversity in modular robotics.

Last but not least, our work fits into a relevant body of literature in ER concerning the joint evolution of morphology and controller. Past studies have either employed directed acyclic graphs (Sims, 1994), L-systems (Hornby et al., 2001), gene regulatory networks (Joachimczak et al., 2016), direct encodings (Pagliuca and Nolfi, 2020), or relied on more complex solutions as co-evolution (Cheney et al., 2018), and evolutionary reinforcement learning (Gupta et al., 2021). In most of the cited works, the key ingredient for achieving the concurrent optimization of morphology and controller is the representation, i.e., how to encode in a genotype the information needed for describing both the morphology and the controller of the robot. While, in principle, some of the approaches mentioned above could be ported to the case of VSRs, in this study we "only" focus on four representations resulting from the combination of two for the morphology and two for the controller. Since we are interested in investigating the impact of representation (and its interplay with EA and environment) on diversity, we choose the representations for the morphology and the controller considering their compactness vs. expressiveness trade-off.

The intrinsic hardness of jointly evolving robot morphology and controller has been discussed by Lipson et al. (2016), using precisely the case study of VSRs. The reason for such difficulty rests on the *embodied cognition paradigm* (Pfeifer and Bongard, 2006), which posits that intelligence emerges from the interaction between the controller (brain), the morphology (body), and the environment: brains evolve to fit a particular body and variations in the body are likely to cause mismatch (Eiben and Hart, 2020b). On the other hand, allowing the concurrent optimization of both morphology and controller of the robots makes their optimization more difficult as diversity seems to vanish quickly (Pagliuca and Nolfi, 2020). An important contribution of our work is showing that it is possible to jointly evolve the morphology and the controller of VSRs that are effective in the task of locomotion, while also being diverse.

3.3 Materials and Methods

We discuss the methods adopted for this chapter

3.3.1 Measuring biodiversity

We take inspiration from the natural sciences and build our definition of biodiversity of populations of VSRs on the concept of *species*. A species groups together individuals sharing the same phenotypic traits[1]. Once we can associate each VSR with a species, we measure the diversity of a population of VSRs as the variety of species of its individuals.

For VSRs, since we deal with the joint evolution of morphology and controller,

[1]Our definition of species takes inspiration from biology, but does not reflect it: in biology, a species is a group of organisms in which any two individuals can mate to produce fertile offspring.

we use both kinds of traits for defining species. That is, we define a species as a pair of morphology species (based on morphology traits) and controller species (based on controller traits). Aiming at defining species that are based on *observable* traits, we consider the behavior, rather than the controller, of VSRs: while we do acknowledge that the behavior is not determined by the controller alone since it depends also on the morphology and the environment, we believe that the behavioral traits we considered carry a significant amount of information about the underlying controller.

To have species that can be observed and discerned by humans, as it happens for living organisms, we *phenotypically classify* morphology species and behavior species by human categorization. Although it is just one of the ways that modern biology uses to define species (Mayden, 1997), we believe this approach aligns with our objective of discovering species as seen by the human eye (rather than, e.g., according to genetic compatibility). However, since we evolve a large number of VSRs (in the order of hundreds of thousands), species classification by human inspection alone is not feasible. For this reason, we use ML for automatically determining the species of a VSR, a common approach also for determining the species of plants (Franklin and Ahmed, 2018) and animals (Tabak et al., 2019).

We rely on *supervised* ML for species classification: we collect a few example cases, each one consisting of a VSR associated with a morphology class and a behavior class (both manually assigned); then, we learn two classification models, based on those examples, for associating any other VSR with one morphology and one behavior class. To use supervised ML, we first define the classes and a criterion for selecting a training set for manually assigning classes. We also define a set of descriptors (features, in ML terminology) useful for characterizing the morphology and the behavior of VSRs and to be fed to supervised learning techniques. In the following subsections, we describe each of the steps in detail.

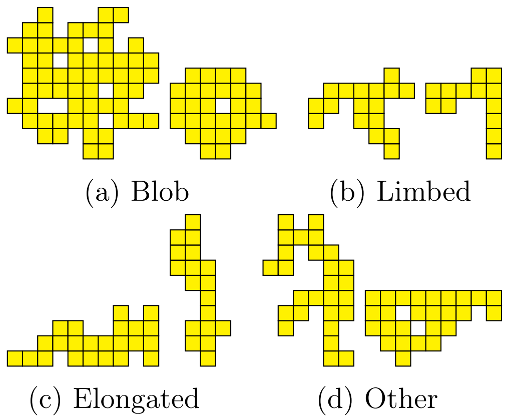


Figure 3.1: Sample morphologies for the four morphology classes.

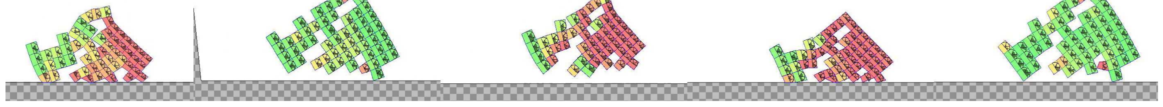
3.3.2 Species classes definition and manual annotation

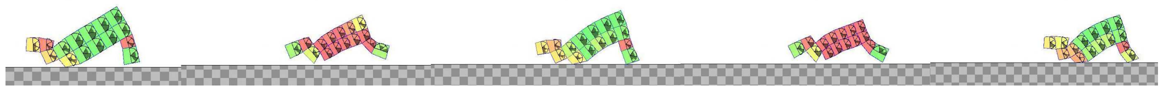
3.3.2.1 Classes definition

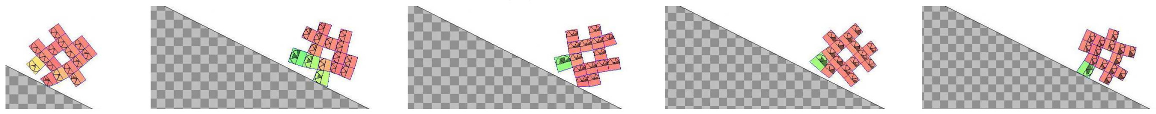
For deciding how many classes to use for morphology and behavior we observed a large number of videos of VSRs performing locomotion on three different terrains (flat, downhill, and uphill, see Section 3.4.3) and obtained through evolution with six different EAs (see Section 3.3.3), two different controller representations (see Section 3.3.5.1), and two different morphology representations (see Section 3.3.5.1).

Based on these observations, we define four classes of morphology. *Blob* is compact and roundish, with no clear direction of development. *Limbed* has extrusions that might resemble limbs. *Elongated* is compact but with a clear direction of development. *Other* is a miscellaneous class for the VSRs that cannot be classified into one of the other classes. Figure 3.1 shows sample morphologies for the four classes.

Concerning the behavior, we define four classes. *Jumping* VSRs alternate between touching the ground and lifting their body in the air. *Walking* VSRs cyclically alternate the parts of the body that touch the ground. *Rolling* VSRs roll on themselves. Finally, *Other* VSRs cannot be classified into one of the other labels, and thus fall in a miscellaneous class. Figure 3.2 show time-lapse images for the movement


(a) Jumping


(b) Walking


(c) Rolling

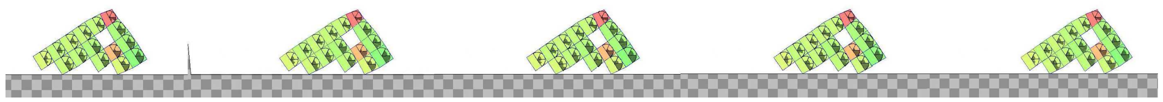

(d) Other

Figure 3.2: Time-lapse showing locomotion for a sample VSR for each behavior class. Figure 3.2d moves forward vibrating at a very high frequency.

of a sample VSR for each behavior class. We also provide a video of those VSRs at `https://youtu.be/tuD8scZ88Xc`.

Since we define four classes for each one between morphology and behavior, and being the species the combination of the two, it follows that we define an overall number of 16 different species.

3.3.2.2 Manual annotation

Having defined classes for morphology and behavior as illustrated in the previous section, we need to collect a training set of examples suitable for learning two classifiers that can automatically associate a previously unseen VSR with the corresponding classes.

Since we want to measure the diversity of populations of VSRs subjected to evolution, the training set should consist of individuals that uniformly cover the spaces of morphologies and behaviors that are likely to arise from evolution. However, a large portion of VSRs being generated during evolution (in particular at its early stage) perform poorly in the task of locomotion and are hence very difficult to associate with behavior class. For this reason, instead of simply selecting a random sample of all the individuals observed during the many evolutions we run (see Section 3.4.1, Section 3.4.2, and Section 3.4.3), we proceeded as follows.

First, we define three descriptors of the VSR morphology:

1. *Number of voxels* d_{num}, i.e., the number of voxels in the VSR.

2. *Elongation* d_{elong}, i.e., how stretched is the morphology in its direction of maximum development. For computing d_{elong} we first consider the smallest ellipse that encloses the VSR morphology; then we compute the ratio of the focal distance (distance between focal points) of the ellipse over the major axis length (Burger et al., 2009). It follows that $d_{\text{elong}} \in [0, 1[$, with $d_{\text{elong}} = 0$ for perfectly "even" morphologies (e.g., a circle or square).

3. *Compactness* d_{compact}, i.e., the ratio between the number of voxels of the VSR and those of the convex hull enclosing the morphology. The intuition is that d_{compact} is higher for morphologies with few concavities. Considering that the convex hull of a 2D shape has an area that is always greater or equal to the area of the shape itself, it follows that $d_{\text{compact}} \in]0, 1]$, with $d_{\text{compact}} = 1$ for perfectly compact morphologies, i.e., morphologies with no concavities.

Then, we computed the values of d_{num}, d_{elong}, and d_{compact} for each VSR and partitioned the pool into 4 equal-size bins per descriptor, thus partitioning the pool of VSRs into $4 \times 4 \times 4 = 64$ bins. Next, we removed from all the 64 bins the VSRs that perform poorly in locomotion, i.e., those whose speed is lower than $2\,\text{m/s}$. After preliminary experiments, we found $2\,\text{m/s}$ to reasonably discriminate poor performing VSRs. Finally, we selected a subset of VSRs to be manually inspected for associating

Table 3.1: Distribution of the manually assigned labels for the morphology and behavior classes on the training set.

	Blob	Limbed	Elongated	Other	Total
Jumping	0	1	291	25	317
Walking	67	82	64	174	388
Rolling	8	48	0	50	106
Other	82	93	132	437	744
Total	157	224	487	687	1555

a morphology and a behavior class by taking randomly 25 VSRs out of each bin and adding 75 slow VSRs taken randomly from the discarded VSRs. This way, we assembled a set of $25 \cdot 64 + 75 = 1675$ VSRs.

We finally had a human operator inspect the VSRs, by looking at their unlabeled simulation videos, and assign VSRs to morphology and behavior classes until collecting at least 100 labels per class. We ended up with a training set of 1555 labeled VSRs, distributed among classes as summarized in Table 3.1.

The figures of Table 3.1 might suggest that our manual annotation procedure is affected by a human bias: human annotators tended to assign to the Other classes all the samples they were not able to assign to the identifiable cases (both for the morphology and behavior). As a result, the dataset is slightly unbalanced and, as we will discuss in Section 3.4, this impacted later analyses. On the other hand, we believe that this bias is positive, in the sense that human operators did observe the robots in the context of a race for locomotion. In the end, we are more interested in distinguishing between a crawling and a jumping artificial organism rather than a multitude of ways of being idle.

3.3.3 Features and learning

As in any ML application, two key design choices concern the features to extract for describing the observations (here, simulations of VSRs) and the learning technique. Since we aim to classify the morphology and the behavior separately, we define different features for the two classification tasks. We describe them in the next subsections.

3.3.3.1 Morphology features

Since the VSR morphology is an arrangement of voxels in a 2D grid, we could extract the features concerning the morphology directly from the grid, e.g., the descriptors described in Section 3.3.2. However, the grid is a static description of the VSR and does not capture the robot as seen during its "life", i.e., as it could be seen by an external observer looking at the VSR while it does locomotion. For this reason, we define the morphology features based on the idea of the *dynamic pose* of the VSR. Intuitively, the dynamic pose can be regarded as a "long-exposure photograph" of the VSR during the simulation. We construct such a pose as follows.

Let a *snapshot* be the complete description of a time step of VSR simulation, i.e., it comprises the ground and every voxel of the VSR. Let S be a sequence of such snapshots, spanning an entire VSR simulation. For each snapshot, we determine the *minimal bounding square* $(x_0, y_0, x_0 + w, y_0 + l)$ around the VSR, that is, the smallest square parallel to the x-axis that completely encloses the VSR. Then, we partition the minimal bounding square in 16×16 inner squares with side lengths $\frac{w}{16}$ $\frac{l}{16}$ and build a matrix $\boldsymbol{d} \in \{0,1\}^{16\times 16}$ where the element $d_{i,j}$ is 1 if and only if the corresponding inner square $(x_0+(i-1)\frac{w}{16}, y_0+(j-1)\frac{l}{16}, x_0+i\frac{w}{16}, y_0+j\frac{l}{16})$ is occupied by the VSR for at least half of the area. Finally, we compute the dynamic pose as the element-wise mode of the matrices computed for the snapshots in S.

We use the 256 values of the dynamic pose of a VSR as a feature vector for its morphology, obtaining $\boldsymbol{f}_{\text{morph}} \in \{0,1\}^{256}$.

3.3.3.2 Behavior features

Since we deal with robots performing the task of locomotion, we define two groups of features that capture the VSR behavior while in locomotion, i.e., its *gait*, from two different points of view: the movement of the center of the VSR over the time and the way the VSR touches the ground while moving. We denote by $\boldsymbol{f}_\text{center}$ and $\boldsymbol{f}_\text{footprints}$ the two corresponding feature vectors, and by $\boldsymbol{f}_\text{behavior}$ their concatenation. We construct these vectors as follows.

Center movement Concerning the features describing the movement of the center of the VSR, let S be a sequence of snapshots; we extract from S the discrete signals of the $x-$ and $y-$coordinate of the center of mass of the VSR. Then, we consider the signals of the first differences and compute their Fast Fourier Transform (FFT) (Cooley and Tukey, 1965), to represent the signal in the frequency domain. Subsequently, we take the magnitude of the two FFTs, filter out the components corresponding to frequencies greater than f_max (by taking into account the simulation time step Δt), and re-sample the remaining components to have n_freq components for each one of the two axes.

We use as feature vector $\boldsymbol{f}_\text{center}$ the concatenation of the two resulting vectors of magnitudes $\boldsymbol{f}_\text{center} = \left[\boldsymbol{f}_{\text{center},x}\ \boldsymbol{f}_{\text{center},y}\right] \in {\mathbb{R}^+}^{2n_\text{freq}}$, with $\mathbb{R}^+ = [0, +\infty[$. After preliminary experiments and leveraging our expertise, we set $f_\text{max} = 10\,\text{Hz}$ and $n_\text{freq} = 100$.

Footprints Concerning the features describing how the VSR touches the ground, we build a definition based on the concept of *footprint*. Given a snapshot, we consider the projection $[x_0, x_0 + w]$ of the minimal bounding square on the x-axis and we partition it into 8 equally-sized segments. After preliminary experiments, we found 8 to be an adequate number of segments. Then we build the footprint of the VSR in that snapshot as a binary sequence $\boldsymbol{m} \in \{0,1\}^8$, where the element m_i is 1 if and only if the VSR is touching the ground for at least half of the corresponding segment $[x_0 + (i-1)\frac{w}{8}, x_0 + i\frac{w}{8}]$.

Given a sequence S of snapshots, we follow the following procedure to determine the set of footprint features. (1) We split S in a sequence $(S_1, S_2, \dots)$ of non-overlapping subsequences, each one corresponding to an interval of $\Delta t_{\text{footprint}}$ simulated time (we set $\Delta t_{\text{footprint}} = 0.5\,\text{s}$ after some preliminary experiments). (2) We build the sequence $M = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \dots\}$ of footprints where each $\boldsymbol{m}_i$ is obtained as the element-wise mode of the footprints computed from snapshots in S_i. (3) We consider all the non-overlapping n-grams of footprints in M, with $2 \leq n \leq 10$, that occur at least twice and compute the overall duration of each n-gram, computed as the product between its number of occurrences and its duration. (4) We select as the *main footprint n-gram* $M^\star$ the n-gram with the greatest overall duration. (5) We compute the following features for $M^\star$: duration $|M^\star|\Delta t_{\text{footprint}}$, average touch area $\frac{1}{|M^\star|}\frac{1}{8}\sum_{\boldsymbol{m} \in M^\star}\sum_{i=1}^{i=8} m_i$, number of occurrences of $M^\star$ in M, mode $\Delta t_{M^\star}$ of the intervals between subsequent occurrences of $M^\star$, rate of intervals that are equals to the mode.

We use as feature vector the five features computed for the main footprint n-gram $M^\star$, i.e., $\boldsymbol{f}_{\text{footprint}} \in \mathbb{R}^{+5}$.

3.3.3.3 Learning technique

We rely on Random Forest (Breiman, 2001) as classifiers for the morphology and behavior classes based on $\boldsymbol{f}_{\text{morph}}$ and $\boldsymbol{f}_{\text{behavior}}$, respectively. We chose this supervised learning technique because studies (Fernández-Delgado et al., 2014; Wainberg et al., 2016) have proved it to be among the best general-purpose classification techniques. We used the default values for the main parameters: 100 trees in the ensemble and $\lfloor\sqrt{p}\rfloor$ features (i.e., $\left\lfloor\sqrt{|f_{\text{morph}}|}\right\rfloor = 16$ for the morphology classifier and $\left\lfloor\sqrt{|f_{\text{behavior}}|}\right\rfloor = 14$) for each tree.

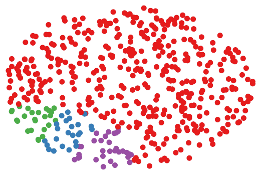
For an estimate of the accuracy of Random Forest on our two classification tasks, we performed a 5-fold cross-validation assessment using the 1555 labeled simulations (Table 3.1) and obtained an average accuracy of 0.833 and 0.891 for morphology and behavior classification, respectively. A trivial classifier (always predicting the most frequent class) taking into account the class imbalances obtained an accuracy of 0.442 and 0.478, respectively.

3.3.4 Simpson index

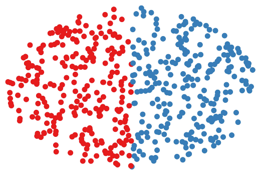
Different measures of diversity have been used in the ecological literature (Magurran, 2013). Among them, the Simpson index is one of the most commonly used (Simpson, 1949). Given a population of individuals that is partitioned based on species, this index is defined as $\lambda = \sum_{i=1}^{i=n} p_i^2$, where n is the number of species and p_i is the fraction of individuals of the i-th species.

Intuitively, the Simpson index measures the probability that two individuals picked at random with replacement belong to the same species. Since its semantics is the opposite of the one of diversity (i.e., $\lambda = 1$ for a population composed of a single species and it is < 1 for more diverse populations), in this study we use the Inverse Simpson index (ISI) λ^{-1} (defined in $[1, +\infty[$): the greater the ISI, the more diverse the population. Since we defined a limited number of possible species, i.e., 16, the actual domain of ISI in our study is $[1, 16]$.

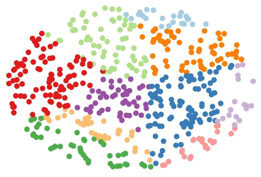
Simpson index is a suitable measure of diversity since it depends on both the total number of species, as well as their relative abundance. To better grasp this intuition, Figure 3.3 plots three example populations colored by species and their corresponding ISI. As can be seen, the red species pollutes most of the population in Figure 3.3a, and, as a result, the corresponding ISI is the lowest. The population in Figure 3.3b is evenly partitioned between two species; nevertheless, there are just two of them, so the total variety of species is not very high, and indeed ISI is only slightly higher than before. Finally, the population in Figure 3.3c witnesses a great variety of species, and all of them are evenly balanced in terms of abundance, which



(a) ISI = 1.41



(b) ISI = 2.00



(c) ISI = 7.82

Figure 3.3: Example populations colored by species, and their corresponding ISI.

is reflected in its ISI.

3.3.5 Evolution of VSR morphology *and* controller

We want to investigate how different factors (namely, the representation, the EA, and the environment) impact effectiveness and diversity in populations of VSRs, i.e., whether VSRs can be optimized for a given task, using EC, while maintaining diversity measured as above. To fully exploit the potential of expressing diverse solutions to a task, we need a way to evolve simultaneously the morphology and the controller.

We here propose different genotypic representations that jointly encode a description of both the morphology and the controller of a VSR in a single numerical vector $\boldsymbol{v} \in \mathbb{R}^p$. The resulting optimization problem is hence a search in the numerical space $\mathbb{R}^p$, for which many techniques do exist. We experiment with two representations for the controller and two for the morphology. We also experiment with four EAs that fit this scenario, two of them being tailored to the specific goal of promoting diversity. In the following subsections, we describe the representations and the EAs.

3.3.5.1 Representations

We define two representations for the controller and two representations for the morphology of the VSR in the form of a numerical vector $\boldsymbol{v} \in \mathbb{R}^p$. In both of them, a portion $\boldsymbol{v}_{\text{morph}}$ of $\boldsymbol{v}$ encodes a description of the morphology and the remaining, disjoint portion $\boldsymbol{v}_{\text{ctrl}}$ describes the controller, i.e., $\boldsymbol{v} = [\boldsymbol{v}_{\text{morph}}\, \boldsymbol{v}_{\text{ctrl}}]$. The controller representations differ in the latter, whereas the morphology representations differ in the former.

Controller We rely on the distributed controller of Section 2.2.2.1. It meets the requirement of the concurrent evolution of morphology and controller: since the architecture of each ANN, namely the size of the input and output layers is dictated only by the parameter n_{comm}, the structure of this distributed controller is agnostic with respect of the morphology of the VSR. Morphology and controller can hence be optimized together. We consider area, touch, and velocity sensors. Considering that the overall dimension of the sensor readings is 4, this results in each ANN having $4+4$ input neurons and $4+1$ output neurons; being there no hidden layers, each ANN is described by $8 \cdot 5 + 5 = 45 = n_{\text{ANN}}$ numerical parameters (the weights and biases of the edges connecting the neurons).

We propose two alternatives for the distributed controller. In the *Homogeneous* controller representation, denoted by Ho, we assume that all the ANNs have the same parameters $\boldsymbol{w}$. It follows that $\boldsymbol{v}_{\text{ctrl}} = \boldsymbol{w} \in \mathbb{R}^{n_{\text{ANN}}}$.

In the *Heterogeneous* controller representation, that we denote by He, we assume that ANNs may have different parameters. To favor the locality of the representation Rothlauf (2006) and to make the controller representation agnostic to the morphology representation, $\boldsymbol{v}_{\text{ctrl}}$ is the concatenation of the weights of the ANNs of all the voxels, i.e., $\boldsymbol{v}_{\text{ctrl}} = [\boldsymbol{w}_{1,1}\, \dots\, \boldsymbol{w}_{w \times l}]$, where $\boldsymbol{w}_{i,j}$ is the vector of parameters of the ANN at the (i,j) position in the enclosing grid. It follows that $\boldsymbol{v}_{\text{ctrl}} \in \mathbb{R}^{w \times l \times n_{\text{ANN}}}$. This representation is the same, for the controller part, as the one proposed in Medvet et al. (2020a).

The two controller representations differ in expressiveness. The heterogeneous representation is the most expressive one, thus resulting in the largest search space. The homogeneous representation is the least expressive one: its search space is smaller and hence, in principle, easier to explore. However, it might be harder for evolution to find the combination of genes that, when translated to the same ANNs for each voxel, results in a VSR that exhibits the desired complex behavior.

Morphology We propose two alternatives for representing the morphology. In the *Direct* morphology representation, we associate each gene with one and only one voxel of the final morphology, i.e., a collection of adjacent voxels arranged in a 2D grid. Given a $\boldsymbol{v}_{\text{morph}} \in \mathbb{R}^{w \times l}$, we build a morphology as follows. First, we build a Boolean matrix $\boldsymbol{b} = \{\text{T}, \text{F}\}^{w \times l}$ where $b_{x,y}$ is set to true if and only if $v_k > 0$, with $k = x + (y-1)l$. Then, we build the morphology by considering the largest connected component of $\boldsymbol{b}$ elements set to true and putting a voxel at each element of such set. As a consequence, $\boldsymbol{v}_{\text{morph}}$ comprises one number for every voxel in the grid, i.e., $|\boldsymbol{v}_{\text{morph}}| = w \times l$. Albeit simple, such direct representations have proved effective for the joint evolution of morphology and control of other kinds of embodied agents Ha (2019); Pagliuca and Nolfi (2020). Moreover, the irregularities that may arise from this direct representation of a VSR morphology are potentially beneficial for the adaptability of morphologies to different tasks Talamini et al. (2021).

In the *Gaussian Mixture Model (GMM)* morphology representation, based on the observations of Hiller and Lipson (2012); Cheney et al. (2013), we use the generative representation based on a mixture of bi-variate Gaussian distributions Lindsay (1995) described in Medvet et al. (2020b). Let n_{GMM} be the number of Gaussians in the mixture. First, we build a real matrix $\boldsymbol{b} = \{\text{T}, \text{F}\}^{w \times l}$ where $b_{x,y}$ is set to true if and only if $f(x', y') > 0$, with:

$$f(x', y') = \sum_{i=1}^{n_{\text{GMM}}} \frac{\phi_i}{2\pi\sigma_{i,x}\sigma_{i,y}} \exp^{-\frac{1}{2}\left(\frac{(x'-\mu_{i,x})}{\sigma_{i,x}}^2 + \frac{(y'-\mu_{i,y})}{\sigma_{i,y}}^2\right)},$$

where $x' = 2\frac{x}{w} - 1$ and $y' = 2\frac{y}{l} - 1$ are the x, y coordinates normalized in $[-1, 1]$, $\boldsymbol{\mu_i} = [\mu_{i,x}, \mu_{i,y}]$ and $\boldsymbol{\sigma}_i = \begin{bmatrix} \sigma_{i,x} & 0 \\ 0 & \sigma_{i,y} \end{bmatrix}$ are the mean vector and the covariance matrix for the i-th Gaussian, and $\phi_i \in [0, 1]$ is its mixing coefficient. Then, we build the morphology by considering the largest connected component of $\boldsymbol{b}$ elements set to true and putting a voxel at each element of such set. Since we restrict every $\boldsymbol{\sigma}_i$ to be diagonal, $\boldsymbol{v}_{\text{morph}}$ comprises 5 numbers for every Gaussian in the mixture, i.e., the two means, the two variances, and the mixing coefficient, so that $\boldsymbol{v}_{\text{morph}} \in \mathbb{R}^{5n_{\text{GMM}}}$. Note that we clip the values of $\boldsymbol{v}_{\text{morph}}$ corresponding to $\sigma_{i,x}$ and $\sigma_{i,y}$, for all i, in order to make them positive—e.g., for the first matrix, we set $\sigma_{1,x} = \max(0, v_{\text{morph},3})$.

The two morphology representations differ along two axes. First, they differ in terms of the compactness-expressiveness trade-off: the Direct representation presents, in general, a larger search space, thus holding the potential for more expressiveness. The GMM representation is potentially more compact (depending on the actual value of n_{GMM}), thus holding the potential for easier exploration of the search space. Second, we confront a direct representation with an indirect (or generative) one. In doing so, we tap ourselves into the debate surrounding direct and indirect encodings in ER Veenstra et al. (2017). As a consequence of those differences, the two representations potentially differ in the types of morphologies they are more suited to encoding. On one side, we expect the Direct representation to encode more irregular morphologies; while Cheney et al. (2013) proved direct representations to be sub-optimal for the evolution of VSR morphologies, they can increase the degree of complexity of a dynamical system Talamini et al. (2021) and put it in a better position to exploit morphological offloading, i.e., moving from the brain to the body the ability to store and processing information Nolfi (2021). On the other side, we expect the GMM representation to generate morphologies that are more regular, symmetrical, and composed of a few limbs. We evaluate how the morphology representations differ along these three axes by looking into the bias of the representation in Section 3.4.1.

For the sake of this study, we performed the experiments with $w = l = 10$ and $n_{\text{GMM}} = 5$, resulting in $|\boldsymbol{v}_{\text{ctrl}}|$ being $45 \cdot 10 \cdot 10 = 4500$ and 45, respectively for He and Ho representations, and $|\boldsymbol{v}_{\text{morph}}|$ being $10 \cdot 10 = 100$ and $5 \cdot 5 = 25$, respectively for Direct and GMM representations.

3.3.5.2 Evolutionary Algorithms

We use four EAs suitable for optimizing in the numerical space $\mathbb{R}^p$. Two of them are general purpose EAs, one is an EA that employs a form of speciation aimed at favoring diversity in the population—yet not based explicitly on the concept of species defined in Section 3.3.2—and one is a quality-diversity algorithm, a family of approaches that aim at returning a population that is both diverse as possible and effective as possible.

Evolutionary strategy The first EA is Canonical-ES (ES) (Chrabaszcz et al., 2018), a state-of-the-art Evolution Strategy. Evolution Strategies (Schwefel, 1965; Beyer and Schwefel, 2002) constitute a family of EAs including some variants that have recently been shown to achieve competitive results for continuous control tasks and game-playing (Salimans et al., 2017). ESs have also been used for evolving the controller of VSRs (Ferigo et al., 2022a).

ES iteratively evolves a fixed-size population of n_{pop} individuals as realizations of a multivariate normal distribution of mean $\boldsymbol{\mu} \in \mathbb{R}^p$ that is updated during the evolution. At iteration, n_{pop} children are born from $\boldsymbol{\mu}$, each one obtained by applying Gaussian noise $\boldsymbol{\epsilon}_i$ with σ_{mut} and zero mean:

$$\boldsymbol{v}_i = \boldsymbol{\mu} + \boldsymbol{\epsilon}_i \tag{3.1}$$

where $\boldsymbol{v}_i$ is the i-th child. Then, we update $\boldsymbol{\mu}$ by selecting the fittest quarter of the children and correcting $\boldsymbol{\mu}$ according to a weighted mean of the corresponding $\boldsymbol{\epsilon}_i$:

$$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \sum_{i=1}^{i=\left\lfloor \frac{n_{\text{pop}}}{4} \right\rfloor} w_i \boldsymbol{\epsilon}_i \tag{3.2}$$

with weights w_i set as in (Hansen and Ostermeier, 1996):

$$w_i = \frac{\log\left(n_{\text{pop}} + 0.5\right) - \log i}{\sum_{j=1}^{j=n_{\text{pop}}} \log\left(n_{\text{pop}} + 0.5\right) - \log j} \tag{3.3}$$

We set $\boldsymbol{\mu}$ by sampling uniformly in the interval $[-1, 1]$ for each vector element. After preliminary experiments and relying on our previous experience, we set $n_{\text{pop}} = 40$, $\sigma_{\text{mut}} = 0.35$, and let ES iterate until $n_{\text{evals}} = 30\,000$ fitness evaluations have happened.

ES is a form of population-based optimization. We remark, however, that the population in ES is indeed a realization of a multivariate normal distribution, i.e., all the individuals are "variations" of a single individual, the mean of the distribution. This observation is relevant in our settings, where we study the diversity of the evolved solutions.

Genetic algorithm As a second EA, we use a standard variant of Genetic Algorithm (GA). Our GA variant iteratively evolves a fixed-size population of n_{pop} individuals according to a $\mu + \lambda$ generational model (De Jong, 2006), i.e., with overlapping: at each generation, the offspring and the parents are merged and the worst half individuals are discarded. For building the offspring, we select individuals with tournament selection of size 5 and then apply Gaussian mutation with $\sigma_{\text{mut}} = 0.35$, with probability p_{mut}, or extended geometric crossover with probability $1 - p_{\text{mut}}$. For extended geometric crossover, given two parents $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathbb{R}^p$, the new individual is born as $\boldsymbol{v} = \boldsymbol{v}_1 + \boldsymbol{\alpha}(\boldsymbol{v}_2 - \boldsymbol{v}_1)$, where each element α_i of $\boldsymbol{\alpha}$ is chosen randomly with uniform probability in $[-1, 2]$. In this way, the new individual may fall outside the hypercube defined by the parents, hence favoring exploration. Moreover, we perturb each child of crossover by applying Gaussian mutation with $\sigma_{\text{mut}} = 0.1$, to prevent having genetically identical children based on selecting similar parents.

As for ES, we build the initial population by sampling uniformly in the interval $[-1, 1]$ and iterate until $n_{\text{evals}} = 30\,000$ fitness evaluations have happened. Moreover, we set $n_{\text{pop}} = 100$ and $p_{\text{mut}} = 0.2$.

Speciated evolver We designed this EA, which is denoted by SE (for Speciated Evolver), specifically for this study. SE employs a form of speciation inspired by

NEAT (Stanley and Miikkulainen, 2002), the popular EA for evolving the topology and the weights of ANNs. NEAT employed speciation to protect innovations introduced by modifications in the topology. In SE, we do not optimize the topology of the ANNs composing the controller of the VSR, while we do optimize the morphology of the VSR. Our goal is hence not to protect innovation, but explicitly to promote diversity.

Similarly to ES and GA, SE iteratively evolves a fixed-size population of n_{pop} individuals, as shown in Algorithm 1. At each iteration, individuals are partitioned into species according to a given criterion (described below) that also elects a single representative individual of each species (lines 5 and 13). Then, the current best individual in the population and the best individual of every species larger than n_{elite} are moved in the offspring (lines 6–11). The remaining individuals in the offspring are generated as follows. First, an offspring slot of size $n'_{\text{pop}}\alpha^{r_i}\frac{1}{\sum_{i=1}^{i=n}\alpha^{r_i}}$ is reserved to each species P_i depending on the rank r_i of the corresponding representative individual $\texttt{repr}(P_i)$ (line 16)—$\alpha \in \,]0,1]$ is a parameter of the algorithm, the closer to 1, the less the preference for fittest species. Then, the offspring slot is filled by applying Gaussian mutation or expanded geometric crossover (as in GA) to individuals of the corresponding species P_i (lines 17–25).

We explore three variants of SE. All three of them use the k-means clustering technique (Lloyd, 1982) for partitioning the population into species and electing as representative individuals the one closest to the centroid of the cluster. They differ in the features that are used for clustering individuals into species. In the first variant, which we denote by SE-g, we use the genotype $\boldsymbol{v}$, whose dimension depends on the representation. In the second variant, denoted by SE-s ("s" comes from shape), we use the vector $\boldsymbol{f}_{\text{morph}} \in \mathbb{R}^{256}$ of the morphology features (see Section 3.3.3). Finally, in the third variant, denoted by SE-b, we use the vector $\boldsymbol{f}_{\text{behavior}} \in \mathbb{R}^{205}$ of the behavior features (see Section 3.3.3). In all cases, we compute the Euclidean distance after having properly normalized the vectors of the individuals of the current population.

```
function evolve():
    $P \leftarrow$ initialize($n_\text{pop}$)
    while $\neg$shouldStop() do
        $P' \leftarrow \emptyset$
        $(P_1, \dots, P_k) \leftarrow$ kmeans($P$)
        $P' \leftarrow P' \cup \{$best($P$)$\}$
        foreach $i \in \{1, \dots, n\}$ do
            if $|P_i| \geq n_\text{elite}$ then
                $P' \leftarrow P' \cup \{$best($P_i$)$\}$
            end
        end
        $n'_\text{pop} \leftarrow n_\text{pop} - |P'|$
        $\boldsymbol{r} \leftarrow$ ranks(repr($P_1$), ..., repr($P_k$))
        foreach $i \in \{1, \dots, k\}$ do
            $c \leftarrow 0$
            while $c < n'_\text{pop} \alpha^{r_i} \frac{1}{\sum_{i=1}^{i=k} \alpha^{r_i}}$ do
                if $U(0,1) \leq p_\text{mut}$ then
                    $\boldsymbol{v} =$ nth($P_i, c \bmod |P_i|$)
                    $P' \leftarrow P' \cup \{$mutate($\boldsymbol{v}$)$\}$
                    $c \leftarrow c + 1$
                else
                    $\boldsymbol{v}_1 =$ nth($P_i, c \bmod |P_i|$)
                    $\boldsymbol{v}_2 =$ nth($P_i, (c+1) \bmod |P_i|$)
                    $P' \leftarrow P' \cup \{$crossover($\boldsymbol{v}_1, \boldsymbol{v}_2$)$\}$
                    $c \leftarrow c + 2$
                end
            end
        end
        $P \leftarrow P'$
    end
end
```

Algorithm 1: The algorithm of SE.

We designed SE with the main goal of studying the factors affecting diversity; the search effectiveness (both in terms of fitness and diversity) was not a design goal, while we believe that SE simplicity favors the analysis and the interpretation of the experimental results. We remark, however, that approaches similar to SE have been proposed and successfully employed in the past for evolving robots, in particular using behavioral similarity for speciating solutions (Trujillo et al., 2011).

In the experiments, we set $n_{\text{pop}} = 100$, $p_{\text{mut}} = 0.2$, as for GA, and iterate until $n_{\text{evals}} = 30\,000$ have happened, as for GA and ES. Moreover, we set $\alpha = 0.75$, $n_{\text{elite}} = 5$, and $k = 10$ (for k-means).

Quality-diversity algorithm Finally, we experiment with an established quality-diversity algorithm, Multidimensional Archive of Phenotypic Elites (MAP-Elites, hence further abbreviated as ME) (Cully et al., 2015).

ME computes a descriptor $\boldsymbol{d} \in \mathbb{R}^q$ for every individual to partition the descriptor space in a grid of n_{bin} cells for every dimension. ME starts with a population of n_{parents} individuals randomly initialized in $[-1, 1]^p$, evaluates them, computes their descriptors, maps the descriptors to the corresponding cell in the grid, and selects the best-performing individual of every non-empty cell. These individuals form the archive. At each iteration, n_{parents} children are born by mutating n_{parents} randomly chosen individuals in the archive with Gaussian mutation with $\sigma_{\text{mut}} = 0.35$, they are evaluated, and their descriptors are computed; if the descriptor of a child maps to a cell that is empty or stores an individual of lower fitness, the child is added to the archive and the individual of lower fitness is discarded. The algorithm iterates until n_{evals} fitness evaluations have happened.

We used the morphology descriptors of Section 3.3.2, so $\boldsymbol{d} = [d_{\text{num}}, d_{\text{elong}}, d_{\text{compact}}] \in \mathbb{R}^3$. We set $n_{\text{bins}} = 10$, $n_{\text{parents}} = 20$, and $n_{\text{evals}} = 30\,000$. These choices resulted in a grid with 10^3 cells.

ME is a simple yet effective algorithm that intrinsically creates an incentive to fill as many cells in the grid as possible, thus covering as much of the descriptor space as possible. Moreover, it has also been recently found particularly effective in the simultaneous evolution of morphology and controller of VSRs (Ferigo et al., 2022b). Here, we employed a simple and widespread version of ME, as we did for the other EAs considered in this study: however, later improvements have been proposed for ME which further increased its effectiveness, as, e.g., the directional variation operator proposed by Vassiliades and Mouret (2018).

3.4 Experiments and discussion

We aim to investigate how the three key factors of representation (both controller and morphology), EA, and environment impact diversity and effectiveness.

For all the experiments in the following sections, we considered the task of locomotion, since it is a classic task of evolutionary robotics (Nolfi and Floreano, 2000). The goal of the VSR is to travel as fast as possible, in the positive x direction, on a flat surface and within a time interval of $t_{\text{final}} = 30\,\text{s}$ (simulated time). The fitness of the individual is the average velocity v_x, measured considering the position of the center of mass of the VSR at the beginning and end of the simulation:

$$v_x = \frac{x_c(t_{\text{final}}) - x_c(0)}{t_{\text{final}}} \tag{3.4}$$

where $x_c(t)$ is the x-position of the center of mass of the VSR at time t. We remark that each simulation of any given VSR is deterministic.

We used JGEA[2] for the evolutionary optimization and 2D-VSR-Sim (Medvet et al., 2020b) for the simulation of the VSRs, with a time step of $\Delta t = \frac{1}{60}\,\text{s}$ and all the other parameters set to default values. The code of the experiments is publicly available at `https://github.com/pigozzif/VSRBiodiversity`. Table 3.2 reports an overview of all the parameter values used in our experiments.

For each experiment, i.e., a combination of representation, EA, and environment, we performed 10 evolutionary runs by varying the random seed of the EA. Table 3.3 summarizes, for each factor of investigation, the experimental settings. During each experiment, we saved the entire population of VSRs every 1000 fitness evaluations (to meet our storage constraints), which resulted in 3000 individuals for every run.

[2]`https://github.com/ericmedvet/jgea`

Table 3.2: Summary of the experimental parameters.

Context	Name	Description	Value
Controller	σ_{noise}	Standard deviation of the additive Gaussian noise applied to sensor readings	0.01
Controller	n_{comm}	Number of communication values between adjacent voxels	1
Representation (all)	n_{size}	Side length of the largest representable VSR	10
Representation (GMM)	n_{GMM}	Number of Gaussian distributions in the mixture	5
EA (ES)	σ_{mut}	Standard deviation of the Gaussian noise applied to children	0.35
EA (ES)	n_{pop}	Population size	30
EA (GA, SE-*)	p_{mut}	Probability of applying only mutation	0.2
EA (GA, SE-*, ME)	σ_{mut}	Standard deviation of the Gaussian mutation	0.35
EA (GA, SE-*)	n_{pop}	Population size	100
EA (SE-*)	α	Species preference	0.75
EA (SE-*)	n_{elite}	Minimum species size to preserve elite individual	5
EA (SE-*)	k	Number of species (found by k-means)	10
EA (ME)	n_{bin}	Number of bins per descriptor dimension	10
EA (ME)	n_{parents}	Number of randomly chosen parents	20
EA, all	n_{evals}	Number of fitness evaluations (as stop criterion)	30 000
Simulation	t_{final}	Duration of a simulation for the locomotion task (in s)	10
Simulation	Δt	Simulation time step (in s)	$\frac{1}{60}$

As a result, $10\cdot4\cdot2\cdot3000+10\cdot2\cdot3120+10\cdot2\cdot2\cdot3000+10\cdot2\cdot2\cdot3000+10\cdot3000 =$ 572 400 VSRs were generated in all of our experiments: for each one of them we applied the two classifiers for predicting the morphology and behavior classes learned on the subset of 1555 manually labeled VSRs, as described in Section 3.3.1.

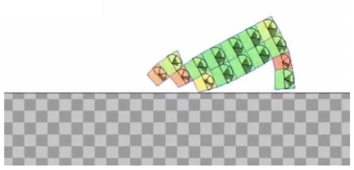
We carried out all statistical tests with the two-sided Mann-Whitney U rank test for independent samples, using, unless otherwise specified, 0.05 as the confidence level.

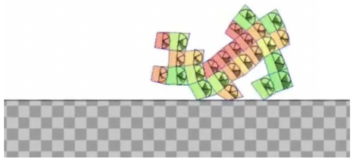
Table 3.3: Summary of the experiments. The table shows one row for each factor potentially impacting diversity and effectiveness. For each factor, the table shows the different experiments we performed, i.e., the different combinations of controller representation, morphology, EA, and environment.

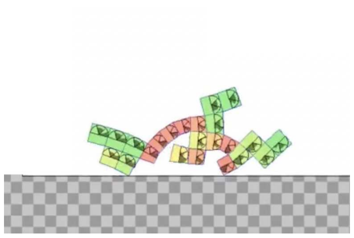
	Representation				
Factor	Contr.	Morph.	EA	Env.	N. of exp.
Contr. repr.	Ho, He	Direct	GA, SE-s	Flat	4
Morph. repr.	Ho	Direct, GMM	GA, SE-s	Flat	4
EA	Ho	Direct	ES, GA, SE-g, SE-s, SE-b, ME	Flat	6
Environment	Ho	Direct	GA, SE-s	Flat, Downhill, Uphill	6

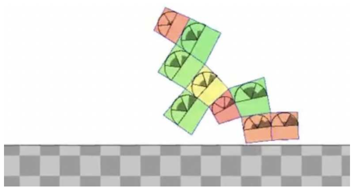
We consider average velocity $v_x^\star$ of the best individual as a measure of effectiveness and ISI (see Section 3.3.1) as a measure of diversity. We inquire into the landscape of diversity in the population of robots by taking the median values (across evolutionary runs) of $v_x^\star$ and ISI at the last generation. To gain more insights, for some experiments, we also present the number of VSRs broken down by classes, at the last generation, and the distribution of morphology (or behavior) descriptors.

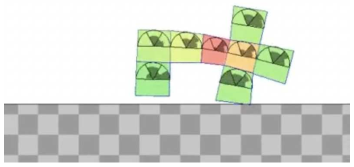
Every experimental configuration was able to evolve effective VSRs for the task of locomotion. We manually inspected a subset of the most effective VSRs (across every combination of representation/EA/environment tested) and found that they looked quite different. We showcase some of those VSRs in Figure 3.4; the corresponding video can be found at `https://youtu.be/_kblILsfivw`. Those hand-picked VSRs strikingly mirror emergent patterns found in nature. We nicknamed "gecko" one individual (Figure 3.4g), for example, for its clinging to an inclined surface and climbing up with a pair of short limbs. Another individual, "bigfoot" (Figure 3.4d), walked by treading a big extrusion that looked like a foot. Others slithered like centipedes or trotted like equines, to name a few traits. In general, both primitive and complex morphologies emerged. Interestingly, evolution succeeded in adaptation

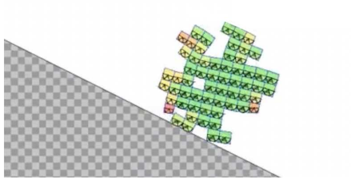

(a) Walrus

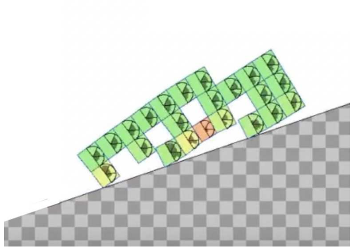

(b) Godzilla

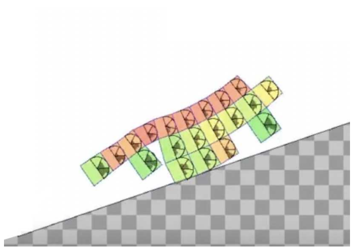

(c) Ankylosaurus


(d) Bigfoot


(e) Rhino


(f) Rolling stone


(g) Gecko


(h) Monkey

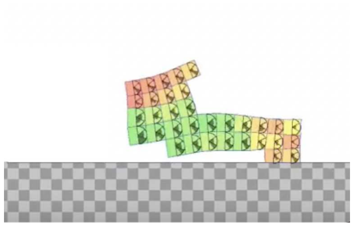

(i) Snail

Figure 3.4: A subset of outperforming individuals, in terms of effectiveness and variety of morphologies and behaviors. A video can be found at `https://youtu.be/_kblILsfivw`.

also with bizarre and unusual solutions. Some individuals covered long distances while possessing a morphology that might have turned out a handicap. As a proof of concept, one individual (nicknamed the "snail", Figure 3.4i) crawled forward despite carrying on its back an uncomfortable hump (resembling a shell indeed) that might have hindered motion.

3.4.1 Impact of the representation

In the next subsection, we investigate the impact of the representation on effectiveness and diversity for both the controller and the morphology.

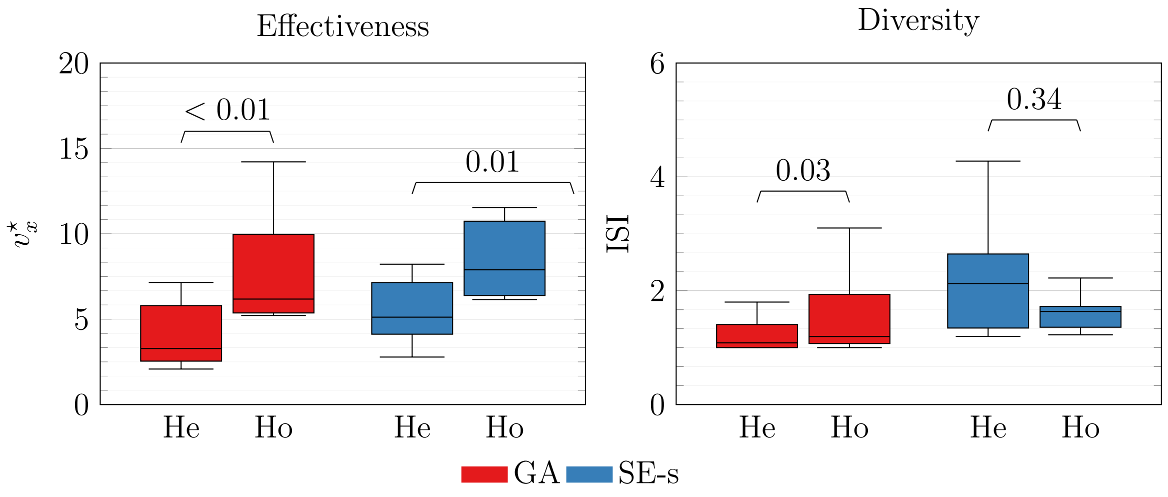


Figure 3.5: Boxplots of $v_x^\star$ and ISI of the population at the last generation, were obtained with four combinations of EA and controller representation (10 evolutionary runs for each combination). Each lower (upper) whisker is at the smallest (larger) data value greater than the lower (upper) quartile − (+) 1.5IQR, IQR being the interquartile range. Numbers above each pair of boxes are p-values. The Ho representation outperforms the He one in terms of fitness, while there is no clear difference in terms of diversity.

3.4.1.1 Controller representations

We performed an experimental campaign of 10 evolutionary runs on the He and Ho controller representations using GA and SE-s (i.e., the variant of SE in which the partitioning criterion is based on morphology features). We chose these two EAs because, as it turned out from our experiments described in Section 3.4.2, they were the ones with the weakest ability to promote diversity (GA) and the strongest ability to promote effectiveness (SE-s). Moreover, for the sake of this experiment, we adopt Direct as the sole morphology representation and discuss the comparison with GMM in the next subsection.

We report the results in Figure 3.5 in terms of $v_x^\star$ and ISI at the last generation, together with the p-values for every EA.

From the plot, two interesting conclusions can be made:

(a) Ho outperforms He with both EAs in terms of effectiveness, and

(b) there is no clear difference in terms of diversity.

The former means that despite the lower expressiveness of the Ho representation, a "single ANN" (i.e., one whose weights and biases are shared among all the voxels) is capable of driving cooperatively an entire robot when proper parameters are found. As it turns out from our experiments, finding these parameters is feasible with both EAs, probably because of the much lower dimension of the search space. p-values are significant for both EAs. Regarding conclusion (b), the much smaller search space induced by Ho does not result in a lower diversity when compared to He. The two representations are indeed comparable in terms of ISI; neither of the p-values is significant.

To conclude, the compactness of the search space—championed by Ho—triumphs over the expressiveness of the representation—championed by He. For these reasons, we adopt Ho as the only representation for the experiments in the next sections.

As an aside, we note that there is a contrast in terms of diversity between the two EAs, regardless of the controller representation (Figure 3.5, right): we discuss this aspect in more detail in Section 3.4.2.

Figure 3.6 reports, for each species resulting from the combination of a morphology class and a behavior class, the rate (bubble size) of VSRs at the last generation belonging to that species and the average velocity v_x (bubble color) of the best VSR of that species (that is not, in general, the best of the entire population).

While there seems to be no clear difference with SE-s, we spot some trends with GA. Diversity for Ho is greater along the behavior axis, with a relative majority of Walking individuals. On the other side, He seems to favor more diversity along the morphology axis, with Limbed and Other individuals being equally represented. Overall, in all the cases the percentage of robots following in the Other class for both morphology and behavior is significant: we remark that this finding may be explained partly in terms of the dataset used for training the classifier, which is

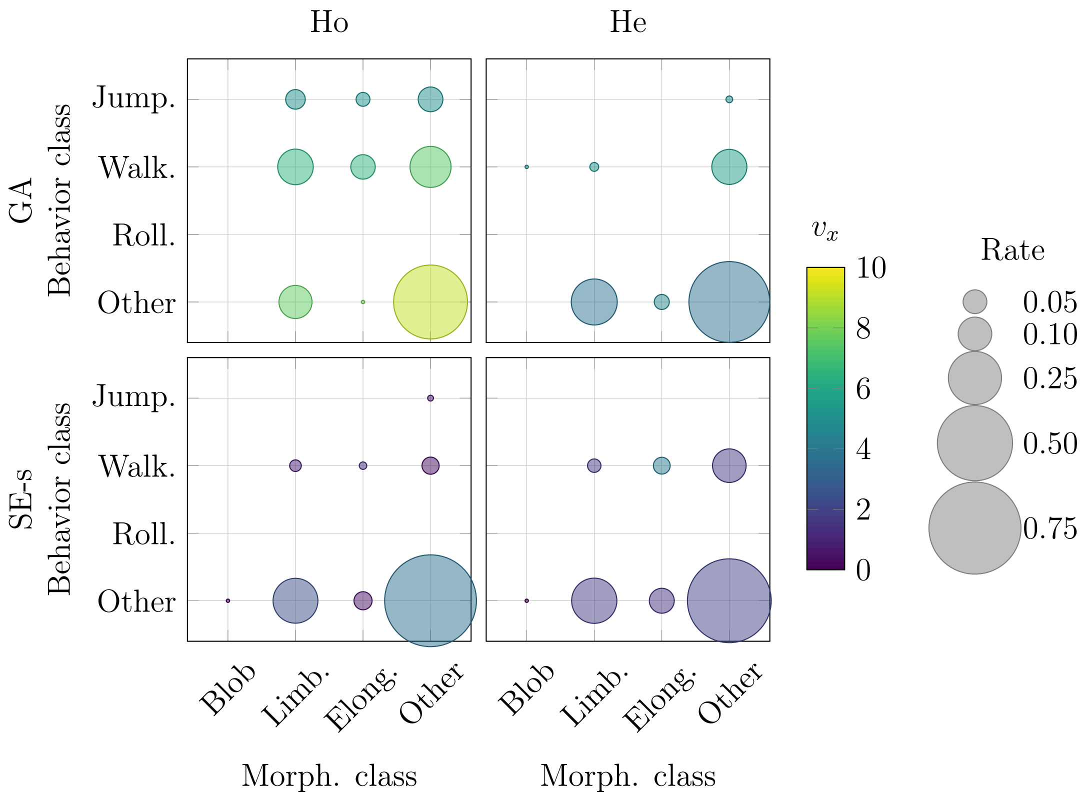


Figure 3.6: Rate (bubble size) of VSRs that belong to each species, resulting from the combination of a morphology class and a behavior class, and average velocity v_x (bubble color) of the best VSR of each species, at the last generation. The plot shows median values computed across 10 evolutionary runs performed with two EAs and two controller representations.

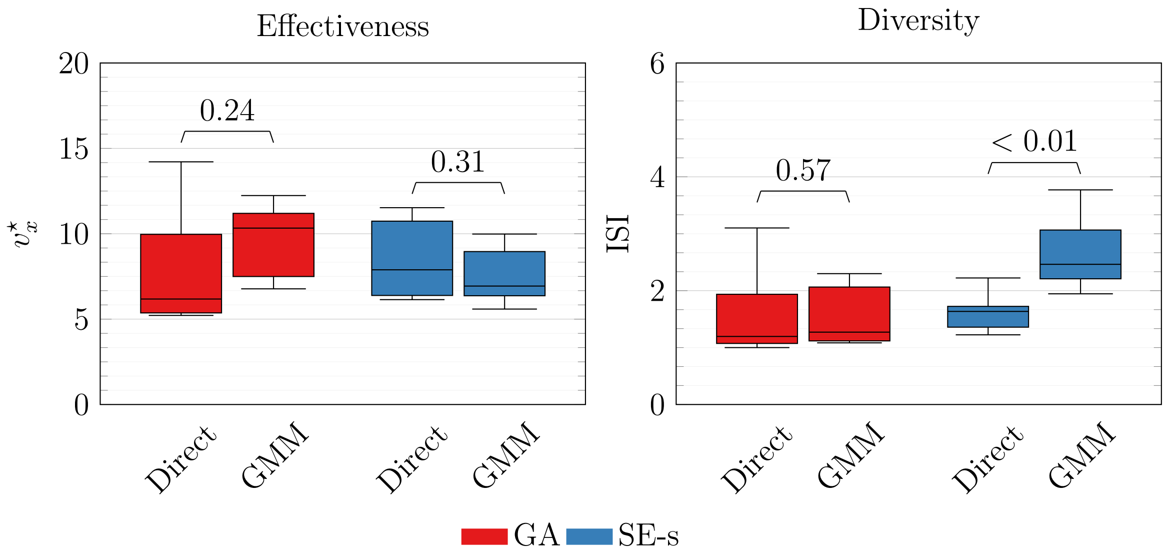


Figure 3.7: Boxplots of $v_x^\star$ and ISI of the population at the last generation, were obtained with four combinations of EA and morphology representation (10 evolutionary runs for each combination). Numbers above each pair of boxes are p-values. The Direct and the GMM representations are comparable in terms of effectiveness, while SE-s foster diversity more with the GMM representation.

slightly unbalanced (see Section 3.3.2). Finally, the absence of Rolling individuals is due to the flat terrain.

3.4.1.2 Morphology representation

We aim to investigate what impact the two morphology representations, namely Direct and GMM, have on effectiveness and diversity. As discussed in Section 3.4.1, we adopt Ho as controller representation, GA and SE-s as EAs. With these settings, we performed an experimental campaign of 10 evolutionary runs with the Direct and GMM morphology representations.

We report the results in Figure 3.7 in terms of $v_x^\star$ and ISI at the last generation, together with the p-values for every EA.

From Figure 3.7, it clearly turns out that the Direct and GMM representations are comparable in terms of effectiveness, with no clear differences across the EAs and no significant p-values. Considering diversity, GMM performs better with SE-s, and the p-value is significant. Once again, we remark that there seems to be a significant effect of the EA on diversity, and we treat it in Section 3.4.2.

We observe that both morphology representations are capable of evolving effective individuals, as well as preserving a fair amount of diversity (if we employ the appropriate EA). Thus, considering also the results of Section 3.4.1, we speculate that joint optimization of morphology and control is more susceptible to the choice of the controller rather than morphology representation. One reason might be that the morphology representations we consider in this work do not significantly differ in the dimension of the search space, whereas the same is not true for the controller representations.

We thus asked ourselves whether the two representations have a different bias in terms of species and emergent forms of “life”, and so what classes (i.e., species) did prevail. Figure 3.8 presents the breakdown by classes with the same visual syntax of Figure 3.6.

As can be seen from Figure 3.8, there are relevant differences in the GA case. In particular, the Direct representation is more capable of evolving individuals in the Walking behavior class.

Both representations witness a proliferation of individuals in the Other behavior class, which consists, for the most part, of idle individuals. This result confirms the previous observation that the joint evolution of morphology and controller is a difficult optimization problem: the majority of offspring individuals are born with brains that are ill-suited for their bodies, or vice versa. To tackle this problem, other studies employed age protection (Cheney et al., 2018) or proposed to embed a learning loop within evolution (Eiben and Hart, 2020b; Gupta et al., 2021). We chose to not consider these further design axes in our investigation and leave them as options for future work.

We asked ourselves whether the class differences in Figure 3.8 could be due to the morphology representations biasing the search towards different regions of the

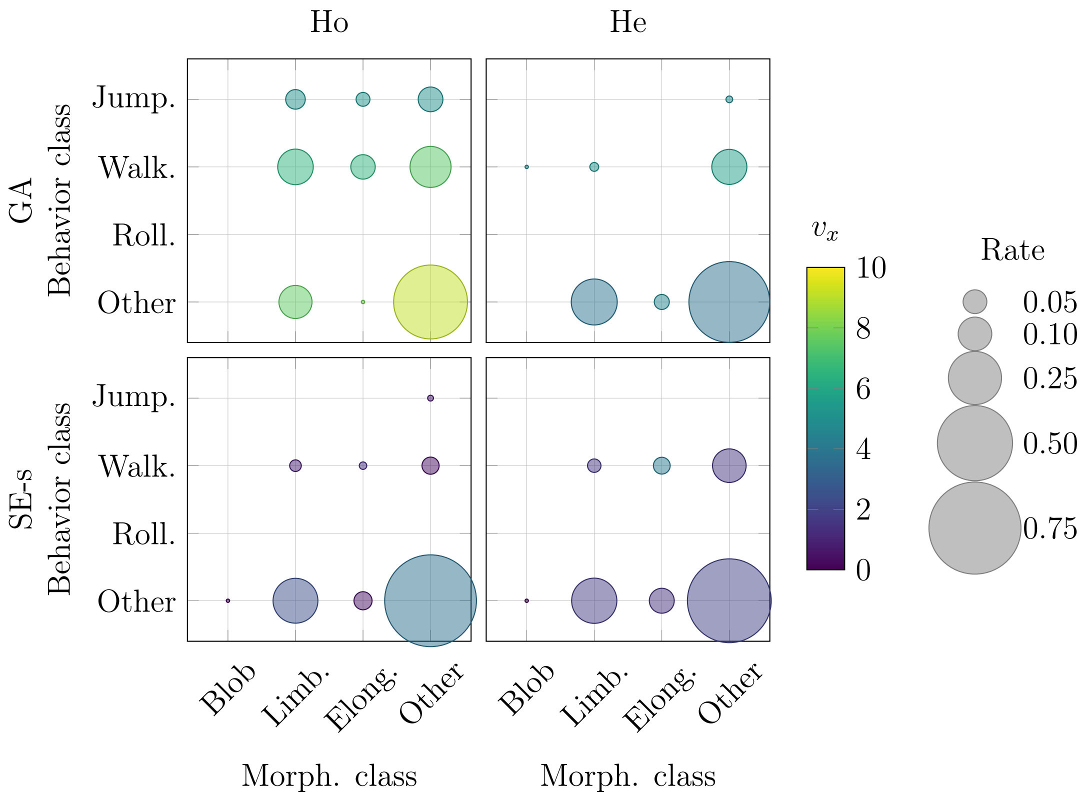


Figure 3.8: Rate (bubble size) of VSRs that belong to each species, resulting from the combination of a morphology class and a behavior class, and average velocity v_x (bubble color) of the best VSR of each species, at the last generation. The plot shows median values computed across 10 evolutionary runs performed with two EAs and two morphology representations.

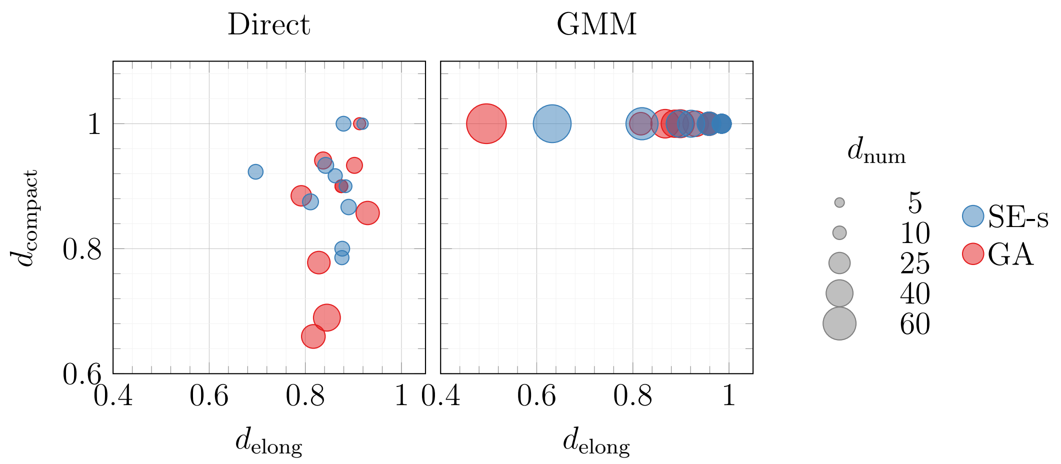
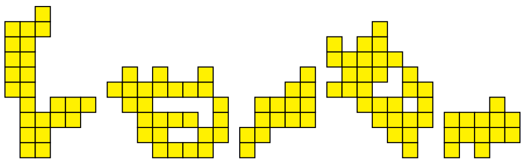


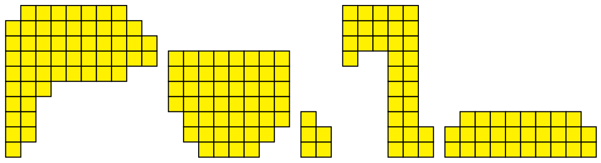
Figure 3.9: Scatter plot for elongation d_{elong} (x-axis), compactness d_{compact} (y-axis), and number of voxels d_{num} (marker size) descriptors computed on the best individual of each run (GA and SE-S together), and colored by the morphology representation. Best individuals evolved with the Direct representation cover more effectively the compactness axis, while best individuals evolved with the GMM representation cover more effectively the elongation axis.

space of morphologies. To verify this hypothesis, we plot in Figure 3.9 the three morphology descriptors introduced in Section 3.3.2, namely elongation d_{elong} (on the x-axis), compactness d_{compact} (on the y-axis), and number of voxels d_{num} (by means of marker size) for the best individuals at the last generation. We use the descriptors computed on the "static" morphology, rather than the features $\boldsymbol{f}_{\text{morph}}$ computed out of a simulation, for two reasons: first, the descriptors are fewer; second, they are not influenced by the behavior and are hence better suited for discussing the bias of the representation alone. We also show in Figure 3.10 a few sample morphologies found throughout the phylogenetic tree for Direct and GMM representations.

Figure 3.9 illustrates that the Direct representation covers more effectively the compactness axis, while the GMM representation covers more effectively the elongation axis. Direct best individuals are usually elongated and small in size, while GMM best individuals are all very compact and usually bigger in size. Figure 3.10 corroborates these observations, by showing how individuals evolved with the Direct morphology representation are, generally, more knotty and less regular, with many



(a) Direct



(b) GMM

Figure 3.10: Sample morphologies evolved with the Direct and GMM morphology representations. Direct individuals tend to be more “limbed”, while GMM individuals tend to be more compact.

protrusions that could potentially be employed as limbs for locomotion. This finding is particularly insightful if we consider that the Direct representation evolves very effectively in Walking individuals. Intuitively, limbs are necessary to generate a walking gait. GMM morphologies, on the other side, are more rounded and compact, sometimes spindly.

These observations raise an interesting point: the morphology representation biases the search towards particular regions of the behavior space (i.e., gaits). This conjecture is in line with the embodied cognition paradigm (Pfeifer and Bongard, 2006), which posits that there is an inextricable relationship between the body and the brain. As our results point out, the Direct representation is better suited for evolving “limbed” individuals, having body extrusions that contribute to locomotion by a walking gait.

Finally, we investigated whether the biases observed so far are due to the representation, or also to the concurrent evolutionary factors of selection and genetic operators (Aaron et al., 2022). To do so, we plot the distribution for the three morphology descriptors, d_{num}, d_{elong}, and d_{compact}, right after population initialization, and compare it with the distribution at the last generation. If a morphology representation did not introduce any bias, we would expect a uniform distribution after

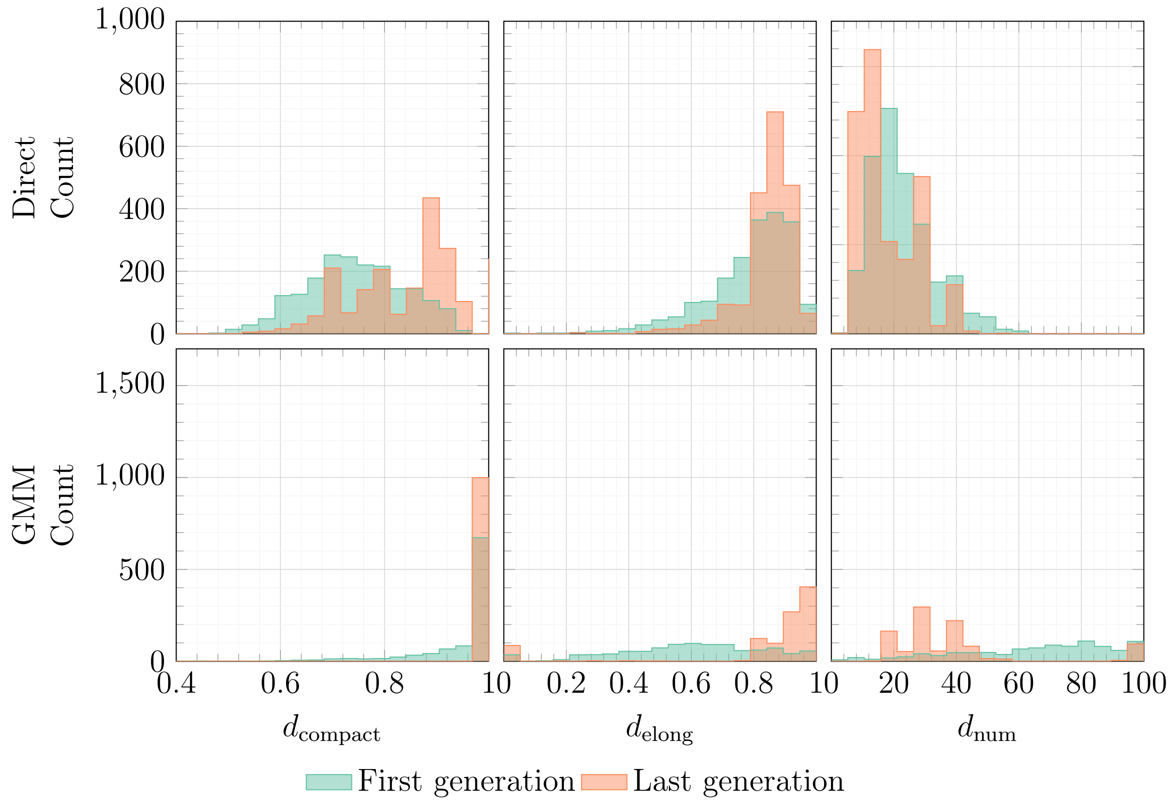


Figure 3.11: Histograms for the distribution of the three morphology descriptors d_{num}, d_{elong}, and d_{compact} computed on the population of the first and last generations (of both GA and SE-s), obtained with the two morphology representations. For a given descriptor, selective pressure biases the search in the same direction regardless of the representation.

population initialization. Figure 3.11 presents the results.

For a given descriptor, selective pressure biases the search in the same direction regardless of the representation, i.e., toward more compact, elongated, and smaller morphologies, as bigger individuals are probably more difficult to evolve.

3.4.2 Impact of the EA

We aim to investigate what impact the EA has on effectiveness and diversity. As discussed in Section 3.4.1, we adopt Ho as controller representation. Since, as reported in Section 3.4.1, Direct and GMM representations delivered comparable

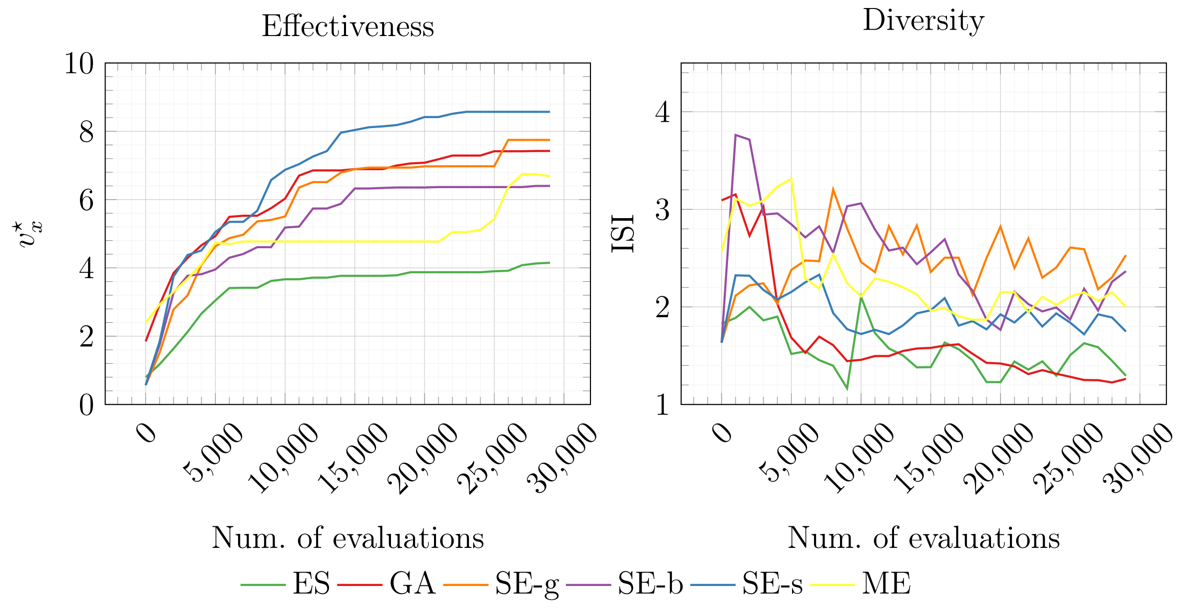


Figure 3.12: $v_x^\star$ and ISI during the evolution (median values across 10 evolutionary runs for each EA). Variants of SE and ME foster diversity the most.

results, we resort just to Direct as morphology representation for the sake of conciseness. With these settings, we performed an experimental campaign of 10 evolutionary runs with the six EAs described in Section 3.3.5, namely ES, GA, SE-g, SE-s, SE-b, and ME.

We show in Figure 3.12 how $v_x^\star$ and ISI vary over the course of evolution. We see that all EAs are able to evolve effective VSRs, i.e., they score well in terms of $v_x^\star$. In terms of ISI, SE variants and ME achieve the best results among the EAs.

Concerning diversity, the ISI plot in Figure 3.12 highlights some differences among the EAs: however, the variability of ISI across runs, as shown in Figure 3.13, is rather large for the majority of the EAs. The latter figure shows the distribution of ISI at the last generation, for all six EAs. By looking at Figure 3.12 it can be seen that SE variants and ME generally maintain a large amount of diversity in the population, in terms of median value during the evolution, whereas ES and GA do not. For ES, the finding is not surprising: as discussed in Section 3.3.5, this EA does not evolve a population of actually different individuals but rather evolves one

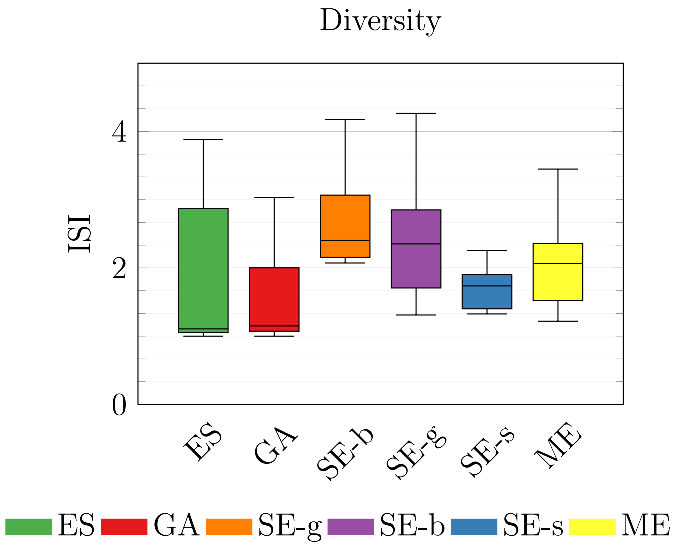


Figure 3.13: Boxplots of ISI at the last generation, obtained with six EAs, Ho controller representation and Direct morphology representation (10 evolutionary runs for each EA). ES and GA exhibit the lowest median diversity.

prototype individual by sampling its variants. GA performs similarly, witnessing a drastic drop in ISI around 5000 fitness evaluations. We looked at the raw results and found that the drastic drop in ISI is the outcome of the joint action of the generational model (that employs overlapping) and the crossover operator: a good individual often mates with a slightly modified copy of itself generating a "duplicate", rapidly swamping the population. We believe our GA turned out to operate with the wrong *exploration-exploitation trade-off*, a long-standing issue in EC (Črepinšek et al., 2013). We speculate that this limitation might be addressed by employing some diversity promotion mechanism (Squillero and Tonda, 2018), possibly acting at different levels of the representation (genotype, phenotype, fitness) (Bartoli et al., 2019). As expected, ME is comparable to SE variants at promoting diversity, whereas its slightly lower effectiveness may be due to its incentive for exploration: filling more cells in the archive does not necessarily entail better individuals.

Figures 3.14 and 3.15 present the breakdown by classes with the same visual syntax of Figure 3.6: Figure 3.14 shows one single plot for all the EAs; Figure 3.15 show six plots, one for each EA. In terms of effectiveness (color of each bubble in the figures), for the majority of the EAs, the fastest VSRs are those of class Other/Other, followed by Other/Walking; for ES, the best VSR belongs to the Limbed/Jumping class. By looking at the relative sizes of the bubbles, it can be seen that some EAs tend to favor a more even distribution of evolved VSRs across classes. Overall, Figure 3.15 is consistent with Figure 3.13: SE-s exhibits the most uneven distribution of bubble sizes. The relative majority of the evolved individuals belong to the Other behavior class, both globally and for each EA. We traced individuals of the Other behavior class to two major sub-classes: idle and vibrating. Not surprisingly, idle individuals abound, as already discussed in Section 3.4.1. Vibrating individuals manifest a behavior similar to that of Walking, but move their body at a higher frequency that makes it hard to discern which parts of the body are touching the ground and which ones are not. In this way, they achieve very high fitness. Nevertheless, the results concerning vibrating individuals call for some further remarks. If we attempted to physically realize those vibrating VSRs, maybe using the approaches of (Kriegman et al., 2020a; Sui et al., 2020; Legrand et al., 2023), they would likely not be as fast as in simulation—i.e., there would likely be a *reality gap* problem (Mouret and Chatzilygeroudis, 2017). We think that the vibrating behavior evolves frequently for two reasons. First, we do not consider energy consumption in our simulations, and hence inefficient behaviors are not discouraged (Joachimczak et al., 2016). Second, the recurrent nature of our neural controller, and in particular the voxel-to-voxel message passing, likely favors the emergence of high-frequency dynamics (Medvet et al., 2020a).

One last comment concerns the way we evaluate effectiveness. So far, we reported results in terms of $v_x^\star$, i.e., the average velocity of the best individual. In Figure 3.16 we plot the distribution of $\overline{v}_x$, i.e., median average velocity within the entire population of the last generation, for all six EAs, side-by-side with the corresponding $v_x^\star$.

As expected, GA witnesses a non-significant difference between the best and median. Figure 3.16 corroborates the finding that SE variants, especially SE-g and

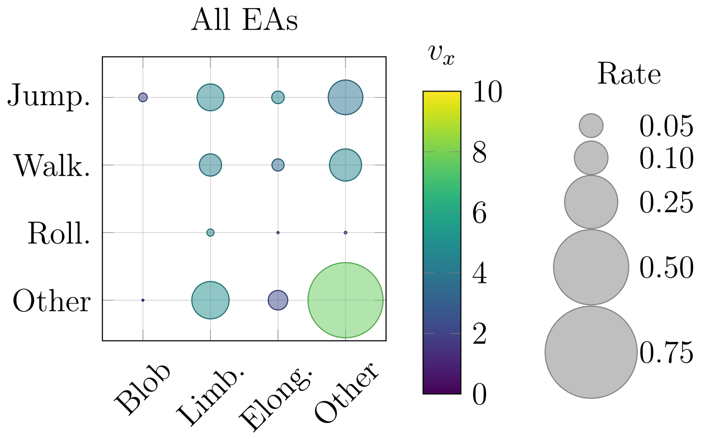


Figure 3.14: Rate (bubble size) of VSRs that belong to each species, resulting from the combination of a morphology class and a behavior class, and average velocity v_x (bubble color) of the best VSR of each species, at the last generation. The plot shows median values computed across all the evolutionary runs performed with all the EAs, considered together.

SE-s, foster a greater amount of diversity: differences between best and median are strongly significant, attesting that speciation promotes diversity by protecting low-performing species in the population. Interestingly, with SE-b the same does not happen: from this point of view, this EA seems the one that better fits the needs of an autonomous robotic ecosystem, by preserving different but still effective species.

3.4.3 Impact of the environment

The third and last factor we consider in our study is the environment. In our experimental setup, the terrain profile plays the role of the environment: more arduous terrains (e.g., uphill) correspond to less hospitable environments and less arduous terrains (e.g., downhill) correspond to more hospitable environments. Based on the results of Section 3.4.1, we adopt Ho and Direct as controller and morphology representations, respectively, while based on Section 3.4.2, we use GA and SE-s as EAs. With these settings, we performed an experimental campaign of 10 evolutionary runs with three different terrains:

(a) flat;

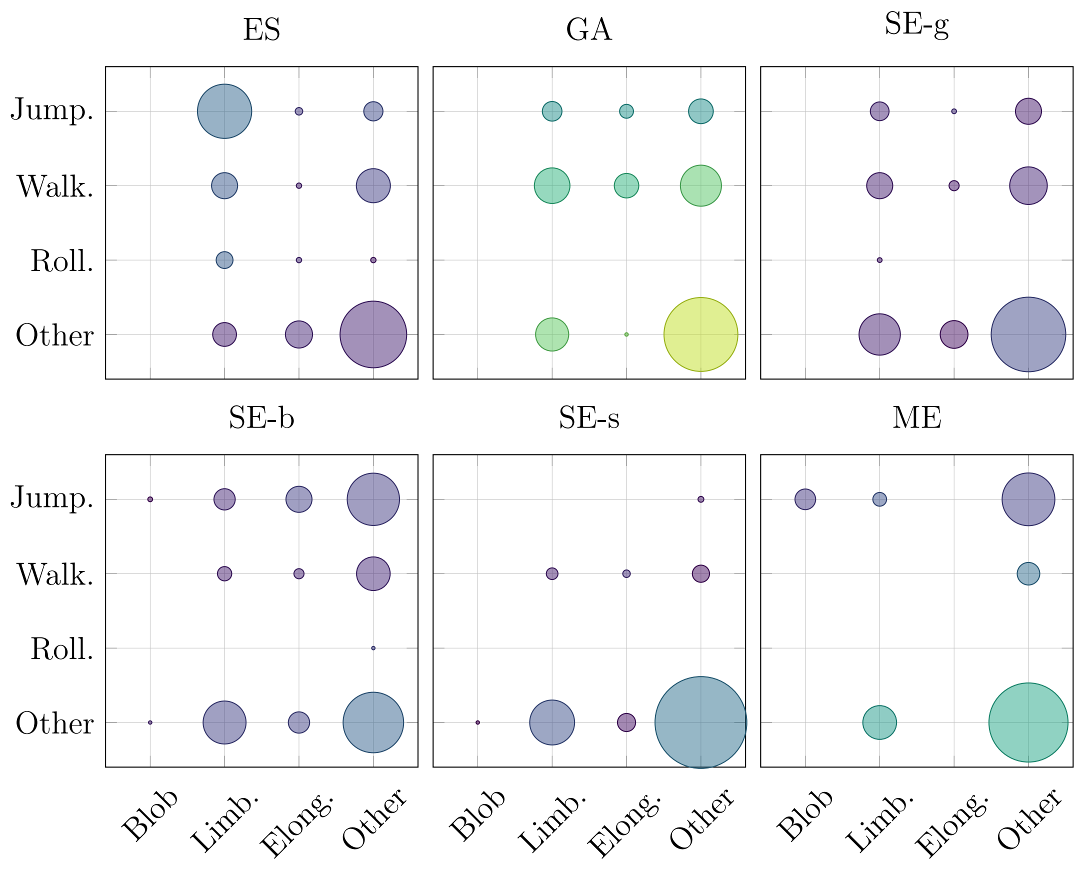


Figure 3.15: Rate (bubble size) of VSRs that belong to each species, resulting from the combination of a morphology class and a behavior class, and average velocity v_x (bubble color) of the best VSR of each species, at the last generation. Each plot shows median values computed across all the evolutionary runs performed with each one of the EAs.

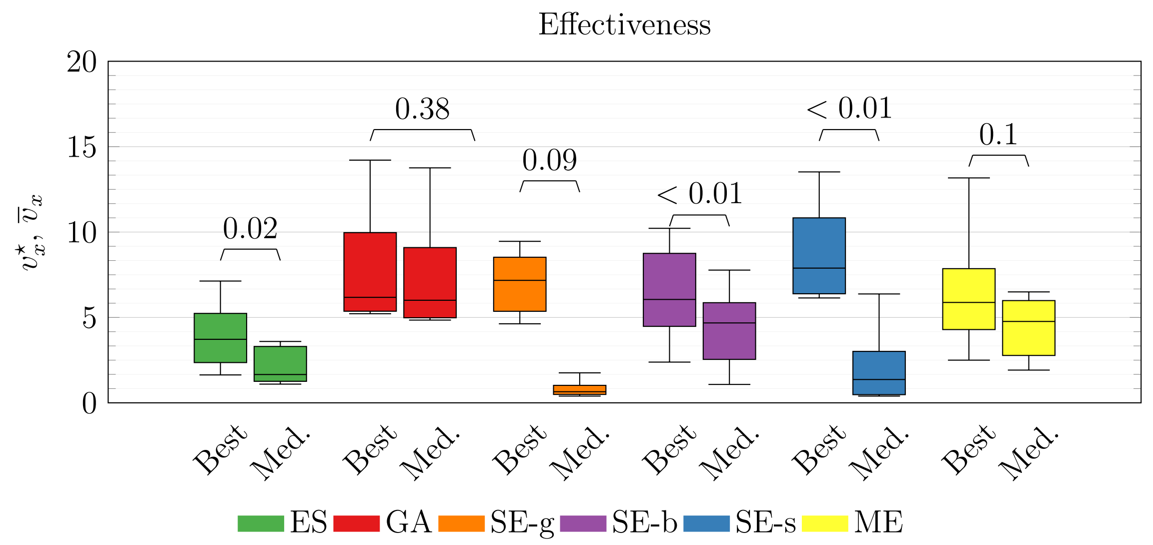


Figure 3.16: Boxplots of best average velocity $v_x^\star$ and median average velocity $\overline{v}_x$ at the last generation, obtained with six EAs, Ho controller representation, and Direct morphology representation (10 evolutionary runs for each EA). Numbers above each pair of boxes are p-values. SE-g and SE-s witness a dramatic difference between best and median average velocity.

(b) downhill, consisting of a flat surface tilted downward by 30°;

(c) uphill, consisting of a flat surface tilted upward by 20°.

We report the results in Figure 3.17 in terms of $v_x^\star$ and ISI at the last generation, together with the p-values.

From the plot, we draw the following observations:

(a) downhill is the terrain favoring effectiveness the most, uphill the least;

(b) downhill appears to favor more diversity than the other two terrains, but only when using SE-s as EA;

(c) as already discussed in Section 3.4.2, the EA impacts diversity a lot, while the same is not true for effectiveness.

We found noticeable differences between the two new terrains in terms of the evolved species. We summarize the results in Figure 3.18, using the same visual

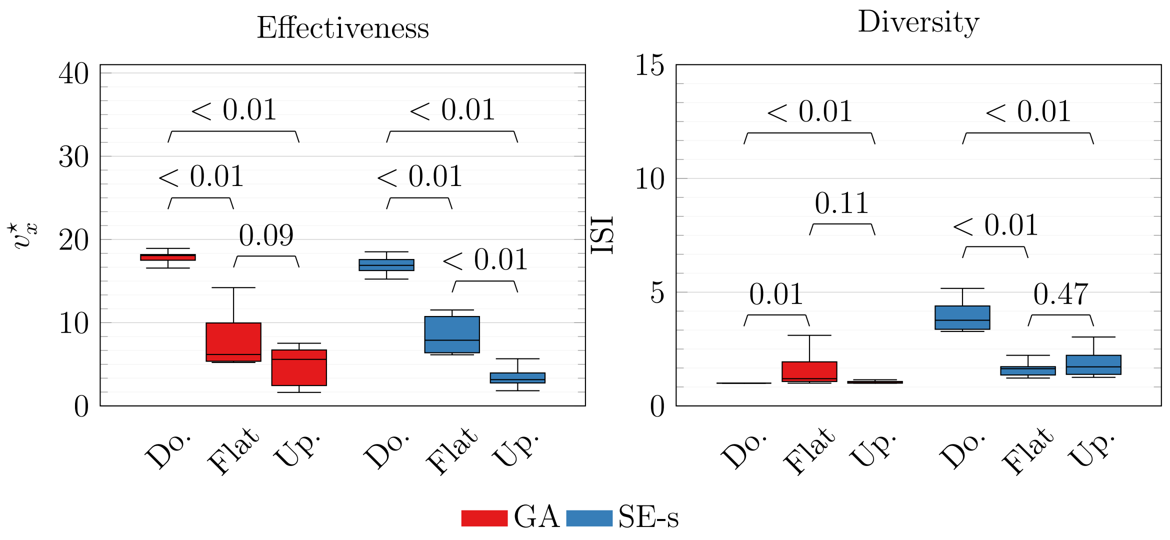


Figure 3.17: Distribution of $v_x^\star$ and ISI at the last generation, obtained on three different terrains with two EAs, Ho controller representation, and Direct morphology representation. Numbers above each pair of boxes are p-values. Evolution on downhill terrain finds more effective solutions, while evolution on uphill terrain finds less effective solutions. The EA impacts diversity, while it does not do so in terms of effectiveness.

syntax of Figure 3.6. The most effective individuals who evolved downhill relied, for the vast majority, on Rolling to achieve very high effectiveness. To our surprise, Rolling did not amount to merely falling down the surface. Instead, successful Rolling individuals nudged themselves to create momentum at the very onset of the simulation and expanded their bodies at every roll to accompany the descent. On the other side, the most effective individuals who evolved uphill mostly relied on crawling to clinch to the upright surface, resist gravity, and slowly move forward (a behavior labeled as Walking). Being an arduous environment, uphill generated a disproportionate amount of idle individuals of class Other, whom gravity dragged down.

As far as the morphology is concerned, downhill did favor Blob individuals, since Blob is a very suitable morphology for the Rolling behavior. On the other hand, uphill exerted high selective pressure on Blob individuals, since such morphology is not suitable for climbing an inclined surface. Instead, evolution favored Elongated individuals, who could stretch over the inclined surface and clinch to it.

At a high level, it might look like populations evolved with downhill become swamped by Rolling/Blob individuals, and so downhill does not favor diversity. This intuition is in contrast to what Figure 3.17 reports, as SE-s achieves a high ISI downhill. We investigated this contradiction and found that although diverse species did indeed coexist within the same generation, Blob/Rolling tended to be the species with better effectiveness.

Finally, we considered the pool of individuals from the last generations of both EAs and performed PCA on either their morphology features or their behavior features from Section 3.3.3 and projected the best individual of each run on the resulting 2D space. Figure 3.19 provides the results, one marker for each best individual, separately for morphology and behavior. Marker style stands for the terrain, marker color stands for morphology or behavior class.

The PCA analysis corroborates the previous discussion, as the best individuals evolved on a downhill cluster together in the morphology and behavior spaces while

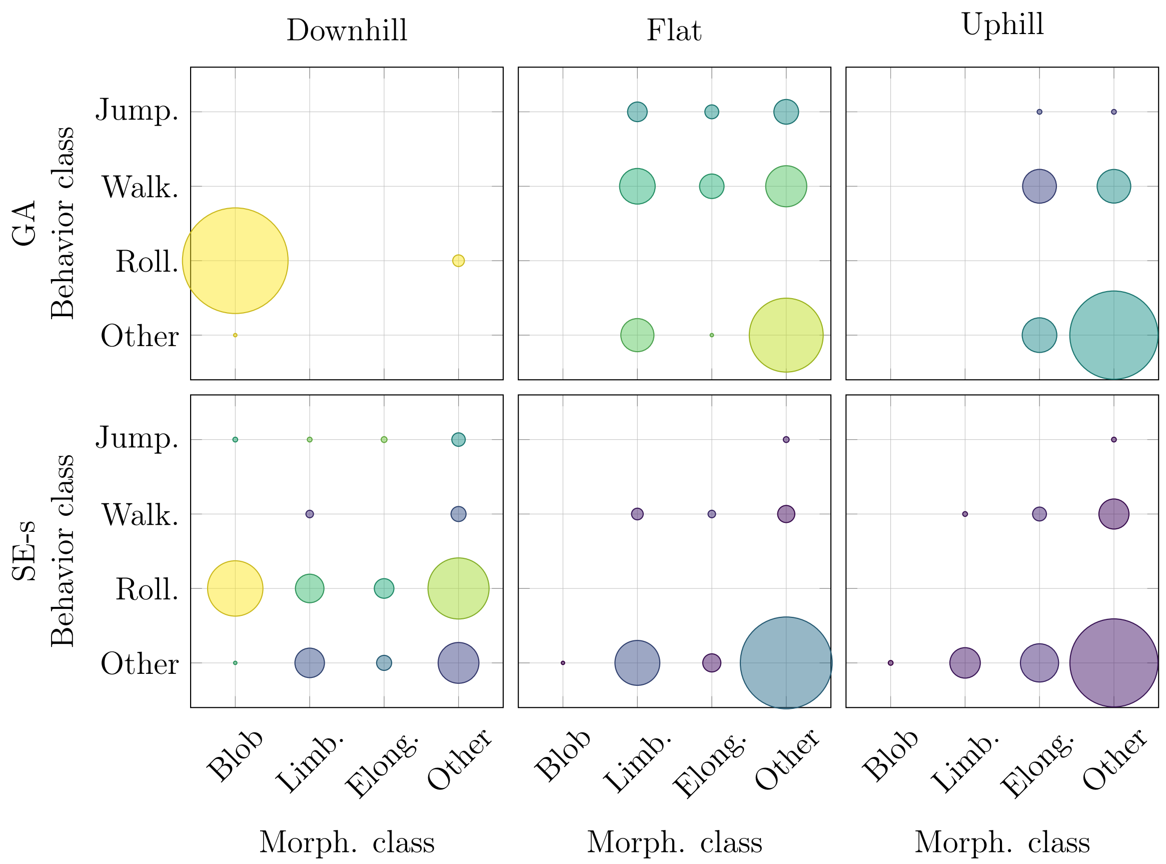


Figure 3.18: Rate (bubble size) of VSRs that belong to each species, resulting from the combination of a morphology class and a behavior class, and average velocity v_x (bubble color) of the best VSR of each species, at the last generation. The plot shows median values computed across 10 evolutionary runs performed with two EAs and three environments. Bubble color and size encoding are the same of Figure 3.6. Downhill terrain favors Blob and Rolling individuals, while uphill favors Elongated individuals.

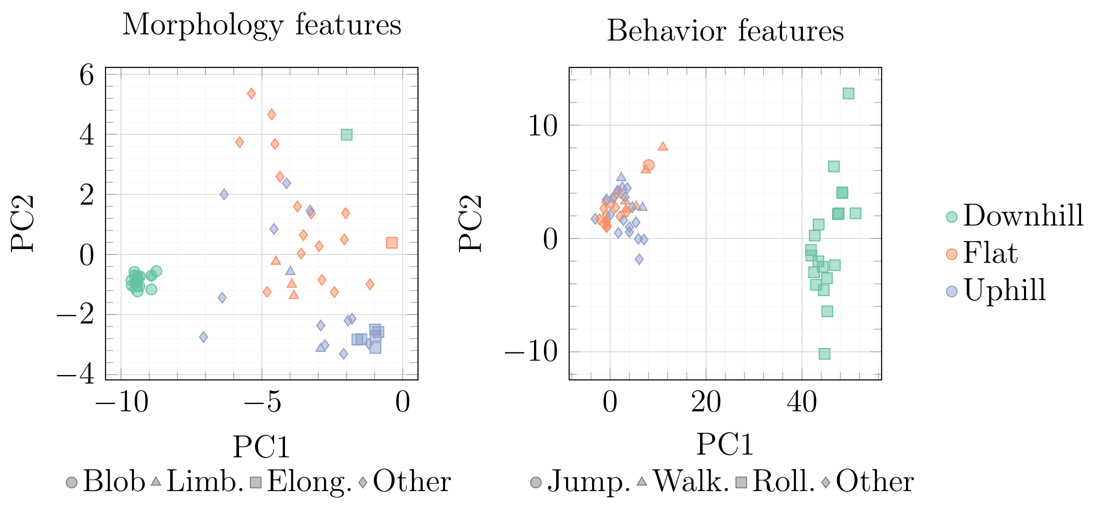


Figure 3.19: PCA analysis on morphology features and behavior features. Each marker corresponds to the projection of the best individual of a run (of both GA and SE-s). Marker style stands for the terrain the individual was evolved with. Marker color stands for the morphology or behavior class. Best individuals who evolved downhill share the same morphology (Blob) and same behavior (Rolling) classes, while a majority of individuals who evolved uphill are Elongated.

sharing the same Blob/Rolling class. Considerations made so far for the uphill terrain stand as well.

To conclude, the effect of the environment on effectiveness is revealed as expected: more hospitable environments correspond to higher values since effectiveness is an absolute measure. At the same time, diversity is higher on downhill terrain, which is a very hospitable environment, and only if the EA is capable of favoring it; the reason for this might be less competition.

3.5 Concluding remarks

In this chapter, we considered the automatic design of a kind of VSRs, through EC and investigated experimentally the impact of three key factors on the effectiveness and diversity of evolved VSRs. In the long-term vision of robotic ecosystems that are capable of staying resilient to environmental changes even without the intervention of human designers, diversity plays a key role. However, evolving effective *and* diverse robots is not an easy task for at least two reasons. First, optimizing concurrently the morphology and the controller of effective robots is known to be difficult (Lipson et al., 2016). Second, the interplay between the quality and diversity of evolved solutions is complex in the more general context of evolutionary optimization (Cully and Demiris, 2017).

We considered three key factors (representation, EA, and environment) and performed several experiments in which we evolved VSRs for the task of locomotion. For analyzing the diversity of the population of evolved robots, we relied on a well-established index inspired by biology, the inverted Simpsons index, that measures how many species there are in a population and how evenly they are distributed. For assigning species to VSRs, we used a supervised machine learning approach: we defined the classes by visual inspection, collected a few examples by manual labeling, engineered suitable features, and learned a model.

We also proposed a novel EA based on a form of speciation, inspired by NEAT, and instantiated it in three variants, with species depending on the genotype, morphological traits, and behavioral traits. We showed experimentally that the proposed EA is in general able to preserve diversity without affecting effectiveness. In particular, SE-b, the variant based on behavioral traits, seems to achieve the best results. We leave to future work the investigation on which other traits of VSRs may be useful for promoting diversity with SE-b or similar EAs. Beyond manually defined descriptors, which have the drawback of being task- and robot-specific, automatic methods like those of (Paolo et al., 2020; Cully, 2019) appear promising. Despite it is not obvious to which degree these results can be extended to other classes of robots and tasks than the one considered in this study, we believe our methodology can be easily ported to other cases.

Moving forward, we will consider the Ho variant of the distributed controller of Section 2.2.2.1, since it is comparable to its He counterpart in terms of effectiveness and diversity while offering a more compact search space.

Chapter 4: Modularity in Collective Intelligence

Modularity is a desirable property for embodied agents, as it could foster their suitability to different domains by disassembling them into transferable modules that can be reassembled differently. VSRs' morphologies are intrinsically modular; yet, controllers used until now for VSRs act as abstract, disembodied processing units: disassembling such controllers for module transferability is a challenging problem. Thus, the full potential of modularity for VSRs remains untapped. In Section 4.1, we propose a novel self-organizing, distributed, and embodied neural controller for VSRs. We evolve it for a given task and morphology: while evolving, the controller spreads across the VSR morphology in a way that permits the emergence of modularity. Our experiments confirm that our self-organizing, embodied controller is indeed effective on locomotion on rugged terrain. By mimicking the structural modularity observed in biological neural networks, different levels of modularity can be achieved.

Distributed robotic controllers usually rely on inter-module communication, a practical requirement that makes modules not perfectly interchangeable and thus limits their flexibility. Indeed, the self-organizing, embodied controller of Section 4.1 still relies on neural synapses reaching from one voxel to another. To address this limitation, in Section 4.2 we use the same neural controller inside each voxel, but without any inter-voxel communication, hence enabling ideal conditions for modularity: modules are all equal and interchangeable. We evolve the parameters of the neural controller—shared among the voxels. Crucially, we use a local self-attention mechanism inside the controller to overcome the absence of inter-module communication, thus enabling our robots to only implicitly communicate through mechanical interactions and be truly driven by the CI of their modules.

4.1 Evolving modularity in soft robots through an embodied and self-organizing neural controller

In this chapter, we answer to question **1**.

4.1.1 Introduction

Albeit their bodies are intrinsically modular, controllers used until now for VSRs act as abstract, disembodied processing units: disassembling such VSRs to reassemble differently, perhaps by combining modules from different VSRs, is a challenging problem. Indeed, modularity is a very desirable property on the road toward fully autonomous robotic ecosystems. Thus, despite the tremendous achievements of VSRs, the full potential for modularity remains unexploited.

To address this dilemma, we propose a novel self-organizing, embodied Artificial Neural Network (ANN)-based controller for VSRs. Nodes and edges are precisely located throughout the VSR body without a topology fixed a priori. We optimize the controller for a locomotion task (on rugged terrain) and a given morphology through EC: while evolving, the controller spreads and self-organizes across the VSR body in a way that permits the emergence of modularity. Self-organization allows one to tune the degree of neural complexity across the body.

We experimentally inquire whether such a controller (i) is effective at solving the task, (ii) is transferable through disassembling and reassembling its body and what role does modularity play in the process, and whether (iii) it allows for the automatic discovery of modules.

Our experimental results confirm that our controller is indeed effective. We also find that, when disassembling and reassembling by combining modules from different VSRs, modularity is crucial: more modular controllers turn out to be more transferable than less modular controllers. Finally, our embodied, self-organizing controller allows for the automatic discovery of modules to be transferred. As such, this work positions itself on the road toward an automatic pipeline for the fabrication of new robots in a human-out-of-the-loop manner by assembling modules from other robots.

4.1.2 Related work

This work is relevant for research on the topic of modularity in the fields of artificial intelligence and robotics. *Modularity* is an emergent property of a complex system, typically a network. We take inspiration from Yamashita and Tani (2008) and distinguish between structural and functional modularity. In the former, dense connectivity within modules and sparse connectivity between modules emerge. In the latter, a network supports many different functional patterns. For the sake of this study, we are mostly concerned with structural modularity.

From a biological viewpoint, many scientists agree that modularity (the ability to separate functional processes into *modules*) played a key role in the evolvability of natural systems (Wagner and Altenberg, 2005; Wagner et al., 2007). Recently, Gutai and Gorochowski (2021) suggested that modern ANNs are much less modular than biological neural networks, and thus suffer from being *monolithic*. Concurrently, Eiben (2021a) put modularity under the spotlight as a key ingredient in the road toward fully autonomous robotic ecosystems. Indeed, Faiña (2021) described the benefits of modularity as simplifying the search space for robotic morphologies and controllers. Additionally, as made clear in Yim et al. (2007), the versatility, robustness, and cheapness of modular robots make them suitable for deployment in a short time.

Under these premises, there has been a growing body of literature devoted to the topic of modular robotics. Starting from the early theoretical formulations (Neumann and Burks, 1966), the last decades saw many physical realizations being proposed (Howison et al., 2020), including soft ones (Sui et al., 2020). Platforms for the automatic design and manufacture of robots from modular components have also been recently explored (Faiña et al., 2015; Moreno et al., 2018). Although interest and progress have been remarkable, there remain practical and theoretical challenges to be addressed (Liu et al., 2016). In this scenario, our work can be seen as a stepping stone toward reconfigurable robots with non-trivial control.

While the idea of optimizing ANNs through EC is not new (Kitano, 1990; de Garis, 1998; Stanley and Miikkulainen, 2002; Stanley et al., 2009), not a lot of studies have investigated the interplay between the evolution of ANNs and modularity. Even more, while their achievements have been tremendous, most state-of-the-art neuroevolutionary algorithms fail to produce modular ANNs (Clune et al., 2010). The handful of studies related to evolution and modularity mostly highlight the many benefits of modular ANNs, especially when applied to robotic tasks. In a series of experiments (Bongard, 2011; Bongard et al., 2015; Cappelle et al., 2016), modularity turned out to improve the generalization abilities of robotic agents, and Ellefsen et al. (2015) showed it can also subdue the infamous problem of catastrophic forgetting in ANNs (French, 1999) via reinforcement learning. Other studies highlighted the interlink between modularity and specialization (e.g., multi-tasking), both under a computational (Schrum and Miikkulainen, 2014) and a biological (Espinosa-Soto and Wagner, 2010) perspective. Finally, in a computational biology study, Clune et al. (2013) demonstrated how modularity can emerge in ANNs as a "spandrel"—i.e., a phenotypic trait that is a byproduct of the evolution of some other trait (Gould and Lewontin, 1979)—to minimize connection costs between neurons.

Despite being ground-breaking, these works suffer from one (or both) of the following pitfalls. First, only some of them (Bongard et al., 2015; Cappelle et al., 2016) are conducted under the embodied cognition paradigm, which postulates that intelligence emerges from the complex interactions between the brain, the body, and the environment (Brooks, 1990; Pfeifer and Bongard, 2006). The others consider ANNs that function as abstract processing units and are not pervasive for the body they control. Indeed, Mitchell (2021) listed a lack of embodiment as one of the four "fallacies" of artificial intelligence, and the dualist bias "body vs. mind" is well-known to be rooted in our culture (Bloom, 2004). Second, even when under an embodiment perspective, none of them addresses how to self-organize modularity inside a robotic agent body and exploit it for the sake of reconfigurability.

In this chapter, we build on our previous research. In Medvet et al. (2020a), we proposed a representation for a distributed and embodied controller and a reconfigurability procedure for VSRs that, together, proved effective. For the sake of this study, we term that procedure disassembly-reassembly and describe it thoroughly in Section 4.1.5.3. Later, in Medvet et al. (2021) and Pigozzi et al. (2023b), we showed how that representation could be made more compact without loss of effectiveness or diversity. We extend our previous studies by introducing a self-organizing controller that is capable of supporting differing degrees of complexity across the body; as a result, it is better suited to further exploit the intrinsic modularity of VSRs.

4.1.3 Limitations of most controllers

Several kinds of controllers have proven effective for robotics tasks, including phase controllers (Joachimczak et al., 2016), central pattern generators (Kamimura et al., 2004), and ANNs (Talamini et al., 2019; Ferigo et al., 2022a). Nevertheless, the majority of them are disembodied and do not self-organize, suffering from limitations that we detail here.

First and foremost, they do not allow for the full exploitation of modularity and reusability. Consider the case of disassembling a VSR for the sake of transferring its components to other robots and reusing them. A disembodied controller, being tightly bound to the morphology, would require to be re-designed from scratch. Once transferred to a new morphology, controllers have to be learned all over again, resulting in the notorious "catastrophic forgetting" problem (French, 1999), i.e., controllers learned for one problem become ineffective when transferred to a new one.

Second, they usually do not self-organize their structure, shrinking the space of possible functions that can be explored. Self-organization is beneficial as it can discover emergent properties that cannot arise by a single intelligence only. In nature, self-organization provides mechanisms that evolution can exploit (Johnson and Lam, 2010) and instances of self-organizing intelligence are countless in biology.

Our controller addresses all of these shortcomings by its embodiment and self-organization properties, as we explain in the following.

4.1.4 Embodied, self-organizing neural controller

4.1.4.1 Design goals

Based on the considerations drawn in Section 4.1.3, we want our controller to be:

(a) Embodied: it has to be located in the robot body.

(b) Self-organizing: it must permit different degrees of complexity in different parts of the body. The actual distribution of complexity that fits a given task and morphology must emerge from optimization.

(c) Optimizable for a given task and morphology.

4.1.4.2 Definition

Based on the vast existing literature on ANNs, which are well-known universal function approximators (Goodfellow et al., 2016; Schäfer and Zimmermann, 2006), and considering previous applications of ANNs for controlling VSRs (Talamini et al., 2019; Ferigo et al., 2021; Medvet et al., 2021), we decided to rely on ANNs for building our embodied, self-organizing controller.

We represent the controller as a directed graph $G = (V, E)$ encoding an ANN where V are the nodes and E are the edges. The nodes of the graph are the neurons, and the edges of the graph are the synapses between them describing how computation flows over the ANN.

Each node is a tuple $v \in \mathbb{N} \times \{0, \dots, w-1\} \times \{0, \dots, h-1\} \times \mathcal{T}$, where the elements of the tuple constitute the node attributes that we denote using the following dot notation, for the sake of clarity. $v.\text{index} \in \mathbb{N}$ is the unique identifier for the node in the graph and its sole purpose is to formally permit the existence in V of nodes with the same values for all the other attributes. $v.\text{x} \in \{0, \dots, w-1\}$ and $v.\text{y} \in \{0, \dots, h-1\}$ are the coordinates of the voxel the node is placed into—recall that a VSR morphology is a grid of size $w \times h$. $v.\text{type} \in \mathcal{T} = \{\text{SENSOR}, \text{HIDDEN}, \text{ACTUATOR}\}$ is the neuron type. Each node of type SENSOR is statically associated with one element of one sensor placed in the same voxel of the node. Similarly, each node of type ACTUATOR is statically associated with the actuator of the voxel the node is in.

Each edge is a tuple $e \in V \times V \times \mathbb{R} \times \mathbb{R}$. $e.\text{source} \in V$ and $e.\text{target} \in V$ are the source and target nodes, respectively. $e.\text{weight} \in \mathbb{R}$ is the edge weight. $e.\text{bias} \in \mathbb{R}$ is the edge bias.

Since each one of the nodes of V is located in a precise voxel of the morphology, the controller is embodied, which meets our first design goal. As a further consequence, nodes and edges can be located more or less densely across the morphology, which meets our second design goal.

4.1.4.3 Computation

During the simulation, at each time step the controller reads the readings from the sensors and inputs them to SENSOR neurons, propagates the values across HIDDEN neurons, and outputs as actuation values the values that reach the ACTUATOR neurons.

In detail, the controller works as follows. Let $\text{in}(v) \subseteq E$ be the set of incoming edges of a node v. Let $h^{(k)}(v) \in \mathbb{R}$ be the *activation value* of node v at step k. Collectively, the activation values $\{h^{(k)}(v)\}_{v \in V}$ of all the neurons $v \in V$ constitute the *state* of the controller at step k. Initially, at $k = 0$, we set each activation value to 0. Then, we update the state as follows. If the node is a SENSOR neuron, then we set $h^{(k)}(v)$ to the tanh of the element of the current reading of the sensor associated with the node. Otherwise, we set the activation value to:

$$h^{(k)}(v) = \tanh\left(\sum_{e \in \text{in}(v)} h^{(k-1)}(e.\text{source}) \cdot e.\text{weight} + e.\text{bias}\right) \tag{4.1}$$

The concatenation of the activation values of the nodes of type ACTUATOR forms the actuation values $\boldsymbol{a}^{(k)}$ at step k.

In other words, at time step k every node sends the activation value to all its outgoing neighbors. At the next time step $k+1$, the neighbors, in turn, apply the activation function to it and save the result as the activation value for the next time step. Every edge propagates a message only once per time step, at every time step. Having no real incoming neighbors, a SENSOR node treats the sensor reading as if it were the message propagating from an incoming neighbor.

We remark that this controller is a dynamical system since there is a delay of one-time steps in the propagation of information along each edge. Moreover, we remark that the controller representation does not forbid cycles in the graph.

4.1.4.4 Optimization

We want to optimize a controller, i.e., a graph as defined in Section 4.1.4.2, for a given task and a given robot morphology.

Graphs are, in general, difficult to optimize: being the search space non-numeric, we cannot employ standard numerical optimization algorithms. We resort to EC for optimization. Indeed, EAs have already proven capable of dealing with graph-like structures (Miller and Harding, 2008; Stanley and Miikkulainen, 2002), provided that a fitness function and an appropriate representation are given. Additionally, EAs have also been argued to be a competitive alternative when optimizing ANNs for continuous control tasks (Such et al., 2017a). EC is thus suitable for achieving the third design goal.

We used the EA in Algorithm 2. It evolves a fixed-size population of n_{pop} individuals, i.e., graphs, initially set randomly, for a fixed number of n_{gen} generations.

At each generation, the current population is used to generate an offspring of n_{pop} new individuals that are then merged into the parent population. Each new individual G' is generated by applying mutation to a parent G selected with tournament selection with size n_{tour}. After parents and offspring have merged, the population for the next generation is selected by picking the n_{pop} best individuals, i.e., using truncation selection.

```
1  function evolve():
2      $P \leftarrow \text{initialize}(n_{\text{pop}})$
3      foreach $i \in \{1, \dots, n_{\text{gen}}\}$ do
4          $P' \leftarrow \emptyset$
5          foreach $j \in \{1, \dots, n_{\text{pop}}\}$ do
6              $G \leftarrow \text{selectTournament}(P, n_{\text{tour}})$
7              $G' \leftarrow \text{mutate}(G)$
8              $P' \leftarrow P' \cup \{G\}$
9          end
10     end
11     $P \leftarrow P \cup P'$
12     $P \leftarrow \text{truncate}(\text{sort}(P), n_{\text{pop}})$
13 end
```

Algorithm 2: The EA used in our experiments.

Since the controller consists of a graph, we used genetic operators suitable for this kind of representation. In particular, we used five mutation operators: edge mutation, edge addition, edge removal, node addition, and node removal. In line 7 of Algorithm 2, we beget an offspring G' by first copying the parent G and then applying one of the five mutation operators according to their relative probabilities p_{mut}, p_{edgeAdd}, $p_{\text{edgeRemove}}$, p_{nodeAdd}, and $p_{\text{nodeRemove}}$. In the following, we detail the five mutation operators:

Edge mutation. We randomly pick an edge $e \in E$ with uniform probability and mutate its weight and bias with additive Gaussian noise, i.e., $e.\text{weight} \leftarrow e.\text{weight} + \alpha$ and $e.\text{bias} \leftarrow e.\text{bias} + \beta$ with $\alpha, \beta \sim \mathcal{N}(0, \sigma^2_{\text{mut}})$.

Edge addition. We randomly pick two nodes $u, v \in V$ with uniform probability, such that $u.\text{type} \neq$ ACTUATOR and $v.\text{type} \neq$ SENSOR. Then, we add an new edge e from u to v, i.e., with $e.\text{source} = u$ and $e.\text{target} = v$, and we set $e.\text{weight}, e.\text{bias} \sim \mathcal{U}(-1, 1)$.

Edge removal. We randomly pick an edge $e \in E$ with uniform probability and delete it from the graph. If, after this operation, hidden nodes have no incoming and outgoing edges left, we remove them from V to decrease the redundancy of the representation (Rothlauf and Goldberg, 2003; Rothlauf, 2006).

Node addition. We randomly pick two nodes $u, v \in V$ with uniform probability, such that $u.\text{type} \neq$ ACTUATOR and $v.\text{type} \neq$ SENSOR. We then create a new hidden node w and set $w.\text{x}$ and $w.\text{y}$ to be the coordinates of a voxel picked at random. Finally, we add an edge e from u to w and an edge e' from w to v and we set $e.\text{weight}, e.\text{bias}, e'.\text{weight}, e'.\text{bias} \sim \mathcal{U}(-1, 1)$.

Node removal. We randomly pick a hidden node $v \in V$ with uniform probability and delete it from the graph, together with all its incoming and outgoing edges. If, after this operation, hidden nodes have no incoming and outgoing edges left, we remove them from V.

We initialize each individual of the initial population as follows. We add to v a sensor node for each sensor reading element in the given morphology, associate v the node with it, and set $v.\text{x}$ and $v.\text{y}$ to the coordinates of the voxel hosting the sensor. We add to V an actuator node v for each voxel in the given morphology, associate v the node with the actuator, and set $v.\text{x}$ and $v.\text{y}$ to the coordinates of the voxel. For each actuator node v and each sensor node u in the same voxel, we add to E an edge e from u to v and we set $e.\text{weight}, e.\text{bias} \sim \mathcal{U}(-1, 1)$.

As a result of this initialization procedure, no edges are encroaching on different voxels for the VSRs of the very first generation, and the controllers can be regarded as being minimal. We adopt this initialization strategy in line with the complexification principle of starting minimally and incrementally growing topologies of ANNs (Stanley and Miikkulainen, 2002).

We remark that the mutation operators that affect the topology of the graph, i.e., edge and node addition (or removal), by changing the preference for edges inside or outside modules, can affect the degree of modularity of the controller. This consideration will reveal itself useful in Section 4.1.5.2.

We used the same parameter values for all experiments, with values determined after preliminary experiments, except for n_{gen} that is detailed separately for every experiment. After preliminary experiments and relying on our previous experience, we set $n_{\text{pop}} = 96$, $n_{\text{tour}} = 5$, $\sigma_{\text{mut}} = 0.7$, $p_{\text{mut}} = 0.5$, $p_{\text{edgeAdd}} = 0.1$, $p_{\text{edgeRemove}} = 0.1$, $p_{\text{nodeAdd}} = 0.15$, and $p_{\text{nodeRemove}} = 0.15$.

4.1.4.5 Advantages and limitations of the embodied, self-organizing controller

The controller presented in this section addresses the design goals presented in Section 4.1.4.1 and has several advantages.

First, as required by the first design goal, it is *embodied*, as the nodes are precisely located throughout the VSR body. By this property, the VSR brain can be partitioned into subgraphs (by selecting subsets of nodes and the edges incident to them), and each of the subgraphs can then be traced back to the portions of the body it belongs to. At the same time, should the VSR be dissected into physical modules (e.g., for the sake of transferring it to assemble a new VSR), each physical module would then be able to refer to the subgraph of the controller it was the embodiment for. As a result, each subgraph (once transferred with its physical counterpart) still finds itself in a body it is acquainted with, and is well-positioned to exploit the representations learned during its "previous life". Other studies have indeed highlighted how modular controllers are less prone to catastrophic forgetting (Ellefsen et al., 2015). Since functional modules, i.e., units sharing the same function or purpose, can arise in embodied agents, the representations learned locally in a given functional module could well be transferred to a new module performing the same, or a similar, function.

Since both the topology and the parameters can freely evolve in the body, our controller is *self-organizing* and thus satisfies the second design goal. This fact bears an important consequence: emergent properties, that would not arise without self-organization, can be observed. This is a consequence of the expressiveness of our representation: the search is performed in the space of every possible topology (and parametrization) encompassing a given morphology. We remark that there is no upper (or lower) bound on the length (measured, for example, in terms of voxels to be crossed) of the edges. Potentially, interesting long-range synapses can be established between the neurons of the controller. Remarkably, self-organization can also be biased purposely to explore some portions of the search space more densely, to ease the emergence of some properties we deem interesting a prior (Templier et al., 2021). Moreover, self-organization could be used as a way of testing how the "brain" of a robot adapts to the sensors and the robot shape, and make parallels with neuroscience. As intriguing as it may be, we did not explore this last topic further, and leave it for future work.

Third, messages propagate over the graph one hop at a time, and these delays could be exploited by the optimization algorithm to introduce timed dependencies that would not be possible otherwise. As a proof of concept, Cheney et al. (2014a) demonstrated the benefits of having messages propagating "physically" across a VSR body, and the claim is biologically grounded (Segev and Schneidman, 1999). Moreover, we remark our model is *stateful*, and could thus show interesting temporal dynamic behavior, as it couples two dynamical systems, the mechanical one and the controller one. Contrarily, a stateless controller (without cycles and delays) couples its static system with only the mechanical dynamical system.

Fourth, *distributing* the controller delivers advantages on its own. Distributed controllers do not suffer from having a single-point-of-failure, and would thus prove resilient to both exogeneous (e.g., environmental changes) and endogeneous (e.g., malfunctions) threats once deployed *in vivo*. As a biological counterpart, animals that excel in regenerating their amputated limbs, like sea stars (Lawrence, 2020), usually have a distributed nervous system[1]; some species can even beget stand-alone new individuals from fragments of their bodies, like flatworms of the genus *Planaria* do (Gentile et al., 2011). Although there is evidence that these properties are due to pluripotent stem cells (Baguna et al., 1989), having a less centralized nervous system intuitively eases regeneration.

We remark that the way we model the embodied, self-organizing controller in this study also implies a few limitations.

First, the location of nodes within the voxel is relevant only for nodes of types ACTUATOR and SENSOR. Since there are no differences in the way the activation value is computed between a pair of nodes located in the same or different voxel, a controller is in practice functionally invariant to the location of HIDDEN nodes. To verify the practical impact of this modeling choice, we conducted an experimental campaign where we explicitly took into account the location of HIDDEN nodes as follows. We modified the edge addition genetic operator in such a way that long connections had a lower probability of being added: as a consequence, the location of all nodes—including HIDDEN nodes—matter when evolving the controller topology. We do not present the full results for brevity, but we found that the controllers that evolved in this way were in general worse than those that evolved without considering node location. As an alternative, we could take into account node location by introducing longer delays in information spreading along long connections: we leave this possibility for future work.

[1]Notable exceptions being salamanders (Joven et al., 2019) and axolotls (Roy and Gatien, 2008).

Second, we do not model the cost of complex and large controllers. As a consequence, no factors are driving the evolution towards parsimony: we expect the complexity of the network to increase virtually unbounded during the evolution. This is what we observed in our experiments. While we acknowledge that the physical realization of arbitrarily complex neural controllers might be unfeasible, the size of the controllers we evolved in this study is in practice small, in the order of a few hundred nodes and edges.

4.1.5 Experimental evaluation

We performed several experiments aimed at answering the following research questions:

RQ1 Is the proposed controller effective?

RQ2 Does it enable module transferability? Can transferability be favored or disadvantaged?

RQ3 Can we automatically discover modules?

To answer these questions, we performed several experiments with two different VSR morphologies, depicted in Figure 4.1. We experimented with a 4×3 (size of the voxel grid constituting the shape) rectangle with a 2×1 rectangle of missing voxels at the bottom-center, that we call *biped*, and with a 6×2 rectangle, that we call *worm*. We use similar sensor configurations for the two shapes: area sensors for every voxel, touch sensors for the voxels in the bottom row of the shape, velocity sensors for the voxels in the top row of the shape, and lidar sensors for the voxels in the rightmost column of the shape. This sensor configuration results in an overall number of $31 = 10 \cdot 1 + 2 \cdot 1 + 4 \cdot 2 + 3 \cdot 5$ sensor nodes and 10 actuator nodes in the controllers for the biped and $40 = 12 \cdot 1 + 6 \cdot 1 + 6 \cdot 2 + 2 \cdot 5$ sensor nodes and 12 actuator nodes in the controllers for the worm.

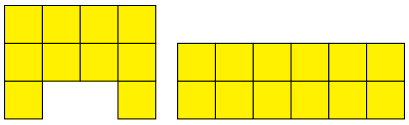


Figure 4.1: The two morphologies used in our experiments.

For all the experiments, we considered the task of locomotion. The goal of the VSR is to travel as far as possible on a terrain along the positive x direction in a fixed amount of simulated time. The fitness of the individual is the average velocity, measured as:

$$\bar{v}_x = \frac{x_c(t_{\text{final}}) - x_c(t_{\text{transient}})}{t_{\text{final}} - t_{\text{transient}}}, \tag{4.2}$$

where $x_c(t)$ is the x-position of the center of mass of the VSR at time t. We set $t_{\text{final}} = 30\,\text{s}$ and $t_{\text{transient}} = 5\,\text{s}$ to discard the initial transitory phase and avoid deceptive and inconclusive behaviors (Whitley, 1991). Locomotion is a classic task in evolutionary robotics and usually consists of making the robot run along a flat surface. We here used instead an uneven (hilly) terrain with bumps. The bumps have an average height of 1 m (3 m being the side length of a voxel) and an average distance of 10 m. Moreover, we used a different hilly randomly generated terrain at every fitness evaluation, to prevent the robots from "overfitting" to a single terrain profile, making adaptation more challenging.

We implemented the software for the experimental evaluation in Java, building on two frameworks: JGEA[2] for the evolutionary optimization and 2D-VSR-Sim[3] (Medvet et al., 2020b) for the simulation of VSRs. In the simulations, we set the time step to $\Delta t = \frac{1}{60}\,\text{s}$ and all other parameters to default values. The code for the experiments is publicly available at `https://github.com/pigozzif/SelfOrganizingVSRController`.

For each experiment, unless otherwise specified, we performed 10 evolutionary runs by varying the random seed for the EA. We remark that each simulation is instead

[2]`https://github.com/ericmedvet/jgea`
[3]`https://github.com/ericmedvet/2dhmsr`

deterministic, given a terrain and a VSR. After verifying the adequate hypotheses, we carried out all statistical tests with the Mann-Whitney U rank test for independent samples (Mann and Whitney, 1947) using, unless otherwise specified, 0.05 as the confidence level.

4.1.5.1 RQ1: effectiveness of evolved controllers

We say that a controller is effective if it meets two criteria: (a) it can successfully solve the task at hand and (b) it is more complex than the minimal controller. The latter point is crucial, since a controller that has not evolved to be more complex than the minimal controller could not be considered "self-organizing", invalidating all of our conjectures. We thus introduce a simple, yet effective measure of controller complexity that measures its *size*:

$$|G| = |V| + |E| \tag{4.3}$$

that is the sum of the number of nodes and edges of the graph. Concerning the success in solving the task, we measure it using the robot average velocity $\bar{v}_x$, as defined in Equation (4.2).

We ran an experimental campaign with 10 runs lasting $n_{\text{gen}} = 3000$ generations each, and for the two shapes introduced previously.

Figure 4.2 shows, for the two shapes, the best individual velocity $\bar{v}_x$ and controller size $|G|$ during the evolution. For both indexes, the figure shows the median $\pm$ standard deviation across the 10 runs.

Table 4.1 reports the results in terms of $|G|$, and its components $|V|$ and $|E|$, for the best individual at the last generation, separately for the two shapes, and compares them with the values at the start of evolution. Recall that, for a given shape, $|G|$, $|V|$, and $|E|$, are the same for every individual in the first generation (given a morphology). For the indexes at the last generation, the table shows the median and standard deviation across the 10 runs.

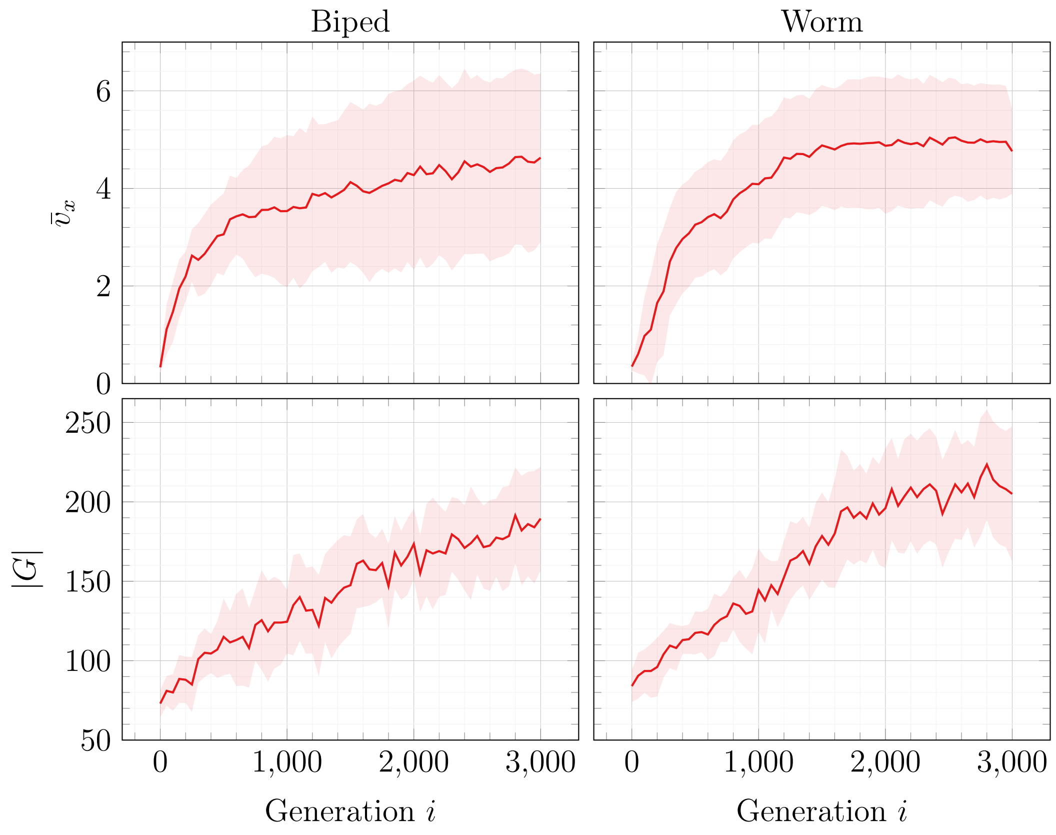


Figure 4.2: Median ± standard deviation (solid line and shaded area) across the 10 runs of the robot velocity $\bar{v}_x$ (top) and controller size $|G|$ (bottom) for the best individual, for biped (left) and worm (right).

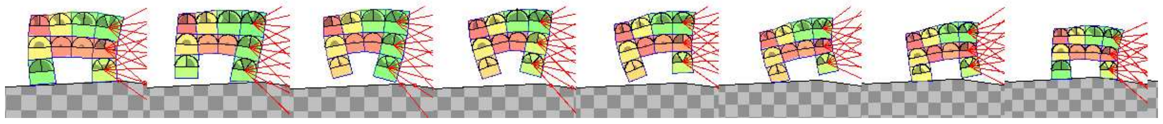
	Initial			At last generation		
Shape	$\|V\|$	$\|E\|$	$\|G\|$	$\|V\|$	$\|E\|$	$\|G\|$
biped	45	35	80	88.5±17.3	96.5±26.8	179.5±42.2
worm	52	40	92	99.0±16.6	115.0±24.8	216.0±39.4

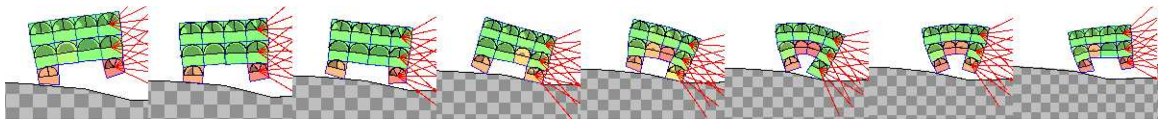
Table 4.1: Median and standard deviation of number of nodes $|V|$, number of edges $|E|$, and controller size $|G|$ for the best individual, at initialization, and the last generation.

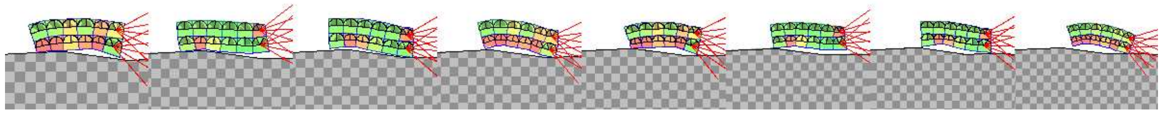
From the table and the figure, we observe that the evolved VSRs turn out to be effective according to our criteria. First, evolution is capable of finding good solutions to the locomotion task. The best individuals run on average at $\bar{v}_x = 5\,\mathrm{m/s}$ at the end of evolution, against approximately $0.5\,\mathrm{m/s}$ at the beginning, signifying a considerable gain in locomotion skills. We visually inspected the behaviors and found them to be highly effective for locomotion on a hilly terrain. As a proof of concept, Figures 4.3a to 4.3d are time-lapse images for the movement of two bipeds and two worms chosen randomly among the 10 best individuals resulting for each of the two shapes. As can be seen from the pictures, bipeds hop on their legs, and worms inch forward as a caterpillar would do. To appreciate even more the adaptive behaviors exhibited by the robots, the videos of all the 10 best bipeds and worms can be found respectively at `https://youtu.be/-ECoa1tffok` and `https://youtu.be/jADw6Qf70g0`. As a side comment, we found the gaits evolved with the self-organizing controller to be particularly fluid when compared to our previous works (Talamini et al., 2019; Medvet et al., 2020a, 2021).

Second, the evolved controllers do increase well beyond their initial size during evolution. The numbers reported in Table 4.1 imply there is a 149 % median increase in $|G|$ for the biped shape, and a 135 % median increase for the worm shape. Figure 4.2 corroborates this finding by showing an upward trend in the $|G|$ lines over the evolution (right plot), even though it appears to decelerate toward the end. At the same time, there does not seem to be an unexpected difference in contribution to $|G|$ between nodes and edges (the latter being slightly more abundant than the former).

By manually inspecting the evolved controllers, in particular the locations of nodes and edges over the body, we noticed some patterns. Most notably, long-range edges do arise. For example, numerous edges connect the front part to the rear part of many worms. Intuitively, they establish a feedback mechanism that is necessary to inch forward as a caterpillar would do. Another recurring feedback mechanism evolved between the two legs of bipeds, contributing to their signature gait (i.e.,


(a) Biped (video available at `https://youtu.be/qkVa9hiOveI`)


(b) Biped (video available at `https://youtu.be/oYeaZwtR8OI`)


(c) Worm (video available at `https://youtu.be/k1f2vQNyrEU`)

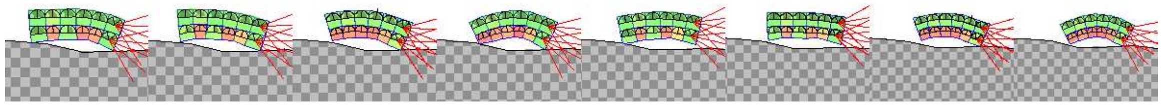

(d) Worm (video available at `https://youtu.be/HaZHnaRIPqA`)

Figure 4.3: Time-lapse showing locomotion for two bipeds and two worms chosen randomly among the 10 best individuals resulting for each of the two morphologies.

hopping between the front and the rear leg). These facts altogether point out that self-organization is indeed at work.

By looking closer at Figure 4.2, it looks like velocity reaches a plateau well before controller growth stops. Indeed, there is no evidence controller growth stops at all. It might be the case that there exists some sort of "critical mass" of the controller such that, after a local optimum in the fitness space is reached, evolution keeps adding redundant genetic material (i.e., neurons and edges that do not necessarily contribute to the locomotion skills of the VSR). Since we do not disfavor in any way the addition of new edges and nodes that do not modify the functionality of the overall controller, we can speculate we are witnessing an instance of the bloat phenomenon (Silva and Costa, 2008) observed in the evolution of computer programs (also known as genetic programming, see Koza (1993)). It has been argued that bloat is beneficial to the individual as it provides a buffer against the deleterious effects of mutation and recombination (López et al., 2011). Interestingly, there is a rich body of literature in biology on the benefits of neutrality, after the work of Kimura (1979).

But then, does self-organization matter? In other words, if (we speculate) bloat is at work, it might be that all those hidden neurons and edges are just excess genetic material, and evolution is simply optimizing the set of initial edges (those between sensors and actuators) and their parameters. To shed light on this argument, we performed an ablation study, aiming at exploring what happens if we evolve only the weights and biases and not the topology of the controller (which is stuck to be the initialization topology, see Section 4.1.4.4). In particular, we reset $p_{\text{mut}} = 1$ and all other mutation probabilities to 0. Figure 4.4 summarizes the results by showing the boxplots for the distribution of the $\bar{v}_x$ of the best individuals at the last generation for the two cases: with and without topology optimization. Interestingly, the distributions are radically different. We carried out a one-sided Mann-Whitney U test with the null hypothesis that, for each shape, the median of the best fitness for evolving the topology is greater than the median best fitness for not evolving the topology. We found that the null hypothesis can be rejected at the 0.05 significance

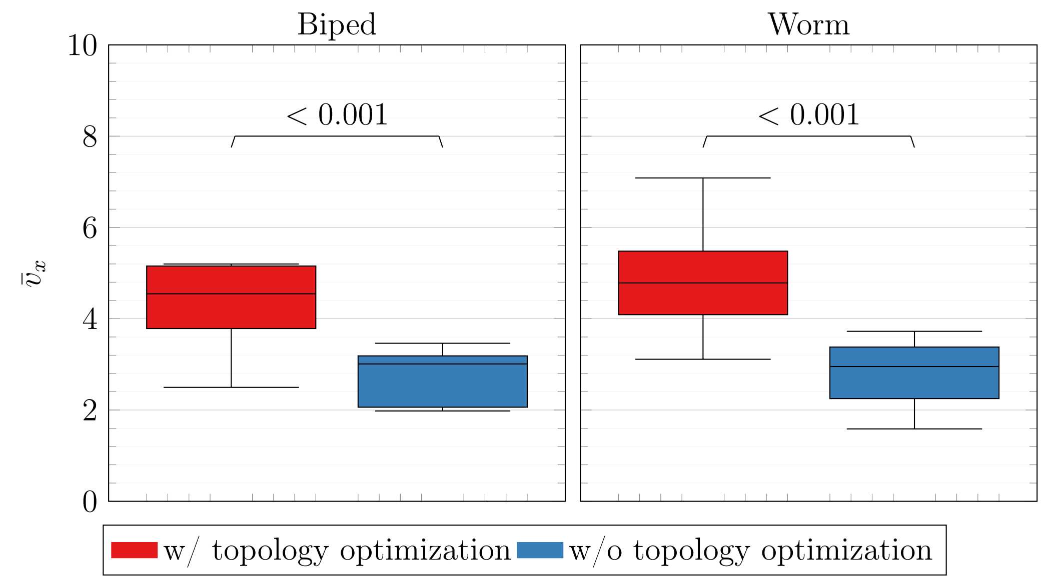


Figure 4.4: Boxplots for the distribution of the velocity $\bar{v}_x$ of the best individuals at the last generation, for biped (left) and worm (right), obtained with or without topology optimization. The bars above pairs of boxes show the corresponding p-value. level (p-value < 0.001 for both shapes). Even though this result is valid only for this kind of initialization, it still suggests that self-organization does indeed benefit evolution.

4.1.5.2 RQ2: module transferability

Because of the way we defined the controller, specifically, since we put nodes inside voxels, it is clear that it can *by-design* be disassembled together with the morphology. Nevertheless, achieving module transferability in practice requires also a reassembly phase, in which disassembled modules are reused for building new VSRs. We hence define a way of transferring modules, using a disassembly-reassembly procedure, and a way for measuring the effectiveness of the transfer.

4.1.5.3 Disassembly-reassembly procedure

Let r_1, r_2 be two *donor robots* that have been optimized for a given task. We assume that each of the robots can be disassembled in at least two modules, a *module* being a connected subset of robot voxels. The disassembly-reassembly procedure consists of building a new, reassembled robot r' by combining at least one module from the first robot and at least one module from the second robot. We measure the *transferability* of the two modules used to build r' as the relative ability of r' to solve the task concerning the ability of the donors r_1, r_2. We remark that this definition of transferability is somehow limited, as it does not measure the degree to which robot modules can be re-used for arbitrarily different tasks. Nevertheless, we believe that the possibility of reusing modules for the same task is indeed useful (and, hence, it is important to quantify this possibility): in the long term, re-using parts of disposed robots might be an enabling factor for fully autonomous robotic ecosystems (Hale et al., 2019b).

In the specific scenario of locomotion and with robots equipped with our controller, we cast this general definition as follows. When disassembling a robot, for each resulting module we drop all the edges whose source or target nodes are not located in voxels belonging to the module. When reassembling a robot from existing modules, we do not add any new edge between nodes located in different modules—in some preliminary experiments, we explored other options, e.g., "rewiring" crossing edges at random, but we found they do not deliver any advantage. We measure transferability as:

$$\rho = \frac{\bar{v}_{x,r'}}{\frac{\bar{v}_{x,r_1}+\bar{v}_{x,r_2}}{2}}, \tag{4.4}$$

where $\bar{v}_{x,r_1}$, $\bar{v}_{x,r_1}$, and $\bar{v}_{x,r'}$ are the velocities of the three robots.

Since some edges of the controller get dropped in the disassembly-reassembly procedure, before measuring the velocity $\bar{v}_{x,r'}$ of the reassembled robot r', we re-optimized it using the same EA used for from-scratch optimization (see Algorithm 2), yet starting from a different initial population. This re-optimization is of crucial importance since we want the new VSRs to be effective and we expect the reassembled robot to benefit from a reasonable amount of fine-tuning. The overall goal, however, is to make reassembly of pre-optimized modules cheaper than optimizing from scratch.

In detail, the initial population of the re-optimization is composed of the controller G' of the reassembled robot r' and $n_{\text{pop}} - 1$ mutations of G' obtained by applying the same mutation operators (with the same probabilities) of the optimization step (see Section 4.1.4.4).

We remark that the disassembly-reassembly procedure requires to (a) define a partitioning of the two donor robots r_1 and r_2, (b) select one or more modules of r_1 and one or more modules of r_2, and (c) define a way to combine the selected modules. It is evident that module transferability strongly depends on these three key choices. As an example, consider the case in which two "trunks" coming from two-legged robots with different morphologies, each with a trunk and a few legs, are glued together: it is very unlikely that the resulting "trunk-only" robot will be effective in locomotion since it has no legs. The representation of the controller on transferability, hence, clearly plays a secondary role. In the next sections, for answering RQ2, we manually choose reasonable options for these three choices. Later, in Section 4.1.5.7, we show how our representation may be helpful while coping with the first choice (partitioning a robot in modules) automatically.

Kriegman et al. (2019) considered a similar case of damage recovery, but focused on the morphology rather than the control: they evolved shape changes to recover the lost function after physical damage, with no adaptation to the control policy.

4.1.5.4 Experimental procedure

We considered the two morphologies of the previous experiments and manually partitioned them into modules as shown in Figure 4.5. Then, we proceeded as follows.

First, we performed 10 evolutionary runs for each of the two morphologies, obtaining ten bipeds and ten worms. Then, we performed two types of reassembly for each morphology: a homogeneous one, where modules of the reassembled robots come from robots with the same morphology, and a heterogeneous one, where modules

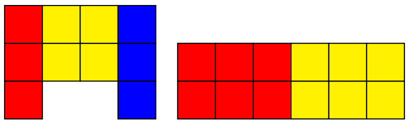


Figure 4.5: Our manual partitioning in modules (one color for each module) of the two morphologies.

came from robots with the other morphology.

In the homogeneous case, we proceeded as follows. For the biped and each $i \in \{1, \dots, 10\}$, we reassembled a new biped by taking the red and blue modules (the "legs") from the i-th biped (i.e., the best individual obtained at the end of the i-th run) and the yellow module (the "torso") from the $((i+1) \bmod 10)$-th biped. Similarly, for the worm and each $i \in \{1, \dots, 10\}$, we reassembled a new worm by taking the red module (the "back half") from the i-th worm and the yellow module (the "front half") from the $((i+1) \bmod 10)$-th worm.

In the heterogeneous case, we proceeded as follows. For the biped and each $i \in \{1, \dots, 10\}$, we reassembled a new biped by taking the red and blue modules from the i-th biped and the yellow module from the i-th worm. For the worm and each $i \in \{1, \dots, 10\}$, we reassembled a new worm by taking the red module from the i-th worm and the yellow module from the i-th biped.

After reassembly, we re-optimized each resulting reassembled robot with the EA of Algorithm 2 with the population initialization procedure modified as described in Section 4.1.5.3. For the re-optimization, we set $n_{\text{gen}} = 510$, a computational budget which is remarkably lower than the one used for optimization ($\approx 17\,\%$).

Upon re-optimization, we measure transferability ρ as defined in Equation (4.4). We also count how many edges were cut during disassembly, and compute the sum of their weights and biases in absolute value, as proxies of how destructive that operation is. To make figures comparable, we cast these indexes as relative to their corresponding values before the disassembly. We denote by η_{num} the ratio between the number of edges dropped from a module upon disassembly and the overall number of edges

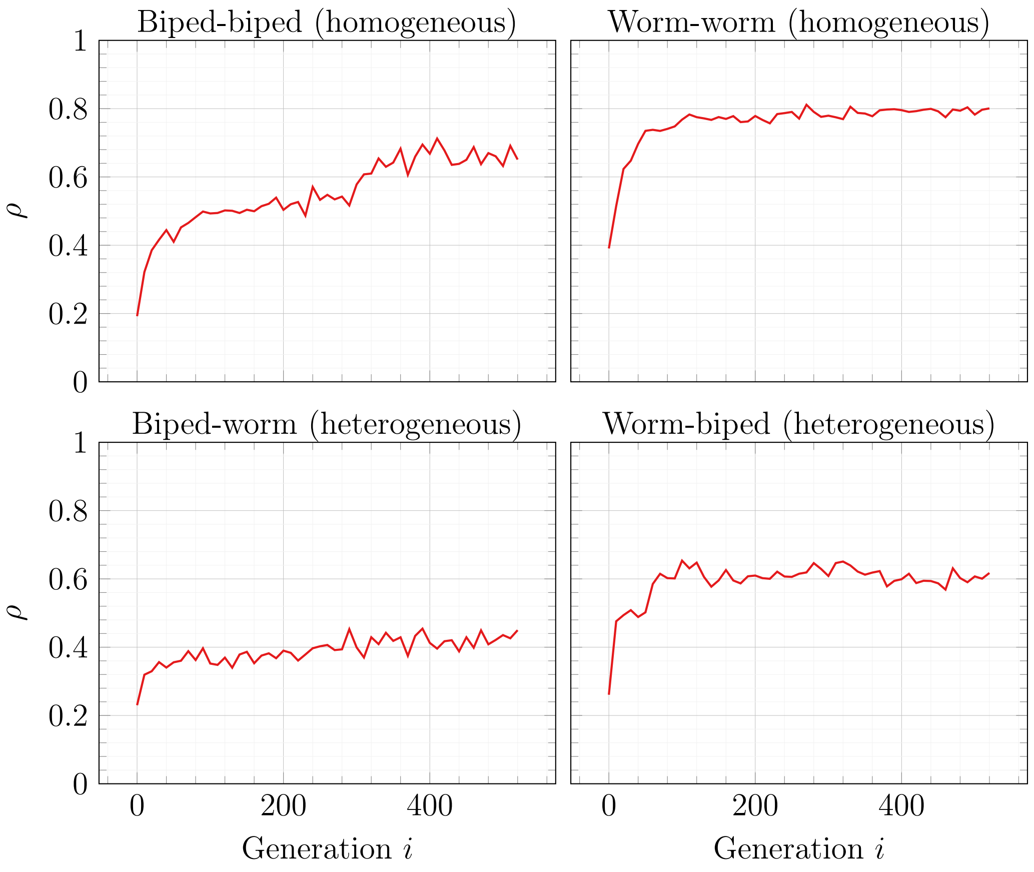


Figure 4.6: Median ρ, across the 10 reassembled robots for each morphology and reassembly type, during the re-optimization for four combinations.

in the module (before disassembly). Similarly, we denote by η_{weight} the ratio between the sum of weight and bias (in absolute value) of edges dropped from a module upon disassembly and the overall sum of edges in the module (before disassembly).

In the next subsection, we present the results and comment on them.

4.1.5.5 Results

Figure 4.6 reports the median (across the 10 reassembled robots for each morphology and each reassembly type, i.e., homogeneous and heterogeneous reassembly) ρ during the re-optimization for four combinations. We remark that a value of $\rho = 1.0$ (on the y-axis) corresponds to fully recovering the average velocity of the donors.

Our disassembly-reassembly procedure partially succeeds in recovering the lost functionality, in the homogeneous as well as in the more challenging heterogeneous combinations. Due to the intrinsic hardness of the heterogeneous reassembly case, and for the sake of brevity, in the following, we discuss only the homogeneous case.

The magnitude of the effect differs by morphology; bipeds hover above 60 % and worms just fall short of 80 %. Nevertheless, we are far from recovering (let alone outclassing) the average velocity $\bar{v}_x$ of donors. Arguably, the morphology also affects transferability, with more "primitive" shapes like worms being advantaged.

To explain why recovering is not complete, we looked at the values of η_{num} and η_{weight}, which measure the impact of disassembly in terms of edges being cut. Median values are $\eta_{\text{num}} = 0.60$ and $\eta_{\text{weight}} = 0.56$, for the biped, and 0.45 and 0.38 for the worm. In other words, the disassembly of a robot in modules is a very destructive operation for the VSR controller and the re-optimization struggles in recovering the lost structure. Interestingly, values of η_{num} and η_{weight} are lower for the worm (fewer edges are cut) and this is somehow reflected in the value of ρ, which is greater for the worm than for the biped.

4.1.5.6 Fostering modularity

Having observed that the removal of edges crossing modules appears to be detrimental to transferability, we designed a variant of the EA for evolving controllers that is aimed at discouraging the use of edges crossing modules. In other words, with this variant, we can foster the modularity of the controller.

In detail, the EA is the same as the one presented in Section 4.1.4.4, except for two mutation operators. We introduce a *biased edge addition* operator, and a *biased node addition* operator, that substitutes the original edge addition and node addition operators (edge removal and node removal are left untouched, as well as the mutation probabilities).

In the biased edge addition operator, when picking the pair of nodes u, v to be connected by the new edge (see Section 4.1.4.4), instead of using uniform probability, we pick pairs whose nodes are in the same module with a probability that is ω larger than the probability of picking nodes that are in different modules.

In the biased node addition operator, the choice of nodes u, v works as above; the new node w is placed in a random voxel of the module, if u, v are in the same module, or a random voxel of the robot, otherwise.

The role of the parameter ω is to determine the degree of preference toward modularity. With $\omega > 1$ we foster modularity, with $\omega < 1$ we discourage modularity, with $\omega = 1$ we are neutral, i.e., we use the same operators of the original EA of Section 4.1.4.4.

While designing this variant, i.e., the two biased mutation operators, we got inspiration from nature. In fact, Wagner et al. (2001) suggests that only two processes can foster modularity in biological systems: parcellation and integration. The latter consists of the selective acquisition of genes that map to phenotypic traits belonging to the same module. In other words, modular phenotypic traits are favored by evolution. In light of this consideration, biased edge addition and biased node addition, by favoring connectivity within modules, can be seen as a simplified form of integration. The same processes (of parcellation and integration) have been fruitfully exploited for modular ANNs in Mouret and Doncieux (2009), but dealing with a different set of tasks and not under an embodiment perspective.

For assessing the effectiveness of this variant in fostering modularity, i.e., favoring module transferability, we performed an experimental campaign following the same procedure described in Section 4.1.5.4 with three EAs: the original one ($\omega = 1$, "neutral" in the figures), one with $\omega = 10$ ("more modularity" in the figures), and one with $\omega = 0.1$ ("less modularity" in the figures), the latter as a sort of control group.

Figure 4.7 summarizes the main outcome of this experimental campaign by reporting the median (across the 10 reassembled robots for each morphology) ρ during the re-optimization for the two morphologies and the three variants.

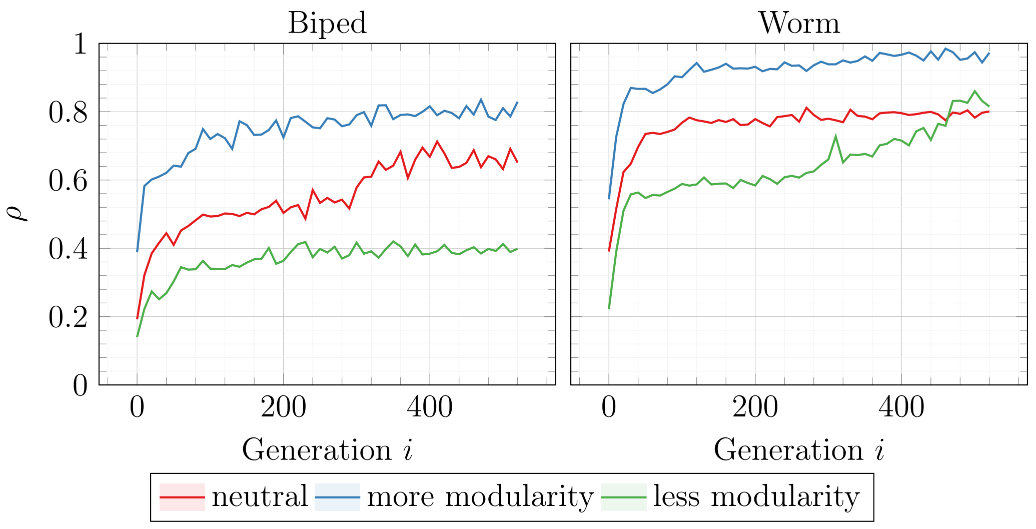


Figure 4.7: Median ρ, across the 10 reassembled robots for each morphology, during the re-optimization for biped (left) and worm (right) and for the three EA variants (line color).

The foremost observation is that the two new variants perform as expected—the result of the "neutral" variant is, obviously, the same as in the previous experiments. There seems to be a stark contrast between the median profile of ρ of the different variants. Not only do modular controllers, i.e., those obtained with $\omega = 10$, recover more quickly than neutral ones, i.e., those obtained with $\omega = 1$, but they also appear to be fitter at the very end of evolution. The same holds when comparing neutral controllers with their non-modular counterparts, i.e., those obtained with $\omega = 0.1$. Statistical tests comparing median ρ across the evolutionary runs showed the differences at the end of re-optimization are indeed significant, except for the neutral vs. modular and neutral vs. non-modular pairings for the worm shape.

We hypothesize that, intuitively, the observed differences are due to the different number of edges that are cut upon disassembly for the three variants, that we report in Figure 4.8. It can be seen from the figure that modular controllers stay in the range of 0.25–0.5 of edges cut (η_{num}), while the proportions are higher for the neutral and non-modular variants. The plots for η_{weight} corroborate this interpretation. Statistical tests, whose outcomes are reported in Figure 4.8, support our claim

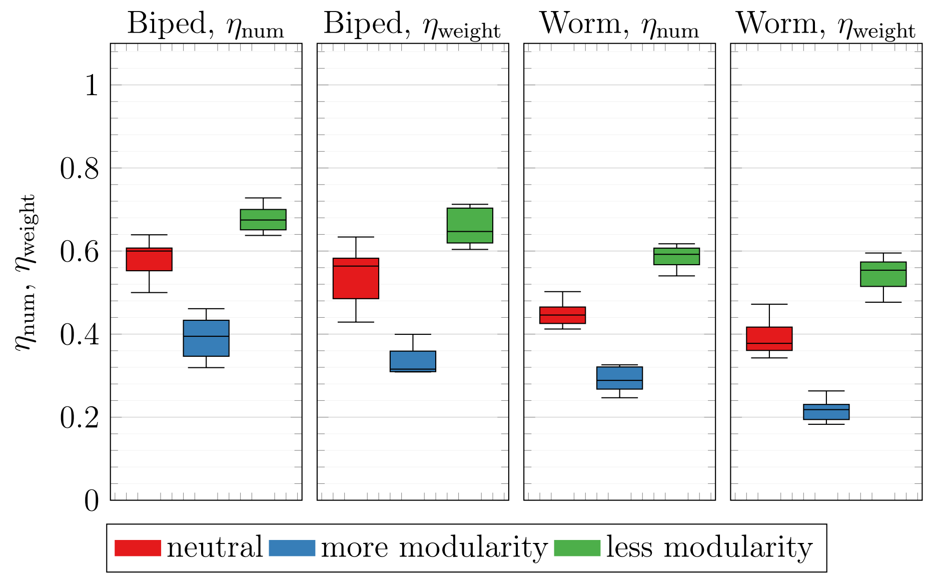


Figure 4.8: Boxplots for the distribution of η_{num} of η_{weight} (rate of edges cut upon disassembly) for the two morphologies and the three EA variants. All p-values are significant.

that the three variants are different in terms of η_{num} and η_{weight}.

As an aside, by looking at the raw values of $\bar{v}_x$ after the re-optimization, we observe that the magnitude of the loss for the value of the donor robots varies with the morphology, being less pronounced for worms, and more evident for bipeds and that worms have, on average, fewer edges cut. Undoubtedly, the configuration influences how large the gap between bipeds and worms, with modular biped controllers being more or less at the same level as their worm counterparts, whereas we witness a 30% hiatus with non-modular controllers. For this, we conjecture the reason to be that the long-range edges (non-modular controllers are biased toward) bear more importance in bipeds rather than worms. This fact is meaningful if we consider the gait of bipeds and the many long-range edges that evolve between the two legs (see Section 4.1.5.1).

To gain further insights into the three variants, we compared them in terms of the outcome of the first optimization, i.e., we looked at the values of $\bar{v}_x$ and $|G|$ obtained before the disassembly-reassembly procedure. Figures 4.9 and 4.10 summarizes the results concerning these indexes, respectively as the median values for $\bar{v}_x$

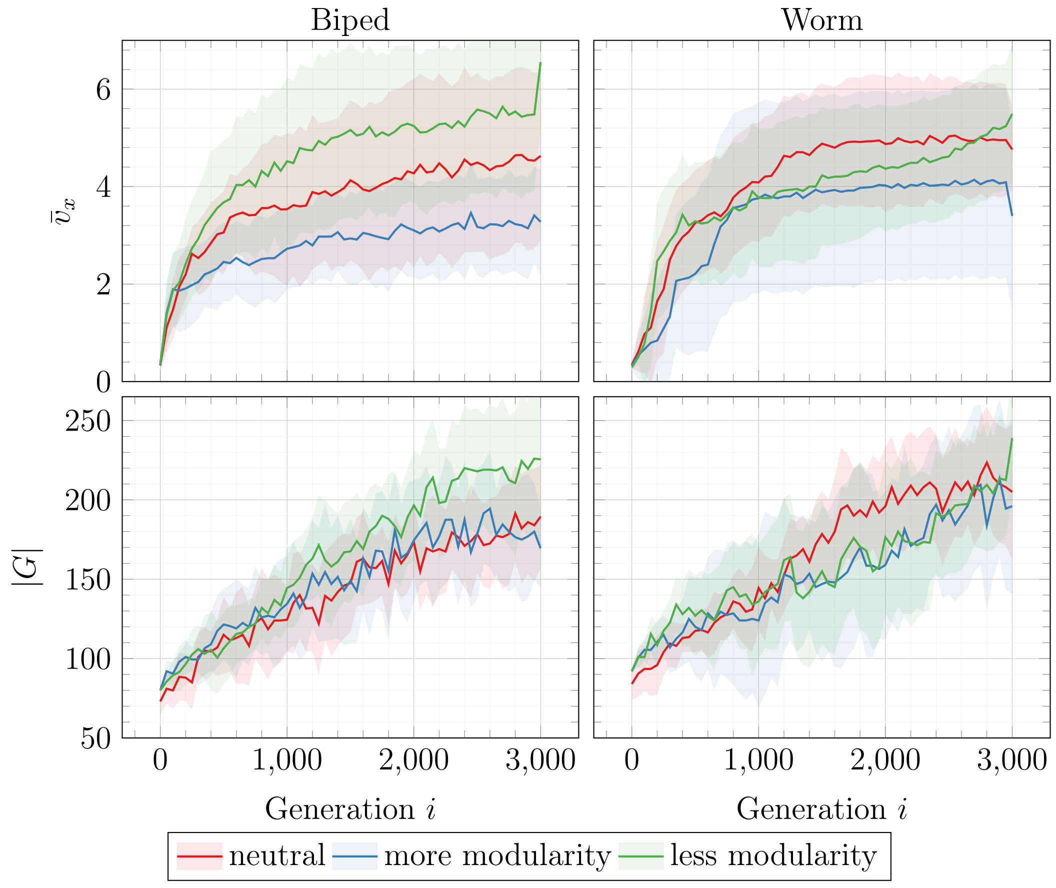


Figure 4.9: Median ± standard deviation (solid line and shaded area) across the 10 runs of the robot velocity $\bar{v}_x$ (top) and controller size $|G|$ (bottom) for the best individual, for biped (left) and worm (right) and for the three EA variants.

and $|G|$ during the evolution and as the boxplots for the distributions of the value of $\bar{v}_x$ of the best individuals at the end of the evolution.

It can be seen that, for what concerns the size $|G|$ of controllers, the three variants exhibit negligible differences. From another point of view, favoring or discouraging modularity does not impact the self-organization of the controller.

Concerning the velocity $\bar{v}_x$, Figure 4.9 seems to suggest that there are some differences. In particular, it looks like modular evolved VSRs are, on average, less fit than neutral ones, which in turn seem to be, on average, less fit than their non-modular counterparts. Despite those differences are not statistically significant (see the p-values in Figure 4.10), we think they can be explained by the role of long-range

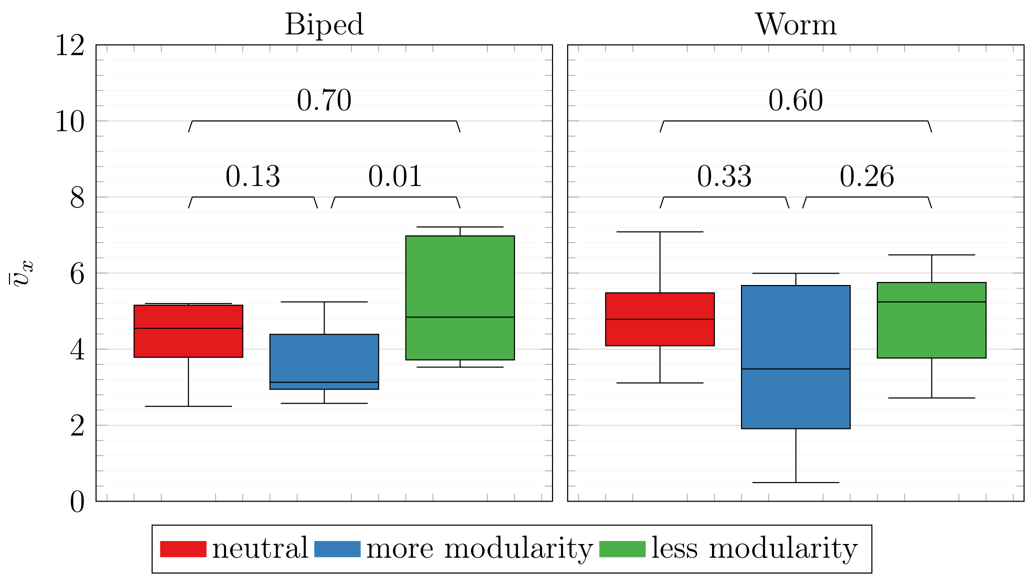


Figure 4.10: Boxplots for the distribution of the velocity $\bar{v}_x$ of the best individuals at the last generation, for biped (left) and worm (right), obtained with the three EA variants. The bars above pairs of boxes show the corresponding p-value.

edges, i.e., edges crossing several voxels. Non-modular controllers might outperform the others because their long-range edges (they are implicitly biased toward) carry "more" information. More broadly, the effectiveness of a procedure for favoring or disfavoring modularity might depend on the choice of how to split the robot into modules. However, we here compared EA variants using the same choice for modules and we do not have any argument for hypothesizing that a different choice might be better for favoring or disfavoring modularity.

To summarize, two conclusions can be made. First, in line with the embodied cognition paradigm, morphology impacts transferability. Second, after disassembly-reassembly, we do not always fully recover the lost functionality of the donor robots. In particular, the degree of modularity does impact the transferability of an embodied controller. There are, however, two caveats: (a) modularity appears to negatively affect average velocity $\bar{v}_x$ before disassembly and (b) favoring modularity requires the manual definition of modules before the optimization. The latter point is undeniable, but we will next introduce a procedure for the automatic discovery of modules inside VSRs having an embodied, self-organizing controller.

4.1.5.7 RQ3: automatic modules discovery

So far, we assumed that the partitioning in modules of a given morphology was known *a priori*, i.e., before the optimization of a controller for that morphology. We were hence considering a *human-in-the-loop* scenario, in which the human designer plays a key role in defining modules.

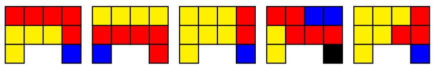
In this section, we propose a way for discovering modules *a posteriori*, i.e., after the optimization, and automatically: we hence consider a *human-out-of-the-loop* scenario. More specifically, we propose a method to partition a VSR with a given morphology into modules, inclusive of both body and brain, after the optimization of an embodied, self-organizing controller for that morphology.

While the human-in-the-loop setting is interesting per se, e.g., for transferring robotic components and assembling new robots, the human-out-the-loop would be interesting even in the long-term, visionary setting of an autonomous robotic ecosystem; in this scenario, the ecosystem could be made more efficient by reusing (recycling) components of robots that have to be disposed of. As a principled instantiation of this scenario, consider the ARE project (Hale et al., 2019b).

The proposed method works as follows. Let r be the VSR to be partitioned in modules and let B be the morphology (i.e., the set of voxels with their positions in the grid) of r and $G = (V, E)$ the controller of r. Let (B', G') be a *candidate module*, with $B' \subseteq B$ being a connected subset of voxels of r and $G' = (V', E')$ the subgraph of G containing all the vertexes $V' \subseteq V$ located in voxels of B' and all the edges $E' \subseteq E$ whose source or target nodes are in V'. We define the *module compactness* $c(B', G')$ of a candidate module (B', G') as:

$$c(B', G') = \frac{\sum e \in E'_{\text{inside}} |e.\text{weight}| + |e.\text{bias}|}{\sum e \in E' |e.\text{weight}| + |e.\text{bias}|}, \tag{4.5}$$

where $E'_{\text{inside}} = \{e \in E' : e.\text{source} \in V' \land e.\text{target} \in V'\}$ is the subset of E' edges that do not cross the boundaries of the module, i.e., whose source or target nodes are in V'.


(a) Biped

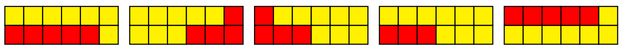

(b) Worm

Figure 4.11: A random subset of robots partitioned into modules with the automatic procedure, one color for each module. The module with the greatest compactness is depicted in red.

Based on the module compactness definition, we propose this procedure for partitioning a VSR into modules. First, we identify all the candidate modules whose size $|B'|$ is in a given range $[b_{\min}, b_{\max}]$. Second, we find the candidate module with the greatest compactness and partition the robot in that module and the modules remaining after removing it. The number of modules resulting from this procedure can be larger than 2, if, upon the removal of the module with the greatest compactness, the remaining part of the robot is not physically connected—see, for example, Figure 4.11a.

The rationale of this procedure is to exploit the findings of the previous experiments, that suggest that cutting edges is deleterious concerning module transferability. We remark that the procedure itself is applicable because the controller is a graph distributed over the body.

Other graph clustering techniques exist in the literature that we could have used in place of module compactness. To name a few, these include minimum cut algorithms (Goldschmidt and Hochbaum, 1988), spectral clustering (Seary and Richards, 1996), and community detection algorithms (Girvan and Newman, 2001). While any of the above would be a legitimate choice, we here adopt a simple approach that is still intuitive, easy to implement, and effective.

We performed a qualitative evaluation of this procedure by applying it to the VSRs evolved in the experiments described in Section 4.1.5.1. After preliminary experiments and taking into account the size of the biped and the worm, we set $b_{\min} = 2$ and $b_{\max} = 5$. Figure 4.11 shows a few of the partitioning that we obtained in this experiment.

By looking at the modules with the greatest compactness (i.e., the red ones), we notice some patterns. For the biped shape, there seems to be a tendency to find modules that span horizontally across the body, connecting the front with the rear. It might be the case that dense neural connectivity subsists between these two body parts. This is likely to be useful in a gait that is alternated like the biped one. To a lesser extent, the same considerations can be made for the worm shape.

We envision such modules to be of high utility when assembling brand-new VSRs from pre-optimized components. As a result, this heuristic could well fit in an automatic pipeline of robot disassembly and assembly, where new VSRs are fabricated in a human-out-of-the-loop manner by assembling modules picked from a repository and briefly fine-tuning them.

4.1.6 Concluding remarks

VSRs are intrinsically modular in morphology. Existing methods for building controllers for VSRs are not, however, capable of exploiting the modularity of the morphology. Addressing this limitation would permit to disassembling of robots in modules, encompassing both the body and the brain, and to reassembling them differently, to cope, e.g., with malfunctions, broken components, or different environments and tasks. In the long term, enhancing the modularity of VSRs would enable the building of libraries of pre-optimized modules that can be reused over and over.

We proposed a representation for an embodied, self-organizing neural controller in which nodes and edges of the ANN are located at precise voxels in the VSR morphology. We also described an EA suitable for evolving a controller with our representation given a task and a morphology. With an extensive experimental campaign, we showed that:

(i) our representation and EA allow to obtain effective controllers for the task of locomotion;

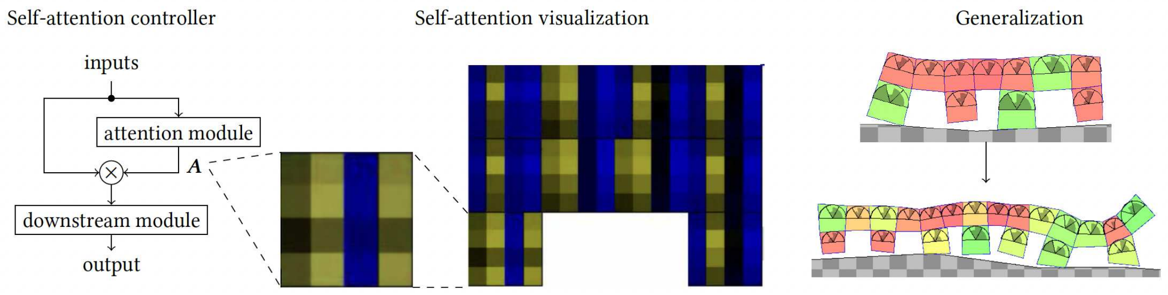


Figure 4.12: Overview of the proposed approach. We use the same neural controller (left picture) inside each voxel, with shared parameters. The middle picture is a biped with the attention matrices of the different voxels. Each controller uses self-attention to compute importance scores ($\boldsymbol{A}$) among the inputs sensed by its voxel. We also find evolved controllers to generalize to unseen morphologies (right picture; color represents the ratio between the voxel current area and its rest area: red stands for contraction, green for expansion, yellow for no change).

(ii) VSRs with evolved controllers can be disassembled and reassembled in different VSRs and are able, after a cheap re-optimization, to recover the original functionality;

(iii) modularity can be favored or discouraged through a simple numerical parameter in the mutation operators of the EA and this is reflected in the ability to recover the functionality;

(iv) due to its self-organizing property, our controller can be helpful for partitioning automatically VSRs in modules that exhibit potentially good transferability.

4.2 Evolving modular soft robots without explicit inter-module communication using local self-attention

In this chapter, we answer to question **2**.

4.2.1 Introduction

Yet, in the path toward autonomous robotic ecosystems, full exploitation of modularity is still a roadblock (Yim et al., 2007), as the flexibility of modular robots depends on how the modules are "wired" together. Minimizing the need for communication among modules would permit to disassembling robots and reassembling them differently, to cope with different tasks. Early controllers for VSRs relied on a single centralized neural network (Talamini et al., 2019) or fixed, rigid inter-module communication between voxels (Medvet et al., 2020a; Horibe et al., 2021), effectively making their design closer to a single, large neural controller with shared weights. This rigid Message-Passing (MP) mechanism, while practical, makes the modules not perfectly interchangeable, thus limiting their flexibility. This leads to the question of whether VSRs will work *at all* if we remove *all* MP channels between modules. Any communication will thus take place through physical interactions in the environment initiated by each module (Sharma et al., 2020), while simultaneously performing a task collectively: an outmost instance of morphological computation (Hauser et al., 2011) (i.e., the "brain" offloading computation to the "body").

In this chapter, we explore the properties of such modular robots without MP channels. We use the same neural controller inside each voxel, but without *any* inter-voxel communication, hence realizing the ideal conditions for modularity: modules are all equal and interchangeable, enabling our robots to truly be driven by the CI of their modules. We leverage EAs to optimize the controller parameters, and experimentally test whether the resulting VSRs are effective in a locomotion task on hilly terrains. We initially found the lack of inter-module communication to be too severe of a handicap for VSRs that rely on traditional fully-connected neural network controllers. However, we find that a local self-attention mechanism—a form of adaptive weights—achieves superior performance since instances of the same controller embodied in the same robot can focus on different inputs. See Figure 4.12 for an overview of the proposed approach. Further, we also find that we can generalize to unseen morphologies after a short fine-tuning, suggesting that an inductive bias related to the task arises from true modularity. Through these findings, we envision this work to position itself as a stepping stone in the road toward autonomous robotic ecosystems.

4.2.2 Related work

Modularity allows robotic systems to present various kinematic configurations beyond what a fixed architecture can, and such robots are usually optimal for solving many robotic tasks (Siciliano et al., 2008; Eiben, 2021b). Modularity in robotic systems takes the form of fabricating physical parts that are interchangeable for a single robot, or designing independent robots that participate in a common task adaptively (Faiña, 2021). Due to the flexibility, versatility, and robustness to changing environmental conditions (Yim et al., 2007), we are witnessing an increasing number of works on modular robots (Howison et al., 2020), including soft ones (Sui et al., 2020). For example, Kamimura et al. (2003) proposed a method to automatically generate locomotion patterns for an arbitrary configuration, and Groß et al. (2006) tackled object manipulation and transportation tasks using self-reconfigurable swarm-bots. The possibilities of such systems are encouraging, but the self-organizing and adaptive properties are even more inspiring. In a life-long gait learning task, Christensen et al. (2013) showed that, given a body configuration, a modular system can automatically figure out the best gaits, while presenting morphology independence and fault tolerance. Taking a step further, Pathak et al. (2019) demonstrated that a modular robotic system can generalize to unseen morphologies and tasks. Finally, Pigozzi and Medvet (2022) partitioned modular robots into independent units based on self-organizing neural controllers.

The concept of modularity appears also in the AL and machine learning communities, most noticeably in the area of neuroevolution and CA (Ha and Tang, 2021). Indeed, we are witnessing a surge in works that incorporate these ideas into modular robots. While most robotic works are controller-focused, AL and ML researchers also explored the co-optimization of configuration and controller (Ha, 2019; Lan et al., 2021; Bhatia et al., 2021). For instance, Cheney et al. (2014b) evolved soft robots with multiple materials through a generative encoding. Inspired by multi-cellular systems, Joachimczak et al. (2016) evolved soft-bodied animats in both aquatic and terrestrial environments, showcasing the concept of metamorphosis in simulation. In another soft robot simulation work, partially damaged robots regenerated their original morphology through local cell interactions in a neural CA system (Horibe et al., 2021). Moreover, Sudhakaran et al. (2021) grew complex functional entities in Minecraft through a neural CA-based morphogenetic process. Finally, there exists a whole body of literature on the evolution of virtual creatures with artificial gene regulatory networks (Cussat-Blanc et al., 2019).

Intuitively, the performance of a modularized system depends on the communication pattern, one therefore naturally wonders if there exist better MP graphs. Recent work suggests that GNNs possess self-organizing properties and are capable of learning rules for established CA systems (Grattarola et al., 2021), which hints at learning novel inter-module communication patterns. On the other hand, the very existence of inter-module MP prevents modular robots from being robust and truly interchangeable. Researchers thus set their eyes on the other end of the spectrum and explored the possibility of creating modular robots with more localized communication (Owaki et al., 2021) or even without inter-module communications (Martius et al., 2013; Kalat et al., 2018; Queralta et al., 2019). In this work, we evolve simulated VSRs for locomotion tasks and demonstrate that it is possible to evolve a shared controller for each voxel that excludes communication with others.

4.2.3 Methods

We provide an overview of the methods for this work.

4.2.3.1 VSR controller

We consider the distributed controller presented in Section 2.2.2.1. In particular, we adopt the "homogeneous" variant discussed in Chapter 3, where the ANNs share the same parameters: we have seen that such homogeneous representation is comparable to one where parameters are different for every ANN, with the additional benefit of a more compact search space, similarly to what happens in most multi-agent reinforcement learning systems (Wong et al., 2021). Moreover, parameter sharing makes the controller agnostic to the morphology, putting us in a vantage point to test generalization to unseen morphologies. Concerning the model of Section 2.2.2.1, we here drop MP among voxels, as the focus of this work is on minimizing (and, possibly, dispensing with) inter-module communication in soft robots: we hence enable interchangeability of modules and thus the full exploitation of modularity. Moreover, we perform actuation every k_{act} steps, rather than at every time step as in Chapter 3 and section 4.1, to prevent VSRs from exploiting the emergence of high-frequency dynamics. Finally, we equip voxels with touch, area, and velocity sensors.

Every ANN takes as input the local sensor readings $\boldsymbol{s} \in \mathbb{R}^4$ and outputs the local actuation value $a \in \mathbb{R}$. We use a one-hot encoding of $\boldsymbol{s} \in \mathbb{R}^4$ to let the voxels know where they are in the body: the actual input of the ANN is hence built as follows. Let n be the number of voxels of a given VSR; for the i-th voxel, let $\boldsymbol{h}_i \in \mathbb{R}^n$[4] be a one-hot vector that is 1 at the i-th entry and 0 everywhere else. We encode $\boldsymbol{s}$ for each i-th voxel as $\boldsymbol{X} = \boldsymbol{s}\boldsymbol{h}_i^T \in \mathbb{R}^{4 \times n}$. $\boldsymbol{X}$ is then a matrix that is equal to $\boldsymbol{s}$ at the i-th column and 0 otherwise. In this way, a voxel can distinguish itself in the morphology. We apply this same pre-processing to every model considered in this chapter.

We remark that one-hot encoding does not invalidate the claim that voxels are all identical: in practice, it just requires the "operator" (i.e., the robot assembler) to set the position of each voxel "in" the voxel controller itself, i.e., to do the proper configuration. Regardless of the ANN architecture, there is a unique vector $\boldsymbol{\theta} \in \mathbb{R}^p$ of parameters that specifies every ANN in the VSR. Thus, we optimize a VSR for a given task by optimizing the parameters $\boldsymbol{\theta}$.

[4] h was the activation in Section 4.1; with a slight abuse of notation, we repurpose it to be a one-hot encoding.

4.2.3.2 Self-attention

Attention can be seen as an "adaptive weights" (Ferigo et al., 2022a) mechanism that computes importance scores for the inputs. Attention mechanisms were first introduced in the context of machine translation (Bahdanau et al., 2014; Luong et al., 2015) to capture relationships in temporal sequences of data, and have thus prospered in natural language processing (Devlin et al., 2018; Galassi et al., 2020). Attention mechanisms have achieved state-of-the-art performance in domains (e.g., computer vision (Khan et al., 2021)) where data are not temporal but spatial (Dosovitskiy et al., 2020; Wu et al., 2020) or even sets (Lee et al., 2019; Tang and Ha, 2021), also considering robotic settings (Choi et al., 2017; Zambaldi et al., 2018; Tang et al., 2020). There is indeed evidence that the nervous system modulates attention on every sensory channel (Driver, 2001). As a result, being attention an adaptive weights mechanism, we argue it could supplant inter-module communication by computing importance scores that are tailored to the specific voxel, while being local (i.e., attending only to voxel-specific information) and shared (i.e., same parameters for every voxel).

Let $\boldsymbol{X} \in \mathbb{R}^{r \times u}$ be a sequence of u inputs of dimension r[5]. In its general formulation, an attention module computes an *attention matrix* $\boldsymbol{A} \in \mathbb{R}^{r \times r}$, to get weighted inputs:

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} \tag{4.6}$$

to be fed to a downstream module $f(\boldsymbol{Y}) = z \in \mathbb{R}$ for some desired output z.

While a great variety of attention mechanisms do exist in the literature (Chaudhari et al., 2021), we resort to self-attention (Vaswani et al., 2017). Self-attention is a generalized form of attention and has already been shown to achieve state-of-the-art results on continuous control tasks that exploit data other than temporal (Tang et al.,

[5]r denoted a robot in Section 4.1; with a slight abuse of notation, we repurpose it to be a dimension.

2020). Self-attention computes:

$$\boldsymbol{A} = \sigma\left(\frac{1}{\sqrt{d}}\boldsymbol{Q}\boldsymbol{K}^T\right) \tag{4.7}$$

where $d \in \mathbb{R}$, $\sigma(\cdot)$ is a non-linear function that constrains $\boldsymbol{A}$ to be in a given range (e.g., tanh), $\boldsymbol{Q}, \boldsymbol{K} \in \mathbb{R}^{r \times d}$ (known as the Query and Key matrices) are the output of linear transformations of the form:

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_q + \boldsymbol{b}_q \tag{4.8}$$

$$\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_k + \boldsymbol{b}_k \tag{4.9}$$

where $\boldsymbol{W}_q \in \mathbb{R}^{u \times d}$, $\boldsymbol{W}_k \in \mathbb{R}^{u \times d}$ are weight matrices, $\boldsymbol{b}_q \in \mathbb{R}^d$, $\boldsymbol{b}_k \in \mathbb{R}^d$ are bias vectors, and $+$ denotes the matrix-vector addition (the vector is added to each row of the matrix). We take the dot product between $\boldsymbol{Q}$ and $\boldsymbol{K}$ to compute compatibility between two different representations of the inputs. The division in Equation (4.7) appears because the dot product grows with the operand dimensions.

Self-attention in VSR controller We use the one-hot encoded sensor reading as input $\boldsymbol{X}$: then, $r = 4$ (number of sensor readings) and $u = n$ (number of inputs, the voxels). The attention matrix is $\boldsymbol{A} \in \mathbb{R}^{4 \times 4}$: the attention is on the sensor readings and $\boldsymbol{A}$ dimension does not depend on the robot morphology. For self-attention to focus only on local information, we set on each i-th voxel $\boldsymbol{W}_q = \boldsymbol{h}_i\boldsymbol{w}_q^T$ and $\boldsymbol{W}_k = \boldsymbol{h}_i\boldsymbol{w}_k^T$, with $\boldsymbol{h}_i$ defined as in Section 4.2.3.1: in this way, we extract the column of $\boldsymbol{X}$ corresponding to the voxel. $\boldsymbol{w}_q, \boldsymbol{w}_k \in \mathbb{R}^d$ are evolvable vectors of parameters and are the same for all the voxels. We summarize the building blocks of our architecture in Figure 4.13.

After preliminary experiments, we set $\sigma(\cdot)$ to be tanh, $d = 8$, and f to be a Multi-Layer Perceptron (MLP) with no hidden layers and tanh activation function (to ensure the output lies in $[-1, +1]$). Moreover, given that we use tanh as non-linearity in Equation (4.7), the entries of $\boldsymbol{A}$ lie in $[-1, +1]$, $+1$ being the highest

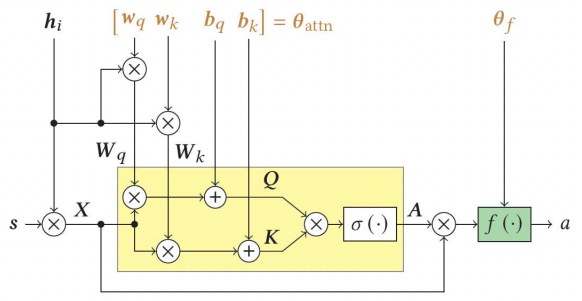


Figure 4.13: The architecture of our self-attention controller. Evolvable parameters are shown in brown, the attention module is shown in yellow, and the downstream module is shown in green. Parameters are the same for every voxel.

compatibility between two inputs and -1 the least. Finally, our model differs from the original formulation of self-attention (Vaswani et al., 2017) in that we set Values to be the identity function, since, after preliminary experiments, we found them to be unnecessary.

In a self-attention model of this form, we optimize the parameters $\boldsymbol{\theta} = [\boldsymbol{\theta}_{\text{attn}} \; \boldsymbol{\theta}_f]$, where the attention module parameters $\boldsymbol{\theta}_{\text{attn}}$ are the concatenation of $\boldsymbol{w}_q, \boldsymbol{w}_k, \boldsymbol{b}_q, \boldsymbol{b}_k$, and the downstream module parameters $\boldsymbol{\theta}_f$ are the weights and biases of the downstream MLP. Then, $\boldsymbol{\theta}$ is the genotype of our EA, which we detail in the next subsection.

4.2.3.3 Evolutionary algorithm

We perform optimization through EC; in particular, we resort to a GA (De Jong, 2006). Indeed, Risi and Stanley (2019) used GAs to effectively evolve complex neural architectures consisting of heterogeneous modules, and Such et al. (2017b) proved GAs to be competitive with state-of-the-art reinforcement learning algorithms.

We adopt the same GA of Section 3.3.5.2. After preliminary experiments and exploiting our previous knowledge, we set $n_{\text{pop}} = 100$, $n_{\text{tour}} = 5$, $\sigma_{\text{mut}} = 0.35$, $\sigma'_{\text{mut}} = 0.1$, $p_{\text{mut}} = 0.2$, and, unless otherwise specified, $n_{\text{evals}} = 30\,000$.

4.2.4 Experiments

Our goal is to answer the following questions with an experimental analysis:

RQ1 Are VSRs evolved with self-attention effective at solving a locomotion task? If so, are they robust to environmental changes?

RQ2 Why does self-attention work?

We evaluate our method on two different VSR shapes, namely a 4×3 rectangle with a 2×1 rectangle of missing voxels at the bottom-center, that we call *biped* (same as Figure 2.2a), and a 7×2 rectangle with empty voxels at the odd x positions in the bottom row, that we call *comb*. The dimension of $\boldsymbol{X}$ is hence 4×10 for the biped shape and 4×11 for the comb shape (the second operand in the multiplication is the number of voxels); as a result, the self-attention controller has $|\boldsymbol{\theta}| = 73$ parameters for biped ($|\boldsymbol{\theta}_f| = 41$ of the downstream MLP and $|\boldsymbol{\theta}_{\text{attn}}| = 32$ for the attention module) and 77 for comb ($|\boldsymbol{\theta}_f| = 45$ and $|\boldsymbol{\theta}_{\text{attn}}| = 32$).

For all the experiments, we considered the task of locomotion. The goal is to travel as fast as possible on a terrain along the positive x direction, in a fixed amount of simulated time t_{final}. We use as fitness the average velocity $\overline{v}_x$ of the center of mass of the VSR during the simulation. We set $t_{\text{final}} = 30\,\text{s}$. As in Section 4.1, we consider a hilly terrain, consisting of bumps that are randomly procedurally generated with an average height of 1 m and an average distance of 10 m. We randomize the seed for the procedural generation at every fitness evaluation and re-evaluate individuals retained from the previous iteration, so that evolution does not unfairly favor individuals with an "easy" terrain and to make adaptation more challenging.

We implemented the experimental setup in the Java programming language, relying on JGEA[6] for the evolutionary optimization and 2D-VSR-Sim (Medvet et al.,

[6]`https://github.com/ericmedvet/jgea`.

2020b) for the simulation of VSRs. For the latter, we set $\Delta t = \frac{1}{60}$ s for the time step, and all other parameters to default values (as a result, all voxels share the same mechanical properties). After preliminary experiments, we set $k_{\text{act}} = 20$ (i.e., one actuation every ≈ 0.33 s). We made the code publicly available at `https://github.com/pigozzif/AttentionVSRs`.

For each experiment, we performed 5 evolutionary runs by varying the random seed for the EA. We remark that, for a given VSR and terrain, the simulations are instead deterministic. We carried out all statistical tests with the Mann-Whitney U rank test for independent samples.

4.2.4.1 Results

We present the results of our work.

RQ1: effectiveness and robustness with self-attention To verify the effectiveness of evolved VSRs equipped with our self-attention model, we measure their performance in two different cases: in the same conditions, they were evaluated during the evolution and in slightly different environmental conditions aimed at testing individual generalization abilities. In both cases, we use $\overline{v}_x$ as the performance index: in the former, it is the value of the fitness function itself, while in the latter it is the average over 10 unseen hilly terrains, obtained with 10 different predefined random seeds.

As a baseline, we compare the self-attention model (hereon Attention) with:

(a) a "communication-less" MLP (hereon MLP), that takes the same input as Attention;

(b) a "communication-based" MLP (hereon MLP-Comm), that takes as input the local sensor readings and the 4 values generated by the four adjacent voxels (if any, or zeros, otherwise) at the previous time step. Then, it outputs the local

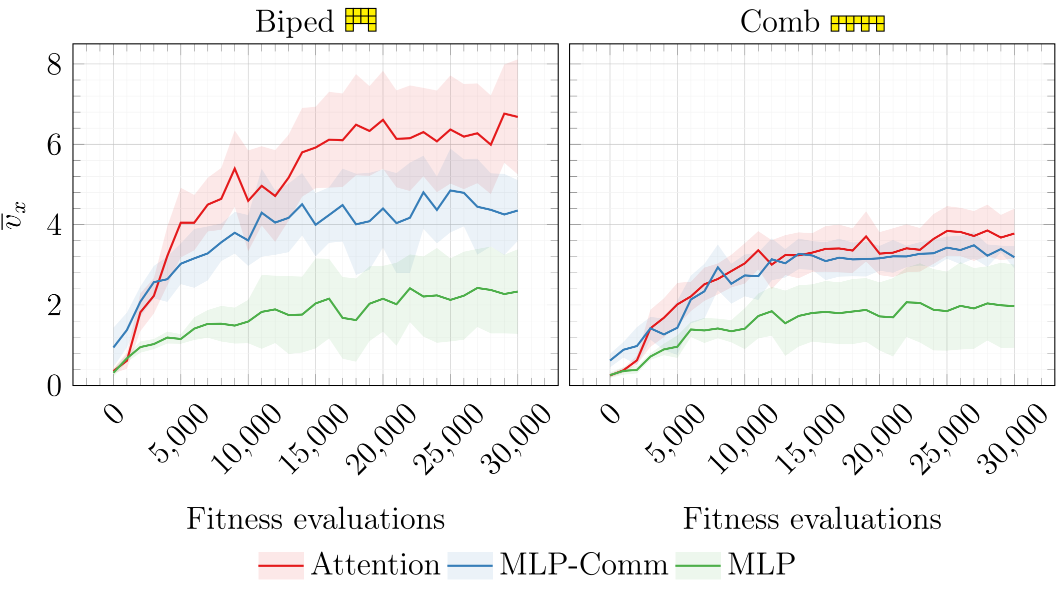


Figure 4.14: Median ± standard deviation (solid line and shaded area) of the average velocity for the best individuals found during each evolutionary run, obtained with three controller models and two shapes. Attention is never worse than the baselines.

actuation and the 4 values that will be used by the adjacent voxels at the next time step. This model is the same as (Medvet et al., 2020a, 2021) and, by the MP mechanism, is an instance of a communication-based controller.

Both models have the same architecture as the self-attention downstream MLP (see Section 4.2.3.2): as a result, MLP has 41 parameters for biped and 45 for comb, while MLP-Comm has 405 for biped and 440 for comb. MLP-Comm and MLP differ from Attention in that they do not employ an attention mechanism to obtain importance scores for the inputs. For the optimization, we use the same EA of Section 4.2.3.3.

We summarize the results in Figure 4.14, which plots $\overline{v}_x$ in terms of median ± standard deviation for the best individuals throughout evolution. Moreover, Figure 4.15 shows $\overline{v}_x$ for the best individuals over the re-assessment terrains. For the same shape, it also reports the p-value for the statistical test against the null hypothesis of equality between the medians.

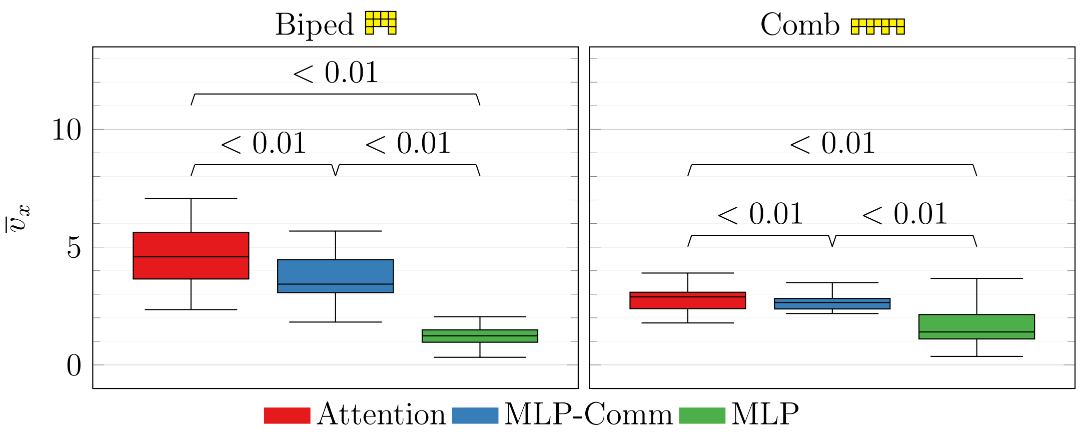
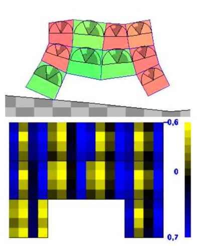
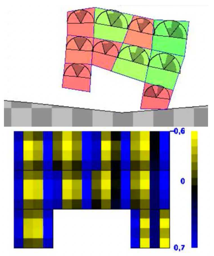


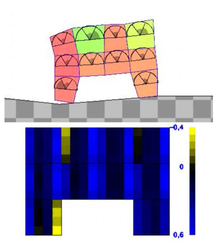
Figure 4.15: Distribution of the average velocity across 10 unseen hilly terrains for the best individuals found during each evolutionary run, obtained with two controller types and two shapes. Attention outperforms the baselines in terms of re-assessment.

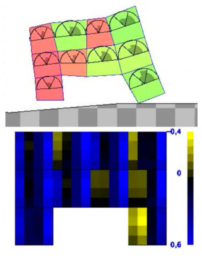
From the figures, we find that Attention outperforms both baselines. The curves of Figure 4.14 also suggest that all models settle on a plateau and that continuing evolution would unlikely bring better results. Attention individuals also perform better in terms of re-assessment: they achieve significantly better $\overline{v}_x$ on unseen terrains. Moreover, MLP-Comm outdoes a poor-performing MLP, as attested by the low $\overline{v}_x$ scores, especially for the biped shape. We visually inspected the evolved behaviors for the best individuals and found them to be highly adapted to a locomotion task on hilly terrain; indeed, bipeds hop on their legs as equines do and combs propagate leg movements from posterior to anterior as millipedes do. We made videos available at `https://softrobots.github.io`[7]. With no communication, a "vanilla" MLP controller fails; it does not enable communication-less VSRs. To reach a decent performance, we must add an MP communication mechanism. Intuitively, the MP mechanism is very important for emerging behavior to arise with an MLP-based controller. Self-attention instead does enable truly communication-less VSRs: it performs better or comparatively to both a communication-less and a communication-based MLP. We believe the reason for this to be that self-attention is a form of adaptive weights, tailoring importance scores to the inputs and, indirectly, to the particular

[7]Number 1 and number 2.


(a)


(b)


(c)


(d)

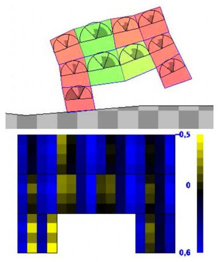

(e)

Figure 4.16: Time-lapse showing locomotion for a sample VSR and the corresponding attention matrices. The color of each voxel encodes the ratio between its current area and its rest area: red for < 1, yellow ≈ 1, green > 1; the circular sectors drawn at the center of each voxel indicate the current sensor readings. The color of each attention score tells the strength of the compatibility between inputs and provides an interpretation of what is driving the behavior of the VSR.
voxel.

In addition to the evolved behaviors, we also visualized the attention matrices. Indeed, interpretability is one of the major strengths of self-attention and allows humans to get an insight into the robot's inner decision mechanisms. To illustrate that point, Figure 4.16 presents a locomotion time-lapse for a sample VSR: at each time step, it displays the robot state in the top row and the corresponding attention matrices in the bottom row; as happens with our distributed controller, there is one such attention matrix for every voxel in the morphology. Columns and rows of the attention matrices correspond to sensors, in particular: first for the touch sensor, second and third for the x- and y-velocity sensors, and fourth for the area sensor. The color of each attention score tells the strength of the compatibility between inputs and provides an interpretation of what is driving the behavior of the VSR.

From the visualization, we realize attention scores are consistent with humans' intuition and common sense. From Figure 4.16a to Figure 4.16b, attention shifts from the touch sensor of the rear leg to the touch sensor of the front leg. In Figure 4.16c, the VSR stops at a hole in the terrain, and the attention matrix witnesses a radical shift as a consequence. In Figure 4.16d and Figure 4.16e, the VSR starts walking again by first focusing on the front leg, and then on the rear leg, initiating locomotion once again. We found the other individuals to present similar attention patterns.

We conducted an ablation study to assess whether evolution found a trivial solution for self-attention or not. Indeed, AL researchers are all aware of the many uncanny and "creative" convergences that artificial evolution is capable of (Lehman et al., 2020). In particular, we test whether the attention update over time is needed or not, i.e., if there is a "one attention to rule them all" case. To this end, we conducted the following procedure. Given an evolved VSR with Attention, we take a snapshot of it at every second of the simulation, alongside its attention matrix at that time step. For every such snapshot, we simulate it on a fixed unseen hilly terrain for 30 s with the attention matrix frozen (i.e., it is the same as the one in the snapshot). We repeated this procedure for the best individual of every evolutionary run. We summarize the results in Figure 4.17 in terms of $\overline{v}_x$ (median $\pm$ standard deviation), with the time (in s) at which we took the corresponding snapshot on the x-axis. We see that performance drops dramatically, meaning that self-attention is indeed a fundamental piece in the architecture and that it is non-trivial.

Through that evidence, we can answer positively to RQ1: VSRs evolved with self-attention can solve the task of locomotion and outperform an MLP baseline without relying on inter-module communication, while being fairly able to generalize to unseen terrains.

RQ2: why does self-attention work? We hypothesize the reasons for self-attention effectiveness to be:

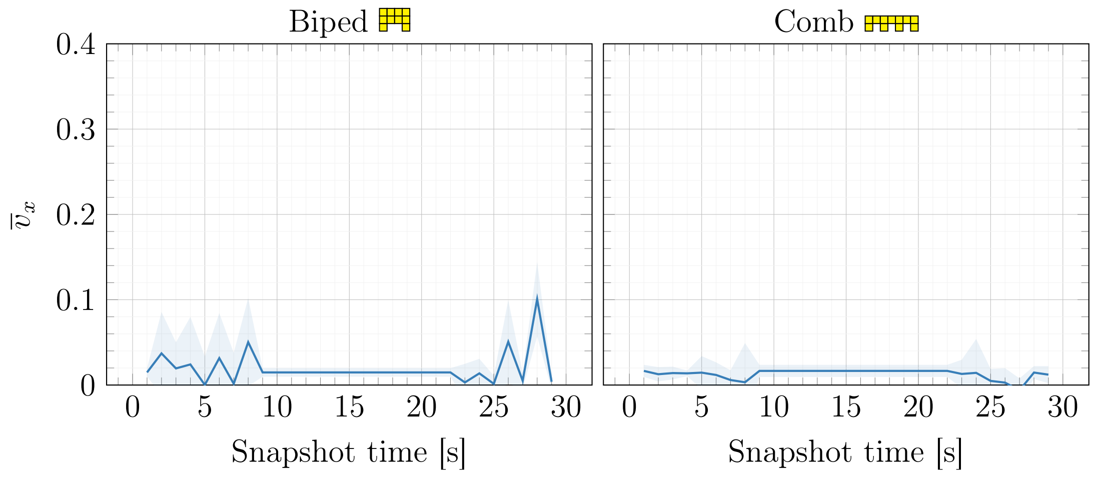


Figure 4.17: Median ± standard deviation (solid line and shaded area) of the average velocity for the best individuals found during each evolutionary run, with the attention matrix frozen at different time steps of the simulation (on x axis in s), obtained with the two shapes. Self-attention evolves to be a necessary component of the controller, and ablating it results in abysmal performance.

(a) self-attention evolves to represent an *inductive bias*, i.e., what tasks the controller is naturally suited at tackling, and

(b) self-attention evolves to be an *instinctive* component (akin to reflexes in biology), that complements decision-making with a fast time-scale of adaptation.

We first focus our attention on the inductive bias hypothesis. In the field of machine learning, an "inductive bias" refers to the set of assumptions made by an algorithm to generalize to novel and unseen data (Mitchell, 1980); in our case, data refers to tasks. If self-attention evolved an inductive bias, as others did point out (Zambaldi et al., 2018), it would represent features that are general for the task of locomotion and are thus suitable for generalization.

To test the inductive bias hypothesis, we experiment with generalization to unseen morphologies. Given the attention module $\boldsymbol{\theta}_{\text{attn}}$ evolved on a specific morphology, we wonder whether we can freeze it and use it as an off-the-shelf "feature extractor" for a different morphology—as mentioned in Section 4.2.3.2, $\boldsymbol{\theta}_{\text{attn}}$ dimension is agnostic concerning the morphology. If self-attention evolved an inductive bias, it would be useful to control the new morphology. In particular, for frozen self-attention, we fine-tune the downstream module. The goal of fine-tuning is to have a VSR that is effective and converges faster than evolving from scratch. If that were the case, self-attention would be useful to quickly assemble new VSRs for a different task, starting from pre-optimized components. This fine-tuning procedure is of crucial importance since we expect one-shot generalization to fail as a consequence of the embodied cognition paradigm (Pfeifer and Bongard, 2006; Shapiro, 2019), which posits a deep entanglement between the morphology and the controller of an embodied agent.

For every best individual $\boldsymbol{\theta}^\star = \left[\boldsymbol{\theta}^\star_{\text{attn}}\, \boldsymbol{\theta}^\star_f\right]$ of an evolutionary run from the experiments of Section 4.2.4.1, we freeze its attention module parameters $\boldsymbol{\theta}^\star_{\text{attn}}$ and fine-tune its downstream module parameters $\boldsymbol{\theta}^\star_f$ using the same GA of Section 4.2.3.3 (i.e., mutation and crossover operate on $\boldsymbol{\theta}^\star_f$ only), yet starting from a different initial population. In detail, the initial population is composed of $\boldsymbol{\theta}^\star$ and $n_{\text{pop}} - 1$ mutations of it, obtained by copying $\boldsymbol{\theta}^\star_{\text{attn}}$ and re-initializing $\boldsymbol{\theta}_f$ by sampling uniformly $[-1, +1]^{|\boldsymbol{\theta}_f|}$.

We conducted an experimental campaign of 5 evolutionary runs, lasting $n_{\text{evals}} = 20\,000$ fitness evaluations each, with one larger version of each of the two shapes used in the previous experiments. We fine-tuned the best individual of every run as explained above, freezing and transferring its attention module to the corresponding smaller or larger morphology, i.e., from biped-small to biped-large, from biped-large to biped-small, from comb-small to comb-large, and from comb-large to comb-small. For fine-tuning, we set $n_{\text{evals}} = 10\,000$ fitness evaluations as termination criterion, as our goal is to quickly adapt to a new task starting from pre-optimized components (i.e., the attention module). We summarize the results in Figure 4.18: for each morphology, we compare $\overline{v}_x$ (in terms of median $\pm$ standard deviation) for fine-tuning (Fine-tune) and from-scratch optimization (From-scratch). For a fair comparison, we plot the results from From-scratch over 30 000 fitness evaluations, since every Fine-tune run is the outcome of evolution with 20 000 fitness evaluations plus fine-tuning with 10 000.

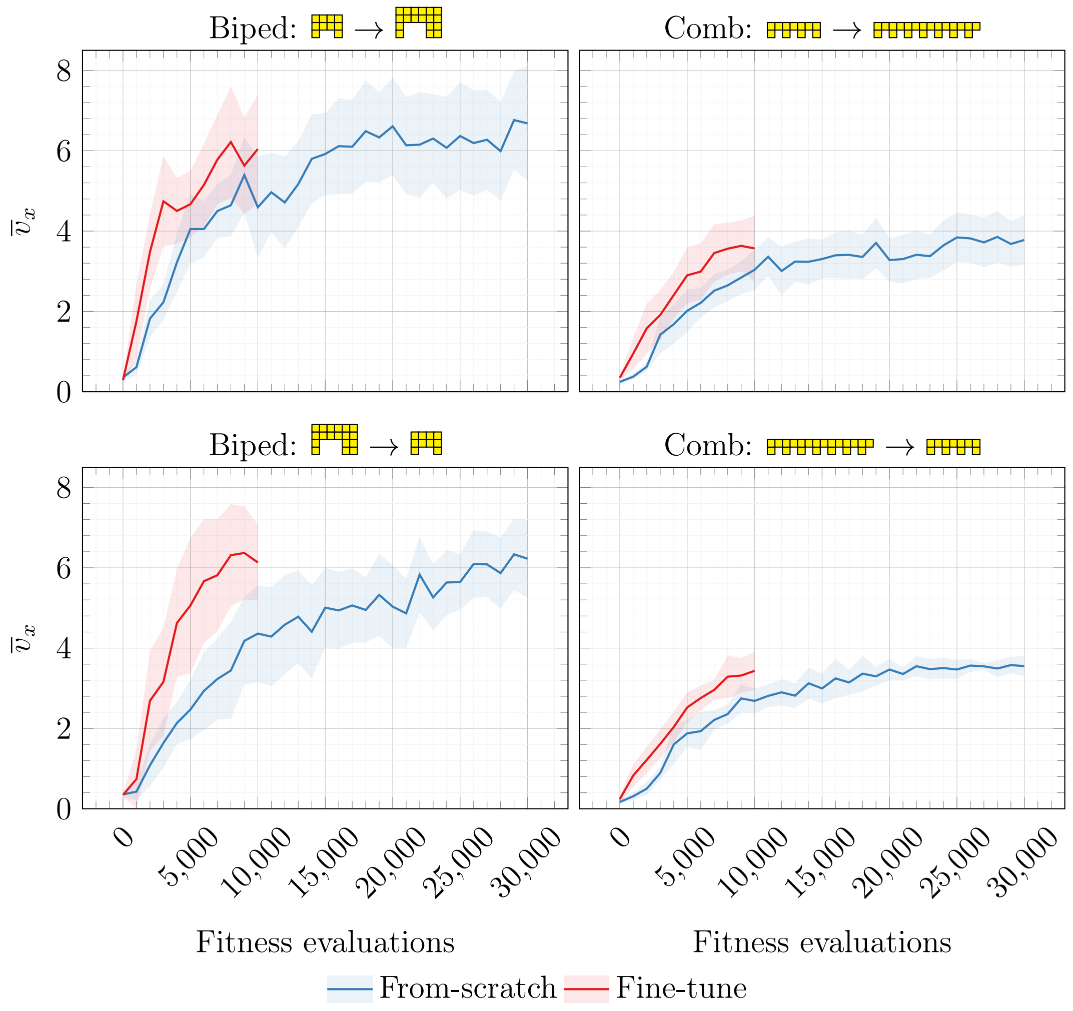


Figure 4.18: Median ± standard deviation (line and shaded area) of the average velocity for the best individuals found during each evolutionary run, with two controller types (color), with optimization from scratch (dashed line) or fine-tuning pre-optimized attention (solid line) on a different morphology. Attention evolves an inductive bias that, generally, represents invariant features for the locomotion task.

From Figure 4.18, we see that Fine-tune succeeds in converging much faster. Moreover, the curves suggest that Fine-tune would benefit from a longer re-optimization; however, we remark that the goal of this study is not to reach the best performance, but to converge cheaply. We visually inspected the behaviors of the Fine-tune individuals and found them to be adapted for a locomotion task and consistent with the behaviors observed in Section 4.2.4.1. We made videos available at `https://softrobots.github.io` for the best Fine-tune individuals[8].

We conclude that it is often possible to re-use an evolved attention module for a smaller or larger morphology, after a fine-tuning stage. That result is hopeful in the context of future autonomous robotic ecosystems: we expect new robots to be assembled from pre-optimized components, so that they are, at the same time, proficient at new tasks and computationally cheaper than optimizing from scratch.

We believe the result is relevant, as, to date, (Kriegman et al., 2021) is the only other work addressing generalization to different soft robot morphologies through EC. Albeit ground-breaking, Kriegman et al. (2021) achieved generalization by computing the fitness function over three different morphologies at every evaluation, which might not be feasible in a fully autonomous setting. We achieve generalization with no penalty during evolution, at the cost of introducing a fine-tuning stage. Finally, we remark it could be argued that other works, e.g., (Wang et al., 2018; Pathak et al., 2019; Huang et al., 2020), achieved generalization to unseen morphologies via techniques more sample-efficient than EC, most notably Reinforcement Learning (RL). While those advances are indeed noteworthy, there are reasons to use EC in the first place: empirically, we find that RL algorithms are notoriously more unstable than evolution; indeed, there is a recent body of literature that shows how even simple EAs can achieve performance competitive to state-of-the-art RL systems (Salimans et al., 2017; Such et al., 2017b; Risi and Stanley, 2019), at the benefit of less complexity and less sensitivity to hyperparameters.

[8]Numbers 3, 4, 5, and 6.

It is worth noting that evolution and learning can complement each other very effectively (Eiben and Hart, 2020b). For example, in evolutionary robotics, learning augments evolution by allowing newborn controllers to adapt more quickly to their bodies (Gupta et al., 2021; Luo et al., 2021). Self-attention is not a form of learning: in particular, there is no memory or state, as every attention matrix is computed just from current observations. We argue that self-attention evolves to have an instantaneous time-scale of adaptation; similarly to reflexes, it immediately reacts to sensory perceptions. Under this light, self-attention belongs to the "System 1" mode of thought: as put forward in the seminal work of Kahneman (2011), System 1 thinking is fast, instinctive, and emotional. System 1 then exists at the level of the subconscious. That hypothesis is in line with the nature of attention in biology (Cohen and Rafal, 1991; Treisman et al., 1992).

To validate that hypothesis, we experiment by slowing down the dynamics of attention. In particular, given the attention matrix $\boldsymbol{A}$ as defined in Equation (4.7), that depends only on current input, we define the matrix $\boldsymbol{A}'^{(k)}$ as:

$$\boldsymbol{A}'^{(k)} = \begin{cases} \alpha\boldsymbol{A} + \eta\boldsymbol{A}'^{(k-1)} & \text{if } k > 0 \\ \boldsymbol{A} & \text{if } k = 0 \end{cases} \tag{4.10}$$

where $\alpha, \eta \in [0, 1]$. α acts as a *learning rate*, weighting the current attention, while η acts as a *forget rate*, weighting the attention compounded from the previous time steps. In the following, we substitute $\boldsymbol{A}$ with $\boldsymbol{A}'^{(k)}$ in Equation (4.6), i.e., $\boldsymbol{A}'^{(k)}$ is the effective attention matrix at time step k, i.e., at time $t = k\Delta t$. As a side note, if we enforce $\alpha + \eta = 1$, Equation (4.10) resembles the "associative weights" memory of Ba et al. (2016).

In the following, we evolve α and η as part of the genotype. To enforce $\alpha, \eta \in [0, 1]$, when mapping a genotype $\boldsymbol{\theta}$ into a robot phenotype, we set both to be the absolute value clipped at 1 of the corresponding genes. In doing so, we let evolution discover what is the fittest value for α and η and, as a result, what is the optimal balance between current information and past information in self-attention. In a certain sense, we are "meta-evolving" self-attention.

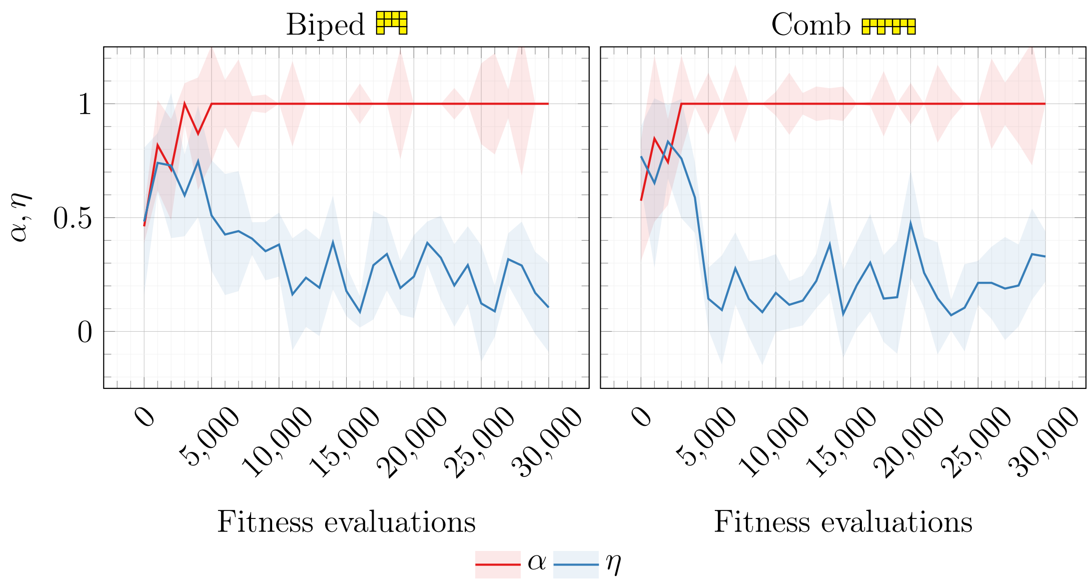


Figure 4.19: Median ± standard deviation (solid line and shaded area) of the α and η values of Equation (4.10) for the best individuals found during each evolutionary run, obtained with two shapes. Self-attention evolves to quickly forget past information and attend to the present.

With those settings, we conducted an experimental campaign of 5 evolutionary runs with the shapes of Section 4.2.4.1. We found the resulting best individuals to be not significantly different from those evolved previously, in terms of effectiveness and qualitative analysis of self-attention, so we simply show in Figure 4.19 the evolution of α and η. We notice a clear trend: α evolves to be 1, while η evolves to come closer to 0. In other words, self-attention evolves to keep all the present information and retain little from the past. Those results mostly confirm our hypothesis that evolution leads self-attention to have an instantaneous time-scale of adaptation, akin to an instinct.

Through those tests, we can answer to RQ2: self-attention works because it evolves to represent an inductive bias that is useful for generalization; at the same time, self-attention evolves to be an instinctive adaptation mechanism, allowing the controller to react quickly to changes in the inputs.

4.2.5 Conclusion

In the path toward autonomous robotic ecosystems, full exploitation of modularity remains a roadblock (Yim et al., 2007). Minimizing the need for communication among robot modules would greatly facilitate the disassembly of robots in components and their reassembly in different forms, to cope, e.g., with different tasks.

Considering the case of VSRs, that, a priori, enjoy a high degree of modularity (in both bodies and brains), those points are precisely what the chapter demonstrates:

(i) We can evolve controllers that dispense with inter-module communication for a locomotion task on hilly terrains;

(ii) We can achieve generalization to unseen morphologies, after a short fine-tuning with evolution.

To do so, we employ a local self-attention mechanism—a form of adaptive weights—to let every voxel adapt to its local perceptions and optimize the controller parameters (which are the same for all the voxels) with EC. On the other side, an MLP baseline is not proficient in the same task. We also observe that self-attention evolves to be an instinctive adaptation mechanism, allowing the controller to quickly react to input changes.

Yet, it is unclear how our results extend beyond the scope of modular soft robots, including modular rigid robots. Future work will develop on these ideas to investigate how critical the soft body dynamics are in supporting full controller modularity, and how to leverage self-attention to evolve communication patterns tailored to different voxels.

Chapter 5: Spatial and Time Scales of Collective Intelligence

No work to date considers the case of modules that, by their local information only, must reason about the global properties of the collective; in other words, the spatial scale of CI. Indeed, a vital component of intelligent action is affordance detection: understanding what actions external objects afford the agent. Affordance detection is relevant to improve the reliable and safe action of robots in the world. In Section 5.1, we find that a robot with an appropriate morphology can evolve to predict whether it will fit through an aperture with just minimal tactile feedback. We also find that some robot morphologies facilitate the evolution of more accurate affordance detection, while others do not.

The time scale of CI is also important. Indeed, that is how nature shaped animal life on Earth: animals are endowed with bodies sculpted by evolution and learn throughout a lifetime. Still, learning the controllers of robots while evolving their morphologies is a complex endeavor in robotics; one issue is the choice of genetic encoding for the morphology. Such a choice can be crucial for the effectiveness of learning, i.e., how fast and to what degree agents adapt, through learning, during their lives. In Section 5.2, we evolve the morphologies of VSRs with two different encodings, direct and indirect while learning the controllers with reinforcement learning. We experiment with three tasks, ranging from cave crawling to beam toppling, and study how the encoding influences the learning outcome. Our results show that the direct encoding corresponds to an increased ability to learn, mostly in terms of learning speed. The same is not always true for the indirect one. We link these results to different shades of the Baldwin effect, consisting of morphologies being selected for increasing an agent's ability to learn during its lifetime.

In addition, we hypothesize that learning, since it operates at a faster time scale than evolution, can help in generalizing to unseen environmental conditions.

According to Hebbian theory, synaptic plasticity is the ability of neurons to strengthen or weaken the synapses among them in response to stimuli. It plays a fundamental role in the processes of learning and memory of biological neural networks. With plasticity, biological agents can adapt on multiple time scales and outclass artificial agents, the majority of which still rely on static ANN controllers. In Section 5.3, we propose a Hebbian ANN controller for VSRs where every synapse is associated with a Hebbian rule that controls the way the weight is adapted during the VSR lifetime. For a given task and morphology, we evolve the controller for the task of locomotion the parameters of the Hebbian rules, rather than the weights. Our results show that the Hebbian controller is more adaptable to unseen environmental conditions (in the form of damage). We also provide novel insights into the inner workings of plasticity and demonstrate that "true" learning does take place, as the evolved controllers improve over the lifetime and generalize well.

5.1 Morphology choice affects the evolution of affordance detection in robots

In this chapter, we answer to question **3**.

5.1.1 Introduction and related works

A vital component of cognition is affordance detection (Gibson, 1977): understanding what actions external objects afford the viewer (i.e., the agent). These include the viewer's body, its surrounding environment, and how these can potentially interact. If we were to deploy robots in hazardous and exotic environments (Hale et al., 2019a; Nitschke and Howard, 2021), we would need to endow such robots with a sense of what their bodies and environments afford them for their safe action. Consider the case of a robot that must pass through an aperture: not every aperture is passable for every robot body, and attempting to pass through an impassable aperture would be disastrous for the robot itself and the agents interacting with it. Thus,

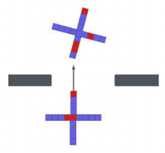
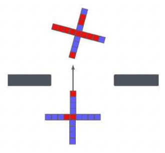
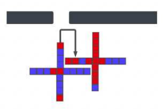
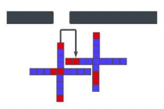
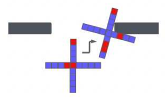
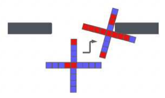
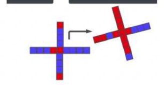
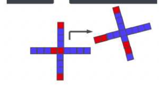
		passable		impassable	
		affordance detection		**affordance detection**	
		correct	incorrect	correct	incorrect
action	correct				
	incorrect				

Figure 5.1: Our robots (here, a starfish-shaped robot) predict their affordances (such as the passability of an aperture given their bodies) and take appropriate actions (attempt to pass through passable apertures). For the sake of the former, voxels vote (based on their local sensor readings) for impassability (red color) or passability (blue color). We evolve robots that can discriminate between passable and impassable apertures while moving in the correct direction (on the other side of the aperture if it is passable, in front of the aperture if it is impassable).

we want robots that behave according to correct predictions of affordances projected by objects in their environment.

Natural agents evolved the innate ability to detect affordances (Stoffregen, 2018). Indeed, neuroscience has argued that no cognition is otherwise possible without understanding the interactions between one's body and the environment (Limanowski and Blankenburg, 2013; Kiverstein and Sims, 2021). Not surprisingly, researchers have striven to endow artificial agents with the same ability. Different communities developed different concepts and terminologies: self-modeling in robotics (Bongard et al., 2006; Cully et al., 2015; Chen et al., 2022), world models (Ha and Schmidhuber, 2018) in machine learning, intrinsic motivation (Oudeyer et al., 2007), empowerment (Klyubin et al., 2005) and information-driven measures (Martius et al., 2013) in the cognitive sciences. More specifically, Slocum et al. (2000), extending the work of (Beer et al., 1996) on the simplest behavior that raises issues of genuine cognitive interest, evolved robots to detect affordances by visually deciding which openings their bodies could and could not fit through.

Still, whether and how morphology affects a robot's ability to detect affordances is yet an unexplored issue, despite the *embodied cognition* paradigm (Pfeifer and Bongard, 2006), positing a deep entanglement between the brain, the body, and the environment, suggests that morphology can facilitate or obstruct the ability to evolve a behavior for an animal or robot.

Here, we set out to study morphology choice in the evolution of affordance detection and consider VSRs. Given that the interactions between a soft robot's body and its environment are hard to predict (Rus and Tolley, 2015), affordance detection is likely to be more arduous for soft robots. These robots, equipped with minimal tactile feedback, decide whether or not their bodies can fit through an aperture, and, if so, then attempt to pass through it. We experiment with three morphologies and evolve the parameters of their closed-loop controllers since EAs allow for the exploration of search spaces unencumbered by a priori assumptions.

We show experimentally that the choice of morphology plays a key role: some morphologies facilitate the evolution of affordance detection, while others do not. In particular, simpler morphologies may be less suitable for detecting affordances. Looking forward, we envision that the joint optimization of morphology and control may facilitate the evolution of affordance detection in robots and thus improve their reliable and safe action in the world.

5.1.2 Embodied agent model

Soft robots have several advantages over rigid robots, such as deforming their bodies (Shah et al., 2021b) for, e.g., squeezing through tight spaces (Cheney et al., 2015) or underwater locomotion (Corucci et al., 2018). However, the interactions between a soft robot's body and its environment are more complex and harder for the robot to predict (Rus and Tolley, 2015). For example, a soft robot may be able to deform itself to fit through a narrow aperture but not be able to determine what that deformation is. For this reason, we herein investigate the evolution of soft robot affordance detection, as it is likely to be more difficult compared to rigid robots.

5.1.2.1 Mechanical model: the Voxcraft simulator

We simulate VSRs and their environment using the Voxcraft soft-body physics engine (Liu et al., 2020), the GPU-accelerated version of Voxelyze (Hiller and Lipson, 2014). Contrary to what was seen in previous chapters, this simulator is a 3D one. Voxcraft simulates each voxel as a point mass connected to up to six neighboring point masses with Euler-Bernoulli beams. Every voxel has a specific temperature that, at time step k, changes according to:

$$\Delta T = \alpha a_k \tag{5.1}$$

where $\alpha \in \mathbb{R}$ is a fixed amplitude across the robot and $a_k \in \mathbb{R}$ is the temperature change (actuation) output by the controller embedded in the voxel. A beam, in turn, sets its resting length to the average of the current temperatures of the two voxels it

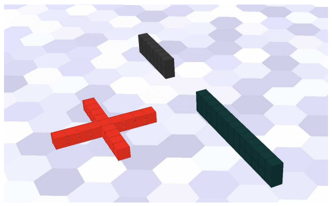


Figure 5.2: A snapshot from the Voxcraft simulator. Red voxels are active voxels (i.e., actuators), while black voxels are immovable and do not actuate.

connects: the higher the two temperatures, the more the beam stretches, the lower the two temperatures, the more the beam contracts. The robot's environment contains objects constructed from voxels with unvarying temperatures (Figure 5.2).

Following Kriegman et al. (2021), we set $\alpha = 14.4714$ and simulate materials of $10\,\mathrm{kg} \cdot \mathrm{m}^{-3}$ density, Young's modulus of $104\,\mathrm{Pa}$, Poisson's ratio of 0.5, coefficient of static friction of 1, and coefficient of dynamic friction of 0.5. Each voxel is $0.01\,\mathrm{m}$ in length.

5.1.2.2 Sensing

We equip the voxels with touch, floor, and velocity sensors, and a central pattern generator. Touch and velocity sensors are as described in Chapter 2. Floor sensors perceive whether the voxel is touching the floor or not and return 1 if yes, and -1 if not. Finally, a central pattern generator—relevant to animal and robotic locomotion alike (Ijspeert, 2008)—inputs the value of $\sin(-2\pi k)$ at time step k. Each voxel thus has 5 sensors.

5.1.2.3 Controller

We formulate the controller architecture to enable the robot to predict affordances (such as the passability of an aperture) and take appropriate actions (attempt to pass through passable apertures) by local processing of sensed interactions between its voxels and external objects.

We do so by beginning with the distributed controller described in Chapter 2, consisting of ANNs (a closed-loop system), one for every voxel. We adopt the "homogeneous" variant presented in (Medvet et al., 2021), where the neural networks share the same parameters: because of the more compact search space, such homogeneous representation is comparable to one where parameters are different for every voxel. Moreover, as seen in Pigozzi et al. (2022), parameter sharing makes the controller architecture agnostic for the morphology: the number of input and output neurons and, thus, the number of parameters does not depend on the arrangement and number of voxels in the morphology.

There are 17 input and 2 output neurons (Figure 5.3). At time step k, every neural network outputs $a_k \in [-1, 1]$ and $v_k \in [-1, 1]$, a *vote* that is a confidence score of the voxel on what the affordance is at the robot level, according to its local observation.

Every neural network takes as input its local observation $\boldsymbol{x}_k \in \mathbb{R}^{17}$: the 5 sensor values described in Section 5.1.2.2, but also, to introduce a form of communication among the voxels, in the a_{k-1} outputs from its six adjacent neighbors of the previous time step and the v_{k-1} outputs from its six adjacent neighbors of the previous time step. If a neighbor is absent, we set those inputs to 0.

In preliminary experiments, we found that memory helped the evolution of successful behaviors. So, in the reported experiments, we implement each neural network as an Elman network, the simplest instance of stateful neural network (Elman, 1990):

$$\boldsymbol{h}_k = \sigma(\boldsymbol{W}_x\boldsymbol{x}_k + \boldsymbol{W}_h\boldsymbol{h}_{k-1} + \boldsymbol{b}_h) \tag{5.2}$$

$$\boldsymbol{y}_k = \sigma(\boldsymbol{W}_y\boldsymbol{h}_k + \boldsymbol{b}_y) \tag{5.3}$$

where $\boldsymbol{h}_k \in \mathbb{R}^{n_{\text{mem}}}$ is a memory vector and $\sigma(\cdot)$ is an element-wise activation function. An Elman network is a three-layer neural network with a recurrence in the second layer. The first layer has one input neuron for every input; the second layer (Equation (5.2)) computes a non-linear combination of the inputs and the memory of the

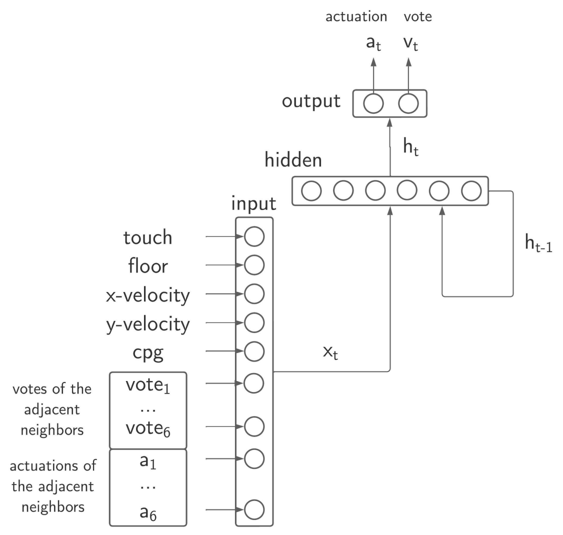


Figure 5.3: The architecture of the Elman network. There is one such network embedded within every voxel. We list the outputs on top and the inputs to the left: these include the vote and the actuation of the six adjacent neighboring voxels of the previous time steps.

previous time step $\boldsymbol{h}_{k-1}$ to output $\boldsymbol{h}_k$, which it then feeds to the third layer (Equation (5.3)) for the final output. $\boldsymbol{h}_k$ then becomes the memory vector for the next time step and the network unfolds over time, hence its recurrent nature. $\boldsymbol{W}_x \in \mathbb{R}^{n_{\text{mem}} \times 17}$, $\boldsymbol{W}_h \in \mathbb{R}^{n_{\text{mem}} \times n_{\text{mem}}}$, $\boldsymbol{b}_h \in \mathbb{R}^{n_{\text{mem}}}$, $\boldsymbol{W}_y \in \mathbb{R}^{2 \times n_{\text{mem}}}$, and $\boldsymbol{b}_y \in \mathbb{R}^2$ are the evolvable parameters of the network. We summarize the architecture of our Elman network in Figure 5.3.

After preliminary experiments, we set $n_{\text{mem}} = 6$, $\boldsymbol{h}_0 = \boldsymbol{0}$, and tanh as activation function (to ensure the output lies in $[-1, 1]$).

We optimize a robot for a task by optimizing the vector $\boldsymbol{\theta} = [\boldsymbol{W}_x \; \boldsymbol{W}_h \; \boldsymbol{b}_h \; \boldsymbol{W}_y \; \boldsymbol{b}_y] \in \mathbb{R}^p$ of parameters that specify every neural network in the robot. For our setting, $p = 158$.

Voting mechanism. At every time step k, the robot performs affordance detection by predicting whether the aperture in front of it is passable or not. It does so as follows.

(a) If voxel i's vote $v_{i,k}$ is greater than 0, this denotes the voxel predicting that the aperture is passable, and we set $v_{i,k}$ to 1. Otherwise, we set $v_{i,k}$ to 0, denoting a prediction that the aperture is impassable;

(b) We compute the majority vote across all voxels and store it in v_k. If there is a split vote, we set v_k to 1.

5.1.3 Experimental procedure

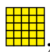
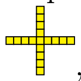
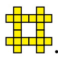
Our goal is to answer the following research questions:

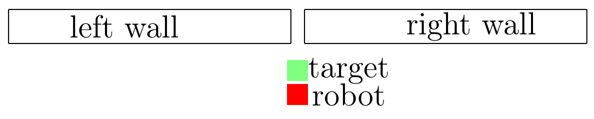
RQ1 Can affordance detection be evolved for soft robots?

RQ2 If yes, do some robot morphologies allow for the evolution of more accurate affordance detection than others?

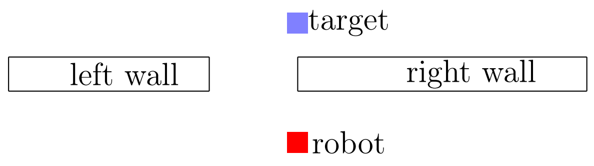
To this end, we evolved three different robot morphologies to perceive whether an aperture is passable and whether they take the appropriate action based on this conclusion (Section 5.1.3.2).

5.1.3.1 Robots

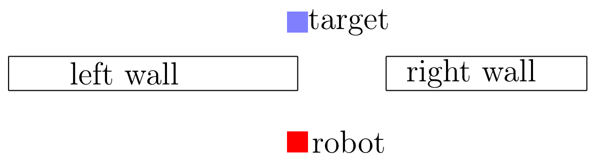
We experimented with three different robot shapes: a 5×5 square of voxels (the "flatworm") , a 9×9 four-legged "starfish" , and a 6×6 "gecko" . The controller inputs, outputs, and parameters are the same for all voxels in all three shapes. The three shapes entail different morphologies: the flatworm has no salient features on its body, the starfish has four limbs, while the gecko has many protrusions that it can potentially use as hooks (hence the name, because geckos climb on trees through tiny hooks on their finger palms).



(a) impassable



(b) passable-to-the-left



(c) passable-to-the-right

Figure 5.4: The robot, with body length l_{body} voxels, faces three different environments. In the first, it must detect that the aperture is impassable, and avoid attempting to move through it: it should minimize its distance from the green point at the end of the simulation. In the second and third environments, the robot should detect that the aperture is passable, pass through it, and minimize its distance from the blue point at the end of the simulation.

5.1.3.2 Task

We initially place the robot in front of an aperture formed by two walls (Figure 5.4). We evaluate the robot multiple times in the presence of apertures of differing widths and positions. The robot's task is to move to a target position on the far side of the aperture if it judges it passable, and to a target position before the aperture (but not at) if it judges it impassable. We build the walls with voxels having the same material properties as the robot. We design the task such that the decision to navigate the aperture or not is dependent on the aperture width and the robot's body length.

If we evaluated each $\boldsymbol{\theta}$ in one environment with a passable aperture and another with an impassable one, the robot could evolve degenerate behaviors rather than affordance detection. For example, the robot could evolve to reach out toward just one of the two walls. Since the wall would be at a different relative position from the robot in the two environments, different parts of the robot would collide with the wall in the two environments. The robot could evolve the ability to predict whether the aperture is passable just from this information, but fail to generalize to new environments. So, we evaluated each $\boldsymbol{\theta}$ in three environments as shown in Figure 5.4.

For a robot with a body length l_{body} voxels long, we set the two walls l_{body} voxels in front of the robot's most anterior voxel. In the first environment, we set the aperture width to 1 voxel, making it impassable, and center it in front of the robot. In the second environment, we set the aperture width to $l_{\text{body}} - 1$ voxels, making it passable, because the robot can move and deform itself. To minimize the search space, we offset the aperture to the left by $\lceil l_{\text{body}}/2 \rceil$ voxels. The third environment is the same as the second except that we offset the aperture $\lceil l_{\text{body}}/2 \rceil$ voxels to the right.

5.1.3.3 Fitness functions

As we wish to evolve robots that attempt to pass through passable apertures, and also explicitly predict whether apertures are passable or not, we must formulate two separate fitness objectives. We thus introduce a bi-objective evolutionary optimization problem with a "locomotion" fitness objective f_{loc} and an "affordance detection" fitness objective f_{aff}.

For the i-th of the three environments, we measure the distance d_i of the robot's center of mass from the relevant target position at the end of the simulation. Given that our EA requires bounded objective values (see Section 5.1.3.4), we set $d_i = d_{\max}$ if $d_i > d_{\max}$. After preliminary experiments, we found $d_{\max} = 5$ to be sufficient. Then, $d_i \in [0, 5]$.

For the i-th of the three environments, we also measure:

$$\mathrm{acc}_i = 1 - \frac{1}{t_{\mathrm{final}}} \sum_{k=t_{\mathrm{contact}}}^{t_{\mathrm{final}}} |v_k - g_i| \tag{5.4}$$

where t_{final} is the total number of time steps in the simulation, t_{contact} is the time step at which the robot first touches the walls, v_k is the robot's prediction about the aperture's passability (see Section 5.1.2.3), and $g_i \in \{0, 1\}$ is the ground truth for the i-th environment (0 for impassable and 1 for passable) and is available only to the fitness function, not the robot. The summation on the right of Equation (5.4) counts the number of time steps that the robot's majority vote (see Section 5.1.2.3) differs from the ground truth. Thus, Equation (5.4) is the accuracy for the binary classification problem of discriminating between passable and impassable environments. Then, $\mathrm{acc}_i \in [0, 1]$. Given that we equip the robot with minimal tactile sensors (see Section 5.1.2.2), no vote on the passability of an aperture is meaningful without first touching it. With this in mind, we start counting votes only after the robot has perceived its first contact with the walls at t_{contact}; in this way, dividing by t_{final} indirectly rewards the individuals for approaching the walls as soon as possible. If the robot never touches the walls, we set $\mathrm{acc}_i = 0$.

After evaluating the robot in all three environments, we set f_{loc} and f_{aff} to the worst of the robot's three actions and the worst of its three attempts at affordance detection, respectively:

$$f_{\mathrm{loc}} = \max_{i=1,2,3} d_i \tag{5.5}$$

$$f_{\mathrm{aff}} = \min_{i=1,2,3} \mathrm{a}_i \tag{5.6}$$

to reward robustness across all environments.

Finally, we found some individuals achieved high fitness while ignoring the aperture by crashing into the walls to climb over them, or by circumventing the walls altogether. To punish these exploitative behaviors, we assigned the worst possible fitness values (i.e., $d_i = 5$ and $\mathrm{acc}_i = 0$) to any individual that, at any time step, has more than half of its voxels outside of the bounding volume of the environment. The bounding volume of the environment is $3l_{\text{body}}$ voxels wide, $\left\lceil \frac{l_{\text{body}}}{3} \right\rceil + 1$ voxels tall, and $5l_{\text{body}}$ voxels deep.

5.1.3.4 Evolutionary algorithm

EAs are effective multi-objective optimizers (Zhou et al., 2011) and have proved competitive on continuous control tasks (Such et al., 2017b; Risi and Stanley, 2019). We employed the Non-dominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al., 2002), an established bi-objective EA. NSGA-II evolves a population of individuals by iterations according to a $\mu + \lambda$ generational model. We initialize individuals at the very first iteration by uniform sampling from the interval $[-1, 1]^p$. At each iteration, NSGA-II sorts the population into Pareto shells according to the two objectives f_{loc} and f_{aff}. As the genetic operator, we used Gaussian mutation of mean $\mathbf{0} \in \mathbb{R}^p$ and step-size σ_{mut}; as the selection operator, we used crowded tournament selection of size n_{tour}. We iterate until n_{evals} fitness evaluations have been computed.

The output of the above procedure is a set of Pareto-optimal individuals, namely, those belonging to the first Pareto shell: of these, we label as "specialists" those that are the best in one particular objective. We then have two specialist individuals $\boldsymbol{\theta}^*_{\text{loc}}$ and $\boldsymbol{\theta}^*_{\text{aff}}$ according to our two objectives f_{loc} and f_{aff}. We are also interested in individuals who perform well for both objectives. Then, we label as "knee" the individual that is the closest (in the fitness space) to the center of the bounding box of the first Pareto shell. We then have one knee individual $\boldsymbol{\theta}^{\text{k}}$. After preliminary experiments and relying on our experience, we set $\mu = \lambda = 50$, $\sigma_{\text{mut}} = 0.35$, $n_{\text{tour}} = 5$, and $n_{\text{evals}} = 10\,000$ (corresponding to 200 iterations).

5.1.3.5 Experimental settings

For every experiment in this work, we performed 10 evolutionary runs by varying the random seed for the EA. We performed all statistical tests with the

Mann-Whitney U rank test for independent samples. As a soft-body physics engine, we employed Voxcraft (Liu et al., 2020), written in C++ and CUDA, and we developed a Python wrapper for the evolutionary optimization. Simulation frequency is 200 Hz and we set each simulation to 50 s of simulated time, for a total of $t_{\text{final}} = 10\,000$ time steps. All simulations are deterministic for a given random seed and controller. The code is publicly available at `https://github.com/pigozzif/Voxcraft-python.git`.

5.1.4 Results

In the following, we provide results for both research questions.

5.1.4.1 RQ1: can affordance detection be evolved for soft robots?

Before delving into whether and how morphology affects a robot's ability to detect affordances, we first verify if we can evolve affordance detection in soft robots. To test this question, we conducted quantitative analysis by comparing the behaviors of robots controlled by random or evolved controllers, and qualitative analysis by inspecting their behaviors. For this question, we only considered the starfish robot.

5.1.4.2 Quantitative analysis.

As performance indexes, we measure the fitness values for the specialist individuals, namely f^*_{loc} for $\boldsymbol{\theta}^*_{\text{loc}}$ and f^*_{aff} for $\boldsymbol{\theta}^*_{\text{aff}}$, but also, the fitness values for the knee individual, namely $f^{\text{k}}_{\text{loc}}$ and $f^{\text{k}}_{\text{aff}}$ for $\boldsymbol{\theta}^{\text{k}}$.

As baselines, we compare our methodology with random controllers. After preliminary experiments, we found controllers that were random in both the outputs (actuation and affordance detection) to perform very poorly: the vast majority of them failed to touch the walls and perform any affordance detection at all. Thus, for a fairer comparison, we devised two controls, one for every fitness function: one control where we randomize the controller output for actuation and one control where we randomize the controller output for affordance detection. We optimize the two controls using the same EA of Section 5.1.3.4 and compare them with evolution where neither of the outputs is randomized (i.e., our approach). As a result, we have three different treatments:

EvolvedBoth: our approach, where neither of the outputs is randomized;

EvolvedAffordance: at every time step and for every voxel of every individual, we discard the output a_t controlling temperature change (and, thus, locomotion) and substitute it with a random number in $[-1, 1]$;

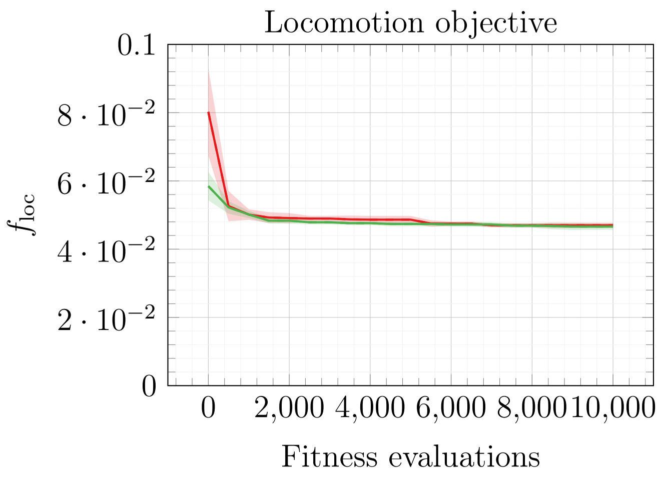
EvolvedLocomotion: at every time step and for every voxel of every individual, we discard the output v_t voting for affordance and substitute it with a random number in $[-1, 1]$.

We remark that all three have the same controller architecture and thus the same size of the search space.

We summarize the results in Figure 5.5, which plots f_{loc} and f_{aff} in terms of median $\pm$ standard deviation throughout evolution, for the specialists and the knee individuals. Moreover, we compare in Figure 5.6 the Pareto shells at the beginning, the halfway, and the end of evolution for three exemplar runs of the BothEvolved treatment. The utopia point (the one optimizing both objectives) sits in the upper-left corner.

Our EvolvedBoth treatment evolves affordance detection specialists that are better at affordance detection than random controllers, and locomotion specialists that are better at locomotion than random controllers. The picture is similar if we look at the knee individuals. The lines of Figure 5.5 also suggest that all treatments settle on a plateau and that more evolution would unlikely occur.

From Figure 5.5, we spot different patterns for the two fitness functions. focusing on the affordance detection fitness (lower side of Figure 5.5), the BothEvolved



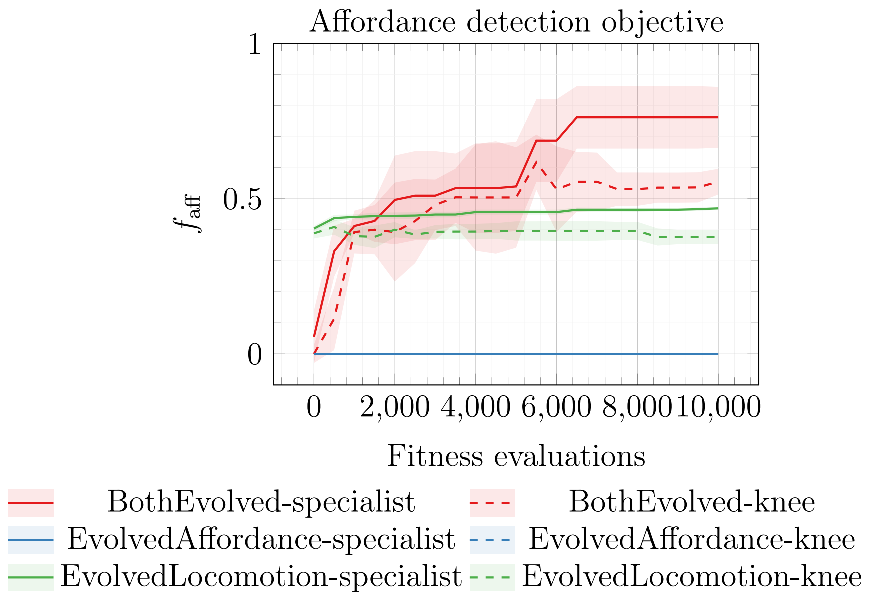


Figure 5.5: Median ± standard deviation (solid line and shaded area) for the two fitness values for the 10 specialist individuals and 10 knee individuals drawn from the 10 evolutionary runs, obtained with one experimental (EvolvedBoth) and two control (EvolvedAffordance and EvolvedLocomotion) treatments. The blue line on the upper plot is well above the upper bound and would have skewed the other lines. For the locomotion objective, specialists and knees mostly coincide. Our EvolvedBoth treatment evolves affordance detection significantly better than the two controls.

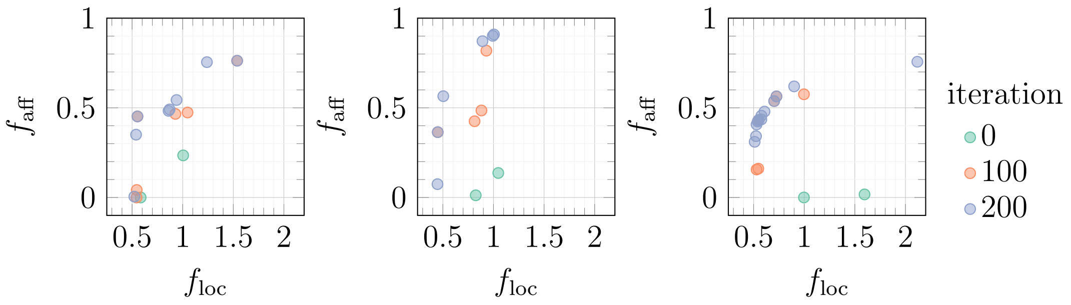


Figure 5.6: Pareto shells at different iterations (corresponding to the start, the halfway, and the end of evolution) for three exemplar evolutionary runs of the Both-Evolved treatment. The utopia point (the one optimizing both fitness functions) sits in the upper-left corner. As evolution progresses, Pareto shells approach the utopia point more and more.

treatment outclasses the two controls in terms of detection accuracy: by the end of evolution, its specialist individuals can, on average, correctly detect whether the aperture is passable or not at least 76.26 % of the time steps. That is significantly better than what the EvolvedLocomotion treatment evolves ($p < 0.0001$). If we look at the knee individuals, the affordance detection fitness is lower, but the p-value is still significant if we compare them with the EvolvedLocomotion treatment. Thus, evolution can find individuals that detect affordances significantly better than random controllers (the EvolvedAffordance line is stuck at the bottom because its specialists are unable to even reach the walls) and that, to some extent, are effective in both objectives.

Looking at the locomotion fitness (upper side of Figure 5.5), both the EvolvedBoth and the EvolvedLocomotion treatments evolve effective individuals (p-value not significant): they take less than 2000 fitness evaluations to evolve the behavior of reaching the target. Moreover, performance is similar for specialists and knee individuals. We do not show the EvolvedAffordance line (the blue line) as it would be stuck at such a high value to skew the remaining lines. We visually inspected the behaviors evolved by EvolvedAffordance and, not surprisingly, found them to drift away over the floor until well exceeding the d_{max} upper bound we set in Section 5.1.3.3. Thus, we have to evolve also the locomotion behavior.

From Figure 5.6, we conclude that evolution is indeed taking place in our EvolvedBoth treatment: as evolution progresses, individuals in the Pareto shell from early during the evolutionary process are dominated by individuals in later Pareto shells.

5.1.4.3 Qualitative analysis.

We analyze the evolved behaviors both in terms of locomotion and in terms of affordance detection.

For locomotion, an individual shows successful behavior if it reaches the target position and, in passable environments, correctly passes through the aperture in the walls.

We plot a top-view for the positions at the end of a simulation for 5 of the specialist individuals $\boldsymbol{\theta}^*_{\text{loc}}$ in Figure 5.7: for the same individual, we then have a red point for the impassable environment, a blue point for the environment passable to the left, and a green point for the environment passable to the right. Moreover, the dashed lines stand for the position of the walls.

From the figure, we conclude that effective locomotion behavior did indeed evolve. Moreover, knee individuals behave nearly the same as specialists. After visual inspection, not only we found them to dodge the walls and slip in the aperture, but also to eventually bend their trajectories to come as close as possible to the target. With the exception of one run, which resulted in a specialist that does not reach the target but passes through the aperture and drifts away, most of them reach and touch the target.

Our robots rely on minimal tactile feedback for sensing their environment. Then, an individual shows successful affordance detection behavior if a majority of the voxels not only stand for the correct vote even if only a few of them are sensing the walls but also if they retain the correct vote over time, possibly even after losing contact with the walls. Our distributed controller, having one neural network embed-

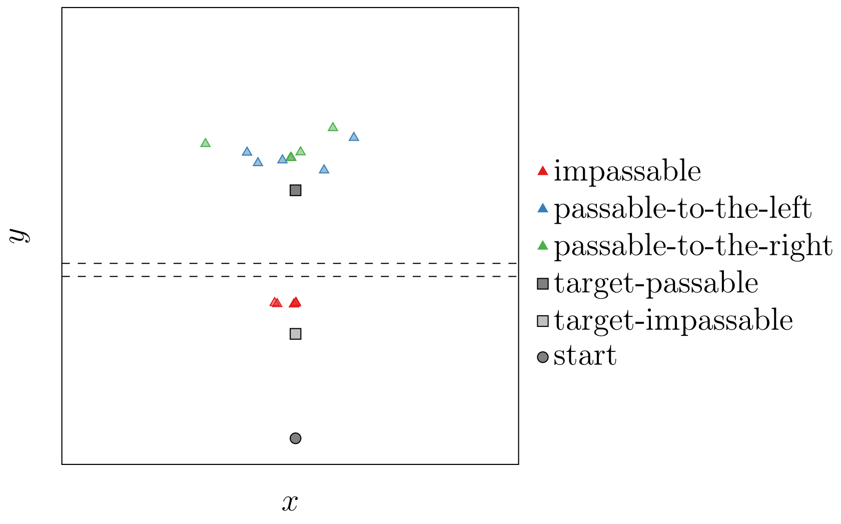


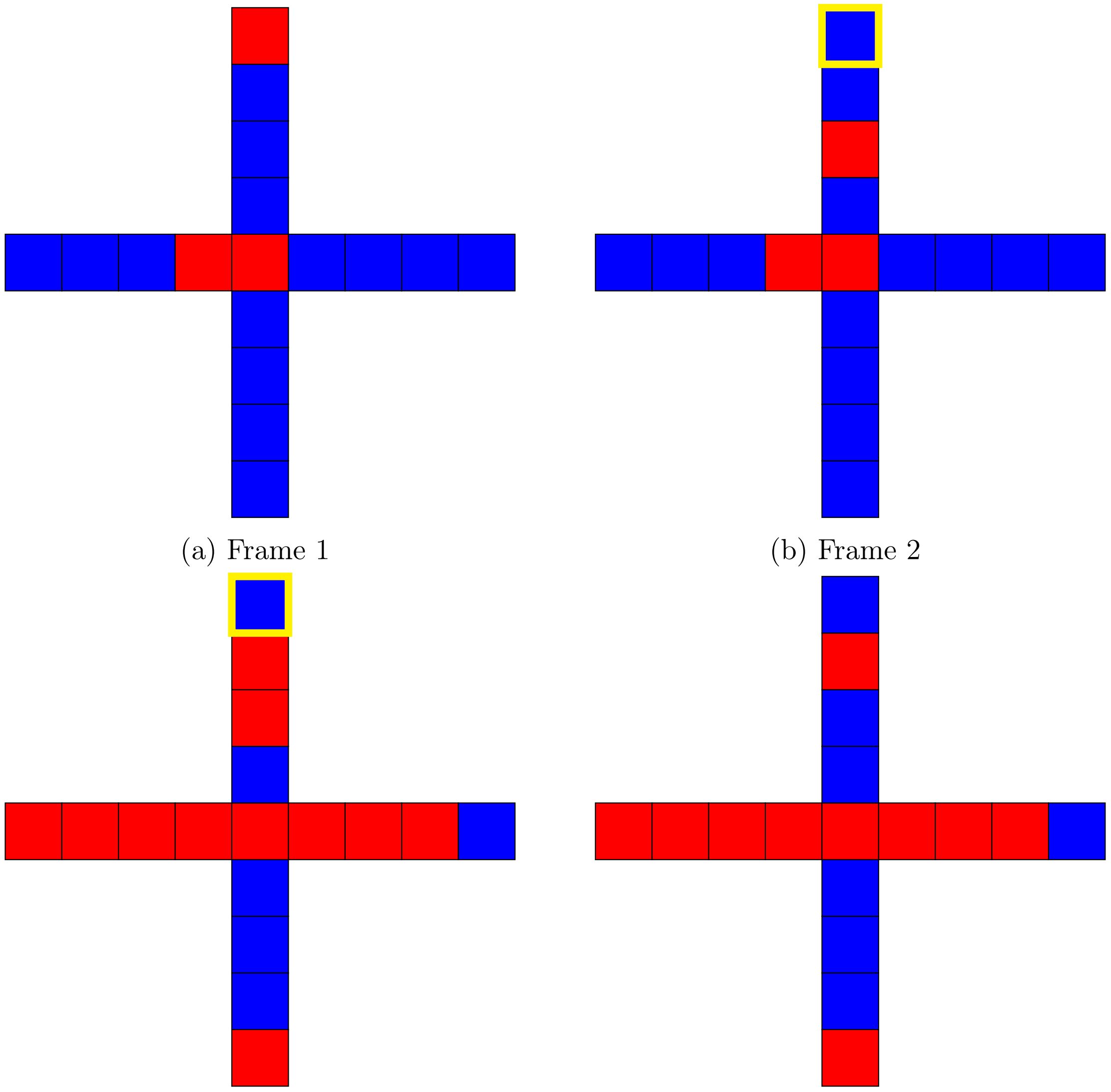
Figure 5.7: Top-view of the final positions of 5 among the locomotion specialists at the end of the simulation. The dashed lines stand for the position of the walls. In passable environments, the specialists evolve to eventually reach the target. In impassable environments, the robots stay on the near side of the walls: most of the red points overlap.

ded inside each voxel, allows us to observe in real time how voting shifts over time across the voxels.

We visually inspected the behaviors of the specialist individuals $\boldsymbol{\theta}^*_{\text{aff}}$ and depict the time-lapse for an exemplar specialist in Figure 5.8 and include more at `https://affordancesoftrobots.github.io/affordancesoftrobots/`. Red voxels vote for impassability, whereas blue voxels vote for passability. In addition, we border a voxel in yellow if its touch sensor is firing (i.e., it is contacting a wall). Frame 1 corresponds to the snapshot at the first time step and we take the others at regular intervals of 1500 time steps (approximately 7.5 s of simulated time).

We found the behaviors of the vast majority of specialist and knee individuals to consist of the same voting pattern. Those robots evolved to vote for passability by default and to switch to impassibility after detecting the pattern of sensory signals corresponding to the impassable environment, or vice versa.

Referring to Figure 5.8, this successful voting pattern unfolds as follows:


(a) Frame 1 (b) Frame 2 (c) Frame 3 (d) Frame 4

🟥 Impassable 🟦 Passable

Figure 5.8: Time-lapse for the voting of an exemplar specialist in the impassable environment. Voxels bordered in yellow are in contact with the walls at that time. Frame 1 corresponds to the snapshot at the first time step and we take the others at regular intervals of 1500 time steps (approximately 7.5 s of simulated time). Frame 1: when receiving no tactile feedback, the robot votes for passability. Frame 2: the robot contacts the wall and the tip's touch sensor sends feedback. Frame 3: the correct vote propagates from the tip to the rest of the body. Frame 4: a majority of the voxels retain the correct vote even after the robot loses contact with the wall.

Frame 1: At the beginning, starting apart from the walls, the robot receives no tactile feedback from its environment. At this point, individuals do not necessarily cast the correct vote (blue color in Figure 5.8). The robot moves forward.

Frame 2: Once the robot contacts the walls with its arm, the touch sensor located on the tip starts firing and sends its feedback to the neural network embedded in the tip voxel.

Frame 3: After a short "burn-in" period has elapsed, where voting happens randomly, we see the correct vote (the red color in Figure 5.8) propagate from the tip voxels to the rest of the body until flipping the majority towards the correct vote.

Frame 4: Crucially, a majority of the voxels retain the correct vote even after the touch sensor at the tip stops firing.

Through that evidence, we can answer positively to RQ1: we can evolve soft robots with the ability to detect their affordances.

5.1.4.4 RQ2: do some robot morphologies allow for the evolution of more accurate affordance detection than others?

To investigate whether and how morphology affects the evolution of affordance detection, we performed 10 additional evolutionary runs for each of the two additional morphologies: the "flatworm" and the "gecko" (see Section 5.1.3.1). We analyze and compare the results with different morphologies both quantitatively and qualitatively using the same methodology of Section 5.1.4.1. By the distributed controller, all three morphologies entail the same size of the search space, since the controller architecture does not depend on the number of inputs and outputs, as well as the number and arrangement of voxels.

We summarize the results in Figure 5.9, which plots f_{loc} and f_{aff} in terms of median $\pm$ standard deviation throughout evolution, for both the specialist and the knee individuals.

5.1.4.5 Affordance detection.

For f_{acc}, the three morphologies differ both in terms of efficiency (i.e., speed of convergence) and in terms of performance of the final specialist and knee individuals. On one side, geckos quickly converge to a very high fitness value, to the point of outclassing the starfishes; on the other side, the f^{*}_{acc} for flatworms plateaus at around 50 %: this is not statistically different from mean f^{*}_{acc} obtained by random controllers. Starfishes evolved f_{acc} values between those of flatworms and geckos. We found the p-values to be significant among different morphologies for both specialists and knee individuals. Finally, we repeated the same EvolvedLocomotion control experiments of Section 5.1.4.4 for the gecko and flatworm morphologies: geckos are statistically better than their random counterparts on f_{acc}, while the same is not true for the flatworms (p-value not significant).

These results suggest that robots with more complex morphologies (e.g., the gecko), where we define complexity simply by the number of parts comprising the robot, might evolve more accurate affordance detection than those with simpler morphologies (e.g., the flatworm). To gain deeper insights into this intuition, we visually inspected the evolved behaviors. We found geckos to exhibit voting patterns similar to those discussed for the starfishes in Section 5.1.4.1: one or a few tips perceive the wall in front of them, propagate the correct vote across the body which retains it even in case the tips lose contact with the walls. Contrarily, we did not find signs of evolution in the flatworms: their voting patterns are random and chaotic. `https://affordancesoftrobots.github.io/affordancesoftrobots/` shows one instance of this.

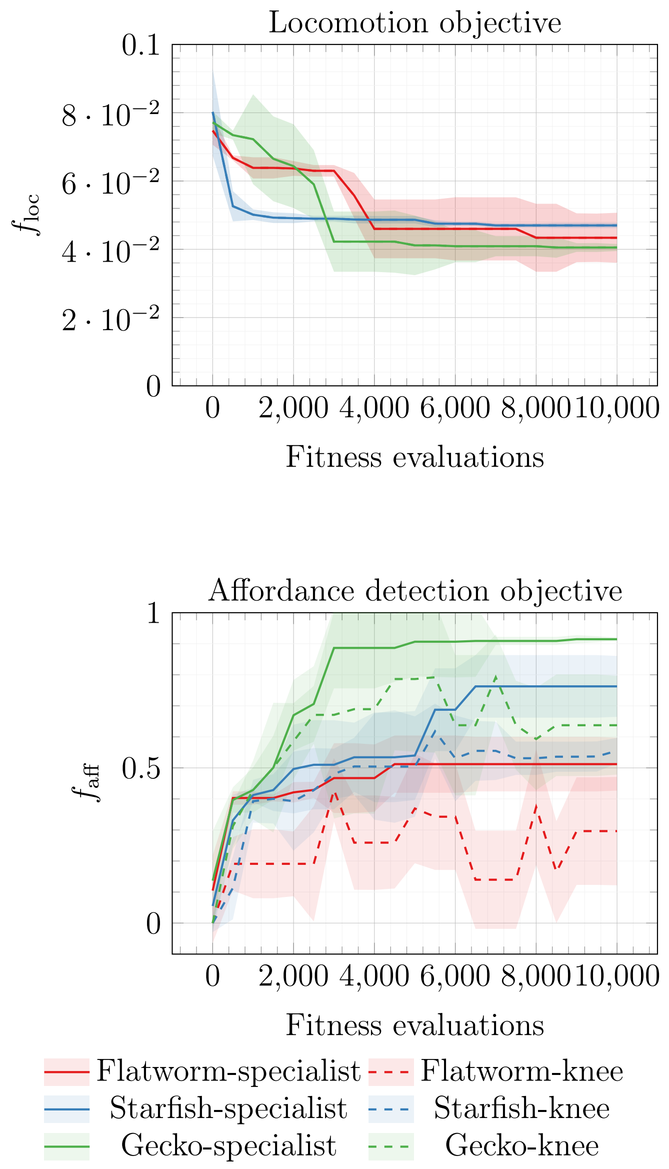


Figure 5.9: Median ± standard deviation (solid line and shaded area) of the two fitness values for the 10 specialist individuals and the 10 knee individuals drawn from the 10 evolutionary runs, obtained with three morphologies. For the locomotion objective, specialists and knees mostly coincide. The flatworm struggles to evolve affordance detection behavior, while the gecko outperforms the other morphologies.

5.1.4.6 Locomotion.

The difference among morphologies is less pronounced when we look at the locomotion fitness (upper side of Figure 5.9): the flatworm shape is thus not deficient per se, as it can evolve to locomote, but it is deficient for affordance detection.

These results suggest that different morphologies can facilitate the evolution of affordance detection behavior. One intuitive explanation is that, in the flatworm, it is more difficult for each voxel to distinguish itself: indeed, recalling that absent neighbors provide inputs of 0, a voxel can distinguish itself by the pattern of absent neighbors, but the majority of voxels in the flatworm have the same neighborhood pattern. Thus, different environments result in patterns of sensing that are more difficult to discriminate. Alternatively, a robot composed of more parts, or that can radically change its shape, may obstruct the evolution of affordance detection because self-contortions can more easily or rapidly alter the kinds of affordances afforded by objects in its environment. Altogether, these insights suggest that morphology plays an important role in the evolution of affordance detection. Albeit we here considered only a few robot body plans, and they remained fixed during evolutionary optimization, in the future, joint optimization of morphology and control may facilitate the evolution of affordance detection in robots and thus improve their reliable and safe action in the world.

5.1.5 Concluding remarks

Considering the case of VSRs—whose softness makes the interactions between their bodies and the environment less predictable—we address the question of whether a modular agent can, by local information processing only, detect the affordances of its body and what role the morphology plays in facilitating it or not. We optimize VSRs to decide whether or not their bodies can fit through an aperture, and, if so, then attempt to pass through it. We evolve the parameters of their closed-loop controllers and experiment with three morphologies. Our results show that different morphologies can facilitate the evolution of affordance detection behavior: some are effective, while others are not and we conjecture the reason to be their differing degrees of complexity. This work demonstrates that sensation and control are necessary but not sufficient for understanding how affordance detection evolves in organisms or robots: we must also take into account morphology.

5.2 How the morphology encoding influences learning ability in body-brain co-optimization

In this chapter, we answer to question **4**.

5.2.1 Introduction

Starting from the seminal work of Sims (1994), researchers have resorted to evolution for co-optimizing embodied agents' bodies and brains. Still, the embodied cognition paradigm (Pfeifer and Bongard, 2006), which posits a deep entanglement between the brain and the body, challenges our ability to do so: parent agents often beget offspring having a mismatch between the morphology and the inherited controller (Eiben and Hart, 2020b). On the other side, learning alone is myopic since adapting the morphology is also essential (Hart and Le Goff, 2022). As a result, co-optimization, by embedding the learning of controllers within the evolution of morphologies (Gupta et al., 2021; Luo et al., 2021), has come into practice, since it gives time for controllers to adapt to their morphologies. Intuitively, that is how nature shaped animal life on Earth. How to design a co-optimization setting is however a complex endeavor; the designer must select—among the others—the genetic encoding for the morphology, which is of crucial importance for evolution (Cheney et al., 2013; Veenstra et al., 2017).

We evolve the morphologies of VSRs using EAs (De Jong, 2006) while learning the controllers with RL (Sutton and Barto, 2018). We resort to VSRs as they provide so many more degrees of freedom and flexibility than traditional robots, providing

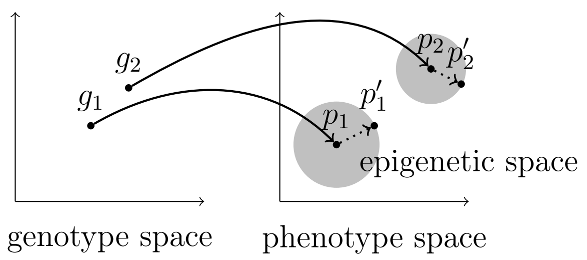


Figure 5.10: The intuition of plasticity. As a matter of example, we show a two-dimensional genotype space together with a two-dimensional phenotype space. A genotype g_i maps to a phenotype p_i, which is plastic: through learning, it can visit any point in its epigenetic space (the shaded circle) during its lifetime, to finally settle to a learned phenotype p'_i. The learning trajectories (the dotted lines) need not be straight.

ideal testbeds for theories related to embodied cognition (Kriegman et al., 2018). We experiment with two genetic encodings for the morphology, a direct and an indirect, and focus our analysis on how *phenotypic plasticity*, which is the result of the interaction between evolution and learning in embodied agents, affects evolution in the two encodings.

In biology, phenotypic plasticity is the capacity of a phenotype to describe a trajectory in the phenotype space during its lifetime; in other words, plasticity enables an agent to explore neighboring regions of the phenotype space. Plasticity smooths the fitness landscape since it eases climbing to peaks (Price et al., 2003). The most straightforward instance of plasticity is lifetime learning (Kelly et al., 2012), also known as the epigenetic timescale of adaptation (Sipper et al., 1997). Thus, learning defines an epigenetic space, the subset of points in the phenotype space that a starting phenotype (i.e., right after genotype-phenotype mapping) can reach through lifetime learning. We illustrate the intuition for plasticity in Figure 5.10.

Plasticity is relevant because it does impact evolution: learning smooths the fitness landscape, but it also incurs in costs, as it is time-consuming and relies on trial-and-error, and there is evidence that biological evolution selects for morphological traits that reduce the cost of (i.e., speed up) learning (Weber and Depew, 2003). In other words, if there is an advantage to making a behavior "rigid" (or, not plastic), it will usually be advantageous to do so, since rigid behaviors are less expensive than plastic (learned) ones. As a homage to its first supporter, James Mark Baldwin, this phenomenon has been labeled—on how plasticity impacts evolution—the *Baldwin effect* (Baldwin et al., 2018; Baldwin, 1897). We remark that the Baldwin effect is not an instance of Lamarckism (Turney, 2002).

We experiment with three tasks from the EvoGym benchmark (Bhatia et al., 2021), consisting of bridge walking, beam toppling, and cave crawling, and analyze how the direct and indirect morphology encodings affect plasticity, namely, the ability to learn. To do so, we measure the trend for speed of learning, i.e., how quickly a phenotype travels in the epigenetic space, but also how much the agent can learn, measured as the radius of the epigenetic space.

Our results highlight differences: the direct encoding results in increased learning ability, while the same is not always true for the indirect encoding. We link these results to the Baldwin effect: apparently, direct encoding is better suited to facilitating it. We attribute the reason to the lower locality (Rothlauf, 2006) and heritability (De Carlo et al., 2021) of the indirect encoding: since similar genotypes do not always correspond to similar phenotypes, it is more difficult for evolution to select for morphological traits that reduce the cost of learning.

We believe our work to be a stepping stone on the road toward a better understanding of the interactions between evolution and learning. In particular, our work can reveal insights into the choice of morphology encoding when co-optimizing agents' bodies and brains with evolution and learning.

5.2.2 Related work

Hinton et al. (1987) first exposed how evolution and learning can complement each other. In their seminal study, learning made it possible for evolution to search on a deceptive fitness landscape. Since then, researchers have investigated the marriage between the two for—among the others—evolving reward functions (Niekum et al., 2010), population-based training (Jaderberg et al., 2017), evolving instinctive behaviors for RL (Hallawa et al., 2021), and optimizing neural networks (Stork et al., 2021).

However far-reaching they might be, only a few of those works considered the interaction between evolution and learning in the case of embodied agents. Those that do usually embed a learning loop inside an outer evolutionary loop, as happened with RL (Gupta et al., 2021), inherited controller archives (Goff and Hart, 2021), inner evolutionary optimization (Miras et al., 2020a), adaptive weights (Ferigo et al., 2022a; Pigozzi et al., 2022), or even Lamarckian evolution (Jelisavcic et al., 2019), and these works are undoubtedly groundbreaking. Still, there are even fewer works that consider how evolution impacts plasticity; and even fewer considering explicitly the ability to learn. Gupta et al. (2021) detected signs of a Baldwin effect in tree-based robots and Luo et al. (2021) in modular robots. Kriegman et al. (2018) studied how morphological development leads to differential canalization of morphological and behavioral traits. Thus, to the best of our knowledge, no other work has explored how phenotypic plasticity changes according to the encoding of the morphology, that is precisely the aim of this work. Less recently, Veenstra et al. (2017) compared a direct and a generative encoding for a different kinds of modular robots, in particular in terms of their ability to foster the evolution of robots composed of different numbers of modules.

5.2.3 Agent model

We employ the open-source discrete-time and continuous-space simulator Evolution Gym (EvoGym) (Bhatia et al., 2021), which also provides a number of benchmark tasks. Contrary to other chapters, we rely on EvoGym since it is a Python software, thus easing the integration of RL algorithms. In EvoGym, agents are 2D VSRs, composed as aggregations of elastic squared blocks, the voxels. EvoGym models each voxel as a spring-and-mass system, with point masses at the vertices and springs to join them. Each voxel can be of one of four different *materials*:

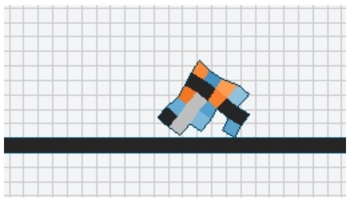


Figure 5.11: An EvoGym VSR. The voxel color stands for the material: black is rigid inactive, gray is soft inactive, orange is a horizontal actuator, and cyan is a vertical actuator.

(a) Rigid inactive, that does not change its area;

(b) Soft inactive, that passively contracts or expands under the contact with external bodies;

(c) Horizontal actuator, that actively contracts or expands horizontally according to an actuation signal;

(d) Vertical actuator, that actively contracts or expands vertically according to an actuation signal.

Figure 5.11 depicts one of the EvoGym agents. For an agent, we denote the total number of voxels made of one of the materials as $n_{\text{pass-rigid}}$, $n_{\text{pass-soft}}$, $n_{\text{act-h}}$, and $n_{\text{act-v}}$. Moreover, a morphology has a total of n point masses.

An agent has a *morphology* (i.e., a body), described by a matrix $\boldsymbol{M} \in \{\varnothing, \text{pass-rigid}, \text{pass-soft}, \text{act}$
where w and h are the width and height of the largest grid enclosing the morphology. $\boldsymbol{M}_{ij}$, the element at position i, j, tells which of the four materials the voxel at position i, j is made of, or takes the value $\varnothing$ if no voxels are present at that position.

An agent also has a *controller* (i.e., a brain), described by a function taking as input $\boldsymbol{o}^{(k)}$ and outputting $\boldsymbol{a}^{(k)}$, where $\boldsymbol{o}^{(k)} \in \mathbb{R}^p$ is an observation vector at time step k, with p task-specific inputs, while $\boldsymbol{a}^{(k)} \in [0.4, 1.6]^{n_{\text{act-h}}+n_{\text{act-v}}}$ is an action vector at time step k, with one action for every actuator. $a_i^{(k)}$ corresponds to the action for the i-th actuator at k and instructs an instantaneous change in springs that is a_i times their resting length, thus causing the actuator to contract or expand accordingly.

Finally, the agent performs a *task*, described by the environment (including terrain and objects). EvoGym models terrain and objects as aggregations of inactive voxels (either rigid or soft). The task provides a reward signal $r^{(k)} \in \mathbb{R}^1$ that is task-specific and captures the completeness of the task by the agent. The task also defines what observation $\boldsymbol{o}^{(k)}$ to feed the controller with. Observations are task-specific and can be sensor readings (e.g., velocity), terrain information (e.g., distance from cave ceiling), or goal information (e.g., distance from an object to manipulate).

We instantiate the controller using an MLP, having as many input neurons as the number of observations for the task and as many output neurons as the number of actuators; in this way, the controller is closed-loop and can exploit sensor readings, that are fundamental for complex tasks like walking on uneven terrain or object manipulation (Talamini et al., 2019). Following (Bhatia et al., 2021), we set the architecture to have two hidden layers with 64 neurons each, and tanh as the activation function for all neurons (we then rescale the output to $[0.6, 1.6]$).

5.2.4 Co-optimization of morphology and controller

Since we focus on the interaction between learning and evolution, we cast the problem of optimizing an agent for a task as a co-optimization problem: an outer optimization loop searches in the space of morphologies, and an inner optimization loop searches in the space of controllers (for a given morphology). We resort to EAs for the outer optimization, as they are gradient-free algorithms and can effectively handle discrete spaces (De Jong, 2006), as well as generating emergent patterns for the morphology of virtual creatures (Cheney et al., 2013; Corucci et al., 2018). Since we have access to a reward signal over the agent's lifetime, we resort to RL for the inner optimization; moreover, RL has achieved state-of-the-art results with neural controllers (Mnih et al., 2015) and there is evidence it mimics learning in the animal

[1] r previously indicated other objects in the text; with a slight abuse of notation, we repurpose it to be a reward.

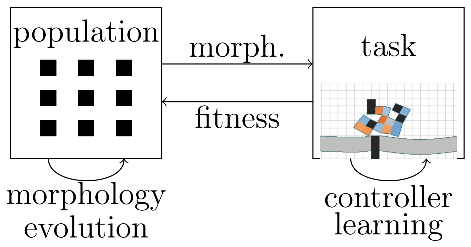


Figure 5.12: Overview of the co-optimization.

brain (Neftci and Averbeck, 2019).

In particular, the EA optimizes a fixed-size population of n_{pop} genotypes by iterating over generations. At every generation, we map genotypes to morphologies and optimize the controller of each morphology for $n_{\text{RL-iters}}$ iterations of RL. After ranking and selecting the parent morphologies with a fitness function, we beget an offspring. We use as fitness function the mean reward of an *episode* (i.e., a simulation of the agent in the environment starting from a task-specific initial condition) of n_{sim} time steps, done "off-line" at the end of RL:

$$f = \frac{1}{n_{\text{sim}}} \sum_{k=1}^{n_{\text{sim}}} r^{(k)} \tag{5.7}$$

We iterate until n_{evals} fitness evaluations have been done. Figure 5.12 is a schematic view of our co-optimization.

To investigate how morphology encoding affects plasticity, we experiment with two variants of encoding (a direct and an indirect encoding), while employing the same outer-inner optimization scheme.

5.2.4.1 Evolution

For the morphology, ER researchers usually resort to either direct encodings, where there is a one-to-one mapping between the genotype and the phenotype, or indirect (also known as generative) encodings, where the mapping is non-trivial. Albeit less intuitive, indirect encodings can allow for the emergence of complex patterns (e.g., "tissues" in (Cheney et al., 2013)); moreover, they strive to emulate the biological genome (Tang et al., 2020; Pedersen and Risi, 2021), which is capable of compressing a whole phenotype with just a “few” genes (generally, in the order of thousands (Gregory et al., 2007; Pellicer and Leitch, 2019)) while still fostering phenotypic variation (Gerhart and Kirschner, 2007). We experiment with one variant for each type.

5.2.4.2 Direct encoding

Our direct encoding represents a morphology with the matrix $\boldsymbol{M}$ of Section 5.2.3 as genotype; then, each matrix entry encodes one voxel of the morphology.

For evolving $\boldsymbol{M}$, we employ a simple genetic algorithm (De Jong, 2006). It evolves a fixed-size population of n_{pop} individuals iterating the following two steps until n_{evals} have been done. First, it initializes the first population with randomly generated matrices, where each element is chosen with uniform probability in the proper domain. Then, at every generation, (i) it takes the best $0.2n_{\text{pop}}$ individuals (i.e., those with the best fitness), selects them as parents and (ii) it builds the offspring, i.e., the population for the next generation, by copying the parents to the offspring (a form of elitism) and by generating $0.8n_{\text{pop}}$ individuals by mutating randomly chosen (with uniform probability) parents. For the mutation, we change each element of the matrix to another material, with 0.1 probability. To decode the genotype into a morphology, we retain the largest connected component of non-empty voxels. We used the implementation of (Bhatia et al., 2021).

5.2.4.3 Indirect encoding

Several indirect encodings exist in the literature that are suitable for VSRs, e.g., L-systems (Hornby et al., 2001), gene regulatory networks (Joachimczak et al., 2016), and Gaussian mixtures (Hiller and Lipson, 2012), but we rely on the work of Cheney et al. (2013) as it proved very effective at evolving morphologies specifically for multi-material VSRs.

The genotype is a Compositional Pattern Producing Network (CPPN) (Stanley, 2007), a feed-forward neural network with three input neurons and four output neurons. To map a CPPN to morphology, we query the CPPN for every voxel (in the maximum enclosing grid of the morphology) by inputting the x- and y-coordinates of the voxel as well as its Euclidean distance from the center of mass of the maximum enclosing grid. The output of the CPPN is a one-hot encoding telling which material to fill the voxel with (5 output neurons, one for the no-material case). Finally, we retain the largest connected component of non-empty voxels.

We evolve CPPNs with the NeuroEvolution of Augmenting Topologies (NEAT) (Stanley and Miikkulainen, 2002) algorithm, an established EA that incrementally evolves the topology, the weights, and the activation functions of neural networks, begetting CPPN-NEAT. NEAT employs speciation to protect innovations and overcome the "competing conventions" problem (Branke, 1995) of evolving neural networks. NEAT also employs genetic operators specific for network structures (e.g., crossover with innovation); we refer the reader to Stanley and Miikkulainen (2002) for details. We used elitist selection as with GA. We used the implementation of (McIntyre et al.) and the same hyperparameters of (Bhatia et al., 2021), with the exceptions of n_{pop} and n_{evals}, for which we use the same values as for the direct encoding case.

5.2.4.4 Learning

For the controller, we resort to RL in order to explore a rich search space for the policy (i.e., the controller) and allow the agent to perform complex behaviors, that would hardly be achievable with other approaches (Cheney et al., 2013).

In particular, we employ the Proximal Policy Optimization (PPO) (Schulman et al., 2017) algorithm, a policy gradient method that has reached state-of-the-art performance in many continuous control tasks (OpenAI et al., 2019) while being simple to implement. PPO iterates for $n_{\text{RL-iters}}$ iterations. At each iteration:

(1) it collects environment interactions (i.e., triplets including the observation, the

action, and the reward) with the current policy;

(2) it updates the policy by minimizing a cost function, that takes into account the observed reward and ensures that the deviation from the current policy is relatively small, in order to eschew the high variance of other policy gradient methods.

In detail, PPO performs $n_{\text{RL-episodes}}$ episodes in parallel, each lasting n_{sim} time steps, and interrupting them every $n_{\text{sim-RL-iter}}$ time steps to perform an iteration—this way, $n_{\text{RL-episodes}} n_{\text{sim-RL-iter}}$ interactions are available for updating the policy. When the episodes reach the end, PPO restarts a batch of $n_{\text{RL-episodes}}$ until $n_{\text{RL-iters}}$ iterations have been performed. We used the implementation of (Kostrikov, 2018) and the same hyperparameter settings of (Bhatia et al., 2021): in particular, we set $n_{\text{RL-episodes}} = 32$ and $n_{\text{sim-RL-iter}} = 128$.

5.2.5 Experiments

We carried out an experimental campaign aimed at answering the following research question: how do a direct and an indirect encoding for the morphology affect the agent's ability to learn?

We perform a set of experiments, i.e., optimizations of VSRs, with both the direct and indirect encoding and three different EvoGym tasks, differing by scope and hardness: `BridgeWalker`, `BeamToppler`, and `CaveCrawler`.

For both encodings, we set $w = 5$, $h = 5$, $n_{\text{pop}} = 25$, $n_{\text{RL-iters}} = 1000$, and $n_{\text{evals}} = 500$; for `CaveCrawler`, which turned out more difficult to solve, we set $n_{\text{evals}} = 1000$. Pure evolutionary optimization would rely on more fitness evaluations; still, we remark that, in our co-optimization, every fitness evaluation amounts to a full RL inner optimization, increasing not only the computational burden but also the likelihood of discovering effective solutions earlier in evolution. We thus found the above values to work well, in line with (Bhatia et al., 2021).

For each experiment, we performed 10 evolutionary runs varying the random seed for the EA and PPO. For a given agent, all simulations are deterministic. We performed statistical tests with the Mann-Whitney U rank test for independent samples.

We made the code to repeat and reproduce the experiments publicly available at `https://github.com/federico-camerota/evogym/tree/baldwin`.

5.2.5.1 Tasks

In the following sections, we briefly describe the environments used in our experiments, we refer the reader to (Bhatia et al., 2021, Appendix B) for more details.

5.2.5.2 `BridgeWalker`

The agent learns a locomotion pattern that allows it to travel as far as possible on a soft rope bridge made up of several sections with different lengths. Let $x_a^{(k)}$ and $y_a^{(k)}$ be the x and y positions of the agent's center of mass at time step k. The reward scheme provides a positive reward equal to the agent movement in the positive x-axis:

$$r^{(k)} = x_a^{(k)} - x_a^{(k-1)}, \tag{5.8}$$

and a final reward of 1 upon reaching the end of the bridge. Figure 5.13a is a sample frame from the task. We adopted this task to be a test of basic cognition, while not as trivial as walking on flat terrain.

The observation vector $\boldsymbol{o}^{(k)} \in \mathbb{R}^{n+3}$ contains the current x and y positions of the n point masses (relative to the center of mass), the x and y velocities of the center of mass, and the orientation of the center of mass. Episodes in this environment consist of $n_{\text{sim}} = 500$ steps.

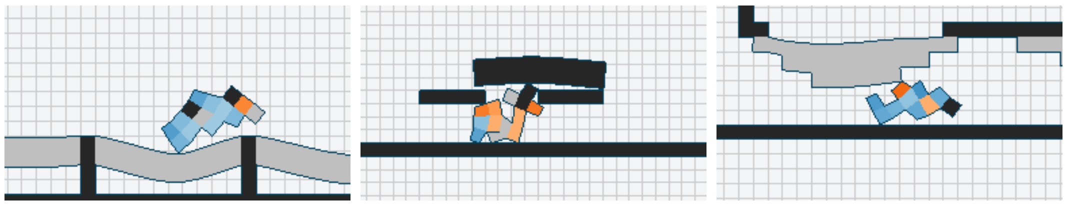

(a) `BridgeWalker` (b) `BeamToppler` (c) `CaveCrawler`

Figure 5.13: Three frames of agents performing the three tasks we considered in this chapter.

5.2.5.3 `BeamToppler`

The second task consists of flat terrain with a beam placed over two pegs. The agent's goal is to push the beam until it falls. Let $x_b^{(k)}$ and $y_b^{(k)}$ be the x and y positions of the beam's center of mass at time step k. The reward is the sum of three components:

$$r^{(k)} = r_1^{(k)} + r_2^{(k)} + r_3^{(k)}, \tag{5.9}$$

with:

$$r_1^{(k)} = |x_b^{(k-1)} - x_a^{(k-1)}| - |x_b^{(k)} - x_a^{(k)}| \tag{5.10}$$

$$r_2^{(k)} = |x_b^{(k-1)} - x_b^{(k)}| + 3|y_b^{(k-1)} - y_b^{(k)}| \tag{5.11}$$

$$r_2^{(k)} = y_b^{(k-1)} - y_b^{(k-1)}, \tag{5.12}$$

where r_1 rewards the agent for getting closer to the beam, r_2 for moving the beam, and r_3 for toppling the beam. Figure 5.13b is a sample frame from the task. We adopted this task as a relevant instance of object manipulation.

The observation vector $\boldsymbol{o}^{(k)} \in \mathbb{R}^{n+7}$ contains the same positional information about the point masses as in `BridgeWalker`, plus the x and y velocities of the beam, the orientation of the beam, and the distance of the agent's center of mass from the beam's center of mass. This environment runs for $n_{\text{sim}} = 1000$ steps.

5.2.5.4 `CaveCrawler`

The last task is the most difficult of the three and requires the agent to both learn a locomotion pattern and to adapt its shape to traverse caves with non-flat terrain while avoiding obstacles hanging from above. The reward function is the same as in `BridgeWalker`. Figure 5.13c is a sample frame from the task. We adopted this task as it is among those classified as challenging in (Bhatia et al., 2021).

The observation vector $\boldsymbol{o}^{(k)} \in \mathbb{R}^{n+24}$ contains the same positional information about the point masses as in the previous tasks as well as the agent's velocity, plus the lowest y positions of the ceiling and the highest y positions of the floor at $[x, x+1]$ voxels of distance from the agent's center of mass along the x-axis, $x \in \{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5\}$. As in `BeamToppler`, also `CaveCrawler` consists of $n_{\text{sim}} = 1000$ simulation steps.

5.2.5.5 Results

Our evolutionary reinforcement learning setting co-optimizes effective solutions for the three tasks. We also analyze how the learning (in terms of duration and extent) and the morphologies (using a few morphological descriptors) vary throughout evolution for the two encodings; witnessing different trends, we link them to the Baldwin effect and discuss the implications for designing encodings in the co-optimization of virtual agents with evolution and learning.

As a first step, we verify whether both encodings are capable of co-optimizing effective solutions. As the performance index, we use fitness f that, we recall, corresponds to the average reward of the evolved agent in a single episode. We plot in Figure 5.14 the fitness for the best individual of each generation throughout evolution, in terms of median $\pm$ standard deviation across the evolutionary runs.

Figure 5.14 confirms that co-optimization succeeds in finding effective solutions with both encodings and in all three task environments and that most experiments succeed in converging to a stable minimum. Direct and indirect encodings perform comparatively on `BridgeWalker` and `CaveCrawler` (p-values not significant), while the direct encoding outperforms the indirect in `BeamToppler` with $p < 0.1$. Last but not least, we notice that fitness curves follow radically different paths: the direct

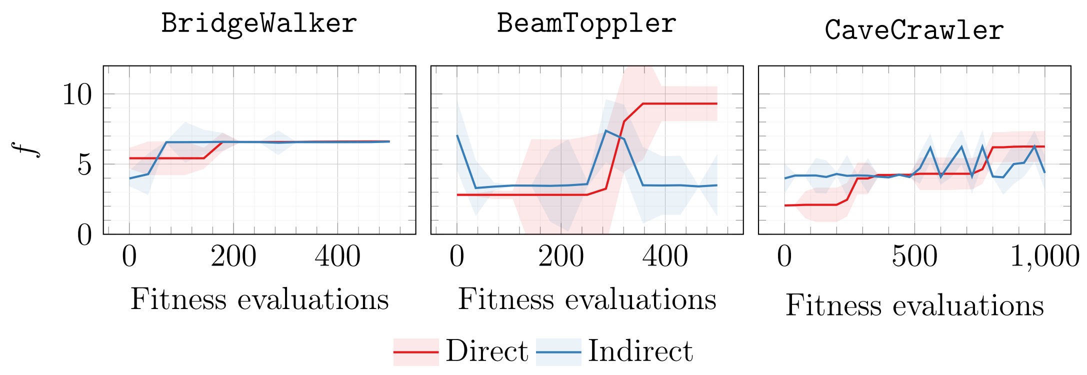
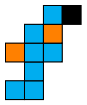


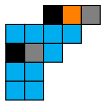
Figure 5.14: Median ± standard deviation across the evolutionary runs for the fitness f of the best individual throughout evolution. Both encodings optimize effective solutions.

encoding co-optimizes monotonously, stepping from one local minimum to a better one; on the other side, the indirect encoding shows more erratic performance.

We visually inspected the co-optimized solutions and found them to be highly adapted to the tasks: a video of the best individuals can be found at `https://youtu.be/jlbBOoprnPA`. Not only do individuals learn adaptive and life-like behaviors, but they also evolve morphologies that are suited to the task at hand. For example, the best individuals for the `CaveCrawler` task always have oblong morphologies, ideal for squeezing through tight spaces. In `BeamToppler`, champions usually rely on extrusions of their upper body to effectively topple the beam, such extrusion proving a limb evolved ad-hoc for the task of object manipulation. With the indirect encoding, an outstanding individual jumped high onto the pegs to hit the beam and cause it to slide. `BridgeWalker`, on the other side, saw a greater variety of solutions, being less "constrained" than the other tasks. We showcase a sample of the evolved morphologies in Figure 5.15 for the direct encoding and Figure 5.16 for the indirect one. As observed in (Cheney et al., 2013) and (Ferigo et al., 2022b), the indirect encoding evolves much more regular material patterns and shapes than the direct one.

While the visual inspection and the fitness values achieved by the best individuals support the claim that our co-optimization was indeed successful, the trend of the curves of Figure 5.14 might be interpreted as a poor convergence or, from an-


(a) `BridgeWalker`


(b) `BeamToppler`

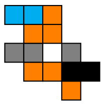
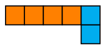
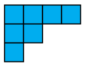

(c) `CaveCrawler`

Figure 5.15: Sample morphologies evolved over three tasks with direct encoding.


(a) `BridgeWalker`


(b) `BeamToppler`

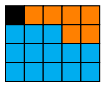

(c) `CaveCrawler`

Figure 5.16: Sample morphologies evolved over three tasks with indirect encoding.

other point of view, as an ineffective evolution. However, we remark that the inner optimization loop with RL allows sampling a neighborhood in the phenotype space, greatly increasing the likelihood of discovering an effective solution during the lifetime of an individual, to the point that even initial generations can achieve decent fitness.

To answer our main research question, in the next sections we analyze more closely the learning ability of the evolved agents, both qualitatively and quantitatively. From the qualitative point of view, we plot the learning curves, showing how the reward achieved by one individual changes over the iterations performed by PPO, and compare them at different generations. From the quantitative point of view, we measure the *learning radius*, i.e., the difference between the reward at the end of learning and the reward at the beginning, to get an estimate of the radius of the epigenetic space and, intuitively, a measure of how much a given morphology allows to learn. Finally, to get more insights and to attempt to explain our findings, we analyze systematically the morphologies being evolved, by computing a few morphological descriptors and plotting them throughout evolution.

5.2.5.6 Qualitative analysis: learning curves

To have an overview of how individuals learn at different stages of evolution, we proceeded as follows. For each task and each evolutionary run, we took the entire population at four evolution stages: at the beginning, i.e., just after initialization,

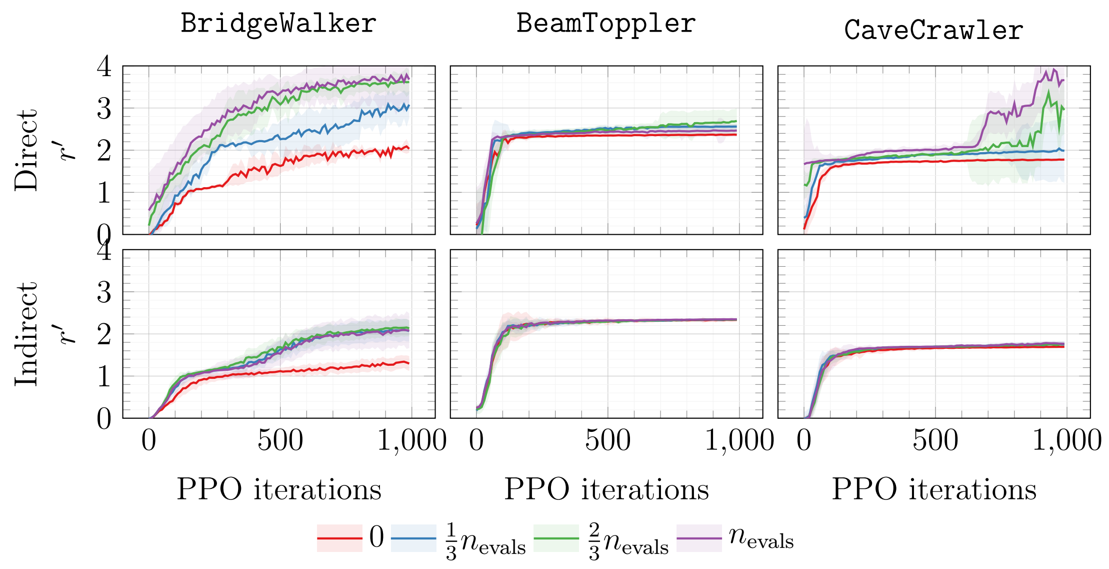


Figure 5.17: Moving average $r^{(k)}$ as a function of PPO iterations, taken at different generations, for three tasks and two encodings. Median $\pm$ standard deviation across evolutionary runs. Speed of learning increases for the direct encoding, while the same is not always true for the indirect.

at $\frac{1}{3}n_{\text{evals}}$, at $\frac{2}{3}n_{\text{evals}}$, and the end of the evolution. Then, for each individual in the population, we instrumented the PPO learning procedure to compute the average reward $\bar{r}^{(j)} = \frac{1}{n_{\text{sim}}}\sum_{k=1}^{n_{\text{sim}}} r^{(k)}$ obtained in one "off-line" simulation performed with the policy available after each j-th PPO iteration; intuitively, hence, we computed the "fitness" during the learning, instead of just at the end of the learning. Finally, we took a moving average in a window of 10 PPO iterations, obtaining $\bar{r}'^{(j)} = \frac{1}{10}\sum_{i=0}^{9} \bar{r}^{(j+i)}$, and computed the median $\bar{r}'^{(j)}$ across all the individuals in the population. Figure 5.17 shows the curves of r' vs. the PPO iteration, briefly *learning curves*, at different evolution stages (line color), for the different tasks (columns of plots), and different morphology encodings (rows of plots).

By observing Figure 5.17, it can be seen that the learning curves differ among tasks, encodings, and evolution stages. While for the `BridgeWalker` task r' clearly increases more steeply as evolution progresses—evolution does matter—for both encodings, probably due to the task being more basic, the same is not true for the other tasks. For `BeamToppler`, learning succeeds in converging very quickly for both encodings and at every generation; nevertheless, learning curves flatten towards the same local minimum for the indirect encoding, whereas there is a positive trend for the direct encoding. In `CaveCrawler`, the indirect encoding shows the same trend as in `BeamToppler`. The direct encoding, on the other side, succeeds in converging faster as evolution progresses: more in detail, the initial r', i.e., the one at the beginning of the PPO learning, increases over the evolution; moreover, only at the latest stages of the evolution, the learning appears to be able to escape the local minimum corresponding to the value of 2 for r'. Interestingly, for `BeamToppler` and the direct encoding the evolution ends up (violet line) favoring agents that learn a lot at the beginning of the learning and stop learning after ≈ 100 PPO iterations; in a previous stage, namely at $\frac{2}{3}n_\text{evals}$ fitness evaluations (green line), the learning is slower at the beginning but then slowly continues for all the 1000 PPO iterations.

In summary, the qualitative analysis of the learning curves shows that (a) the two encodings do differ in how they impact the ability to learn of the agents and, in particular, that (b) with the direct encoding, the evolution increases the ability to learn in two on three cases.

5.2.5.7 Quantitative analysis: learning radius

While one aspect of phenotypic plasticity is how a phenotype travels across the epigenetic space, another is the radius (or, more generally, the size) of such space: arguably, one can travel very fast if the radius of the space to be covered is very small, and vice versa.

We compute the radius ρ as the difference $r'^{(n_\text{RL-iters})} - r'^{(0)}$ of the reward r' at the end of learning and the beginning of learning; we measured ρ at the end of the evolution. While a more sophisticated measure might be found in the behavior space, we believe our notion of reward effectively captures enough about an agent's phenotype and we leave other measures as future work. Figure 5.18 visualizes the results as the distribution of ρ across evolutionary runs for the three tasks and the

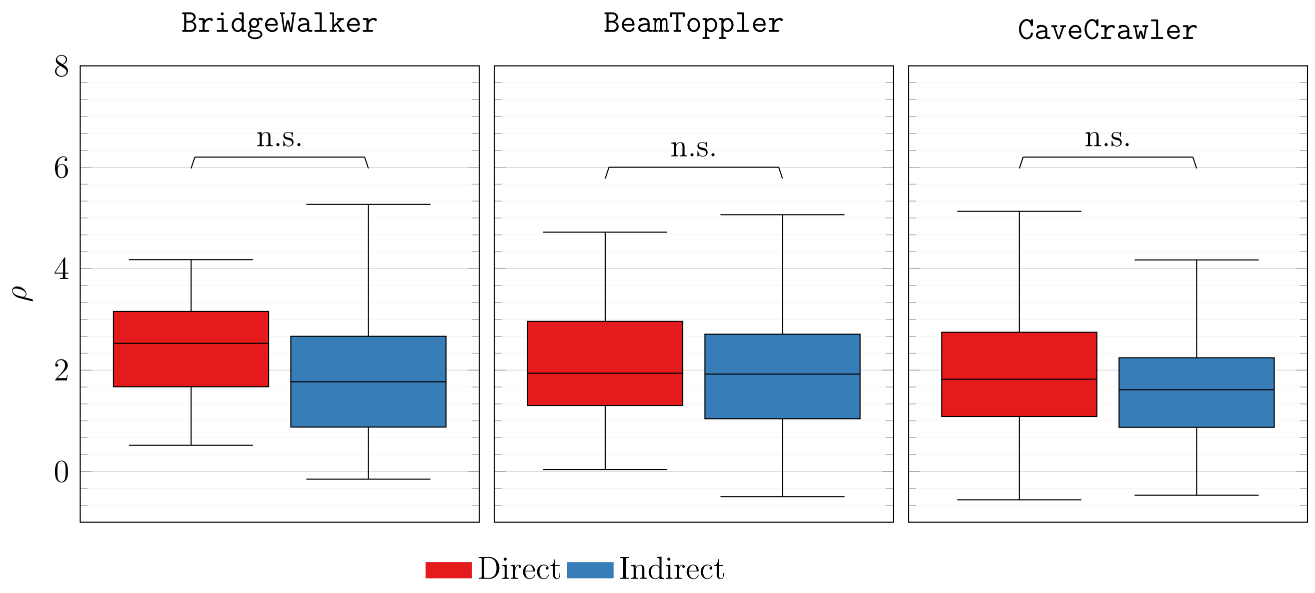


Figure 5.18: Distribution of learning radius ρ across evolutionary runs. There is no significant difference between the two encodings.
two encodings. Above each pair of boxplots, the brackets report the p-value for the statistical test against the null hypothesis of equality between the medians.

Indeed, there is no clear difference in Figure 5.18 between each pair of boxplots and the p-values are not significant for all the tasks. We can thus argue that the size of the epigenetic space is mostly the same across the two encodings and that the higher speed of learning observed for the direct encoding is not the result of shorter trajectories for the agents.

5.2.5.8 Morphological descriptors

To corroborate our analysis, we quantitatively investigate how morphologies change over the course of evolution by visualizing a set of *morphological descriptors*. For each morphology, we consider the fraction of non-empty voxels d_{size}, the fraction of rigid inactive voxels d_{rigid}, the fraction of soft inactive voxels d_{soft}, the fraction of horizontal actuators d_{h}, and the fraction of vertical actuators d_{v}. Then, we compute the average of each descriptor across all the individuals in the population and look at how the values change during the evolution. We summarize the results in Figure 5.19,

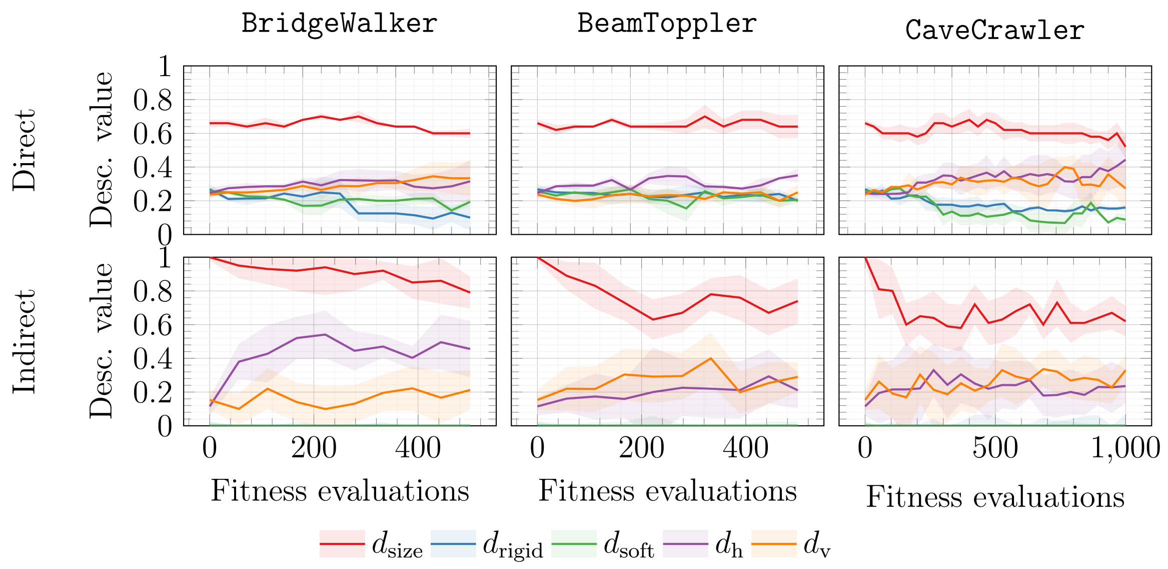


Figure 5.19: Median ± standard deviation across evolutionary runs for the morphological descriptors (fraction of non-empty voxels, rigid material voxels, soft material voxels, horizontal actuators, vertical actuators) of the population over the course of evolution. Paths of morphological descriptors are mostly monotonous for direct encoding but erratic for indirect encoding.
in terms of median ± standard deviation across evolutionary runs.

From Figure 5.19, we find that the two encodings radically differ in the trend of the descriptors. Morphological descriptors for direct encoding mostly follow either decreasing or increasing paths, signaling there is a tendency towards some morphological traits. Morphological descriptors for the indirect encoding, on the other side, show erratic and less clear paths and exhibit, in general, a larger variability (y-axis extent of shaded area in the plots). Albeit exhibiting some trends over the long run, most of the descriptors oscillate between increasing or decreasing.

We look at Figure 5.19 also to appreciate how different tasks differ in terms of selective pressure on the morphologies. For example, morphologies evolve to be smaller and more "muscular" (i.e., with less inactive material) for the `CaveCrawler` task, where every excess voxel might turn out a hindrance when squeezing through tight spaces. In the `BeamToppler` task, morphologies evolve to have more vertical actuators, as the task requires stretching vertically to topple the beam.

Moreover, Figure 5.19 also suggests that the two morphological encodings induce different biases. With the direct encoding the evolution starts with an even distribution of materials across voxels and then slowly favors materials which make morphologies more adapted to the task. With the indirect encoding, the morphologies tend to be larger and, in general, composed mostly of active materials already at the beginning of the evolution (similar observations have been done by Ferigo et al. (2022b) with other indirect encodings): this can be explained by the very nature of the encoding itself, for which a uniform distribution in the CPPN space does not necessarily correspond to a uniform distribution in the morphology space. Clearly, big and strong (since composed of many active voxels) morphologies are favored and tend to fill the entire population starting from the beginning of the evolution. However, they appear to be harder to control and, hence, make the learning less efficient, as visible in Figure 5.17 and previously discussed.

5.2.6 Discussion

The above results suggest that when co-optimizing virtual agents by evolution and learning, the degree of plasticity of the learned controllers differs across our direct and indirect encodings. In particular, we find that, in the case of direct encodings, effective controllers emerge more and more quickly and morphologies converge. In other words, morphologies "canalize" (Waddington, 1942) towards designs that allow for learning to take place more swiftly (Waddington, 2014). Recalling Figure 5.10, we interpret this canalization as evolution that selects for morphologies that allow for phenotypes to "roll" over canals in the epigenetic space (Waddington, 2014).

We also find the indirect encoding to be less suited for evolving agents with increased plasticity. We attribute the reason to be the different *locality* (Rothlauf, 2003), one of the properties of genetic encodings (Rothlauf, 2006), between the direct and the indirect encoding. We say an encoding is "local" if it preserves the distances between individuals when mapping from genotypes to phenotypes. In other words, a local encoding maps individuals with close genotypes (in the genotype space) to close phenotypes (in the phenotype space) and individuals with distant genotypes (in the genotype space) to distant phenotypes (in the phenotype space). It is likely that our direct encoding enjoys more locality than the indirect, where the genotype-phenotype mapping is non-trivial. Moreover, indirect encodings generally have less *heritability* (De Carlo et al., 2021), with fit parents begetting poor offspring (and vice-versa) more frequently than with a direct encoding. Altogether, these facts mean that, with an indirect encoding, it is more difficult for evolution to select morphologies that lead to canalization. That is exactly what Mayley (1996) argued: for canalization through the Baldwin effect to take place, a small distance between two individuals in the phenotype space must imply a small distance in the genotype space.

5.2.7 Conclusion

The co-optimization of virtual agents, by the evolution of morphologies and the learning of controllers, despite promising, poses several challenges. One of these is the choice of the genetic encoding for the morphology. However, no work to date has ever delved into how that affects phenotypic plasticity.

That is precisely what we set out to answer with this work. We evolve the morphology of VSRs—whose many degrees of freedom make them ideal testbeds—with EAs and, for each morphology, learn a controller through RL. For the EA, we test two encodings, direct and indirect. We experiment with three tasks from the Evolution Gym (Bhatia et al., 2021) benchmark, consisting of bridge walking, beam toppling, and cave crawling.

Our results show that the two encodings differ. The direct encoding results in increased learning ability and selection of morphological traits that reduce the cost of learning, thus witnessing a Baldwin effect. The indirect encoding does not always result in increased learning ability and sustains high levels of morphological diversity, thus it is difficult to argue in favor of a Baldwin effect. We conjecture the reason to be the different properties of the encodings, in particular, the lower locality (Rothlauf,

2003) and lower heritability (De Carlo et al., 2021) of the indirect one: since similar genotypes do not always correspond to similar phenotypes, it is more difficult for evolution to select for morphological traits that reduce the cost of learning.

We believe our work delivers key insights into the choice of morphology encoding, which is crucial for the co-optimization of agents with evolution and learning. Still, we acknowledge there are limitations to our work that we will address in future work. For the sake of generality, we will experiment with more variants for both the direct and the indirect encoding and, possibly, for learning.

5.3 Evolving Hebbian learning rules in VSRs

In this chapter, we answer to question **5**.

5.3.1 Introduction

It is not unlikely that future robotic agents would undergo the same stages of life (birth, maturity, death, or disposal) of biological agents (Eiben et al., 2013). As such, learning and adaptation should inherently guide "infant" robots towards maturity (Eiben and Hart, 2020a), just like in animals. Indeed, learning has been acknowledged as one of the main challenges on the road towards fully autonomous robotic ecosystems (Eiben, 2021b).

For robotic societies to become a reality, adaptation must take place across all three timescales: *phylogenetic* (evolution), *ontogenetic* (development), and *epigenetic* (learning) (Sipper et al., 1997). As of now, the vast majority of robotic systems are either evolved (Nygaard et al., 2018) or learned (Ha et al., 2021). While some works at the intersection of evolution and learning do exist (Hallawa et al., 2021), they mostly rely on a human-assigned reward signal that must be decided by a "supervisor", and thus might not be adequate for an autonomous robotic society. On the other side, biological agents share both an "innate" evolved component and a learned component. A robot that only evolves, albeit as powerful as evolution might be, can in principle beget offspring that are not viable. As a matter of example, when jointly evolving brain and body, the mismatch can happen between the two (Pagliuca and Nolfi, 2020). A robot that only learns, without an innate evolved "instinct", since learning usually relies on trial-and-error, incurs the risk of damaging itself and the entities surrounding it (Grbic and Risi, 2021). For instance, an autonomous vehicle can provoke serious damage if it optimizes a policy by sampling random actions. Then, there is a need for robotic systems that adapt on more than one timescale, quelling the longstanding "nature vs. nurture" debate (Moore, 2003).

Among the shapes learning can take, we are concerned with plasticity (Zilles, 1992), since it is "unsupervised" in the sense that it does not require a human-assigned reward signal. Additionally, plasticity is known to play a key role in the learning processes of biological agents (Patten et al., 2015) and it is a simple yet effective model. Hebbian theory (Hebb, 2005), which states that "neurons that fire together, wire together", enshrines it. Synapses between neurons are plastic, i.e., they adapt their strength in response to stimuli, following synapse-specific rules known as "Hebbian rules". These rules are in turn expressed through well-defined parameters, according to the model employed (Soltoggio et al., 2018).

We apply Hebbian learning to an ANN controlling the VSR. We embed Hebbian learning inside an EA that optimizes the parameters of the Hebbian rules; in this sense, there are two scales of adaptation: evolution and learning, the former being the longer and slower one. To the best of our knowledge, this is the first work on applying Hebbian learning to VSRs.

We evolve Hebbian learning rules for two fixed VSR morphologies in a locomotion task on hilly terrains, and re-assess the VSRs performance on unseen terrains, in order to quantify their ability to generalize. With experiments, we:

(a) confirm that the Hebbian controller is never worse (and often better) than a non-Hebbian baseline, and truly shines when adapting to unforeseen damages in its body;

(b) verify that Hebbian controllers indeed learn, i.e., improve over the lifetime and can generalize;

(c) unveil some of the weight dynamics underlying learning.

This work falls in the more general study of the link between learning and evolution, which is rooted in both biology and artificial intelligence, with fundamental consequences on robotics. In this regard, a recent position paper (Eiben and Hart, 2020a) argued that evolutionary robot systems should always "contain a learning component where a newborn robot refines its inherited controller to align with its body, which will inevitably be different from its parents". The authors summarized this concept with the motto: "If it evolves it needs to learn". As already pointed out in (Pontes-Filho and Nichele, 2019), this paradigm can be shifted though, as we do here, where we show that the learning-evolution link is actually bidirectional. In our experiments, not only the evolved agents need to have a learning phase, but also the learning strategy (i.e., the parameters of the Hebbian rules) needs to evolve over the course of the generations. In other words, learning emerges through evolution. We can summarize this concept by paraphrasing the aforestated motto as follows: *if it evolves it needs to learn, but if it learns, it needs to evolve.*

5.3.2 Related works

Neural plasticity (Zilles, 1992), a form of learning, plays a key role in the development of biological neural networks. Intuitively, it is crucial in adapting to changes in environmental conditions, as well as in shaping memories (Patten et al., 2015). Indeed, starting from (Baldwin, 2018), there has been growing evidence about the Baldwin effect, i.e., an acceleration of evolution when learning happens during a lifetime. In a seminal computational study, Hinton et al. (1996) showed that learning can provide a gradient for evolution to follow even on an extremely deceptive fitness landscape.

Following (Shaw et al., 2001), we distinguish between structural and functional plasticity. The former refers to the ability of the nervous system to rewire its neural connections, creating new pathways among neurons and undoing existing ones. The latter refers to the ability of the nervous system to alter over time the functional properties of neurons. In this work, we are concerned with functional plasticity and, in particular, with changes in synaptic strength in response to previous activity (activity-dependent plasticity) (Ganguly and Poo, 2013). Previous activities can induce persistent strengthening of synapses (long-term potentiation) (Cooke and Bliss, 2006) or persistent weakening (long-term depression) (Massey and Bashir, 2007).

As a result of the well-known benefits of plasticity, there has been a considerable amount of literature devoted to engineering "plastic" ANNs. Schmidhuber (1993b) first proposed "fast weights", where a slow-learning ANN learns the weights of a fast-learning ANN, as a thought experiment. In the last decade, new studies have taken inspiration from fast weights, including adaptive HyperNEAT (Risi and Stanley, 2010), fast weights for recurrent neural networks (Ba et al., 2016), and hypernetworks (Ha et al., 2017). Moreover, there is a growing amount of research on the attention mechanism (Bahdanau et al., 2014; Pigozzi et al., 2022), which can be seen as a form of "adaptive weights", and synaptic pruning of ANNs (Hoefler et al., 2021), which can be seen as a very "sharp" form of structural plasticity.

However creative and ground-breaking these works might have been, they are still very complex to engineer. Hebbian learning (Soltoggio et al., 2018) provides a more succinct, yet biologically-plausible representation. According to Hebbian theory (Hebb, 2005), if a pre-synaptic neuron often stimulates the activation of a post-synaptic one, then their synapse increases in strength[2]. Past studies (Nolfi and Parisi, 1996; Floreano and Urzelai, 2001) applied Hebbian learning to evolve ANN controllers

[2]This statement should be read with a pinch of salt: if two neurons activate at the same time, then there cannot be causation between the two activations. For causation to subsist, it must happen that one neuron took part in activating the other (Caporale and Dan, 2008). In this work, we consider connectionist ANNs, where there is no need to take into account the temporal dimension.

for mobile robots and found improved performance over non-plastic ANNs. More recently, several studies have successfully evolved Hebbian learning rules (Miconi et al., 2018; Najarro and Risi, 2020; Yaman et al., 2021; Jordan et al., 2021) and achieved competitive results in reinforcement learning scenarios. For these reasons, we adopt Hebbian learning to model synaptic plasticity

5.3.3 Methods

We present the methods for this work.

5.3.3.1 Hebbian learning

As discussed above, Hebbian learning provides a way to optimize an ANN while performing a given task. In an ANN, the weights play the role of the synapses, as each of them modulates the connection between any pair of pre-synaptic and post-synaptic neurons. From a computational point of view (Brown et al., 1990), the general formulation of Hebbian learning updates the weights according to:

$$w_{ij}^{(k+1)} = w_{ij}^{(k)} + \Delta w_{ij}^{(k)} \tag{5.13}$$

$$\Delta w_{ij}^{(k)} = \eta x_i^{(k)} y_j^{(k)} \tag{5.14}$$

where $x_i^{(k)}$ and $y_j^{(k)}$ are the activation of the pre-synaptic and post-synaptic neurons, respectively, at time step k, and η is the learning rate. In synthesis, Equation (5.14) dictates to strengthen the synapse value if $x_i^{(k)}$ and $y_j^{(k)}$ are positively correlated, weaken it if they are negatively correlated, and keep it constant if at least one of the two is zero. This model takes inspiration from biological systems, in which there is evidence that plasticity in the frontal-striatal synapses arises from changes in the concentration of dopamine, which in turn is a function of the difference between observed and expected outcomes (Averbeck and Costa, 2017; Neftci and Averbeck, 2019).

While several works (Coleman and Blair, 2012; Floreano and Urzelai, 2000; Yaman et al., 2021) have successfully employed this "generalized" Hebbian learning to train ANNs, many variations of it do exist (Soltoggio et al., 2018). In this work, we use the so-called Hebbian *ABCD* model (Mattiussi and Floreano, 2007; Niv et al., 2001; Soltoggio et al., 2007; Najarro and Risi, 2020), which updates the weights according to:

$$\Delta w_{ij}^{(k)} = \eta \left(A x_i^{(k)} y_j^{(k)} + B x_i^{(k)} + C y_j^{(k)} + D \right) \tag{5.15}$$

where $A, B, C, D \in \mathbb{R}$ are the eponymous coefficients and $\eta \in \mathbb{R}$ is the learning rate. The four coefficients determine the local weight dynamics, with A modulating the relation between the two signals, B and C modulating the pre-synaptic and post-synaptic values, respectively, and D acting as a bias specific to the synapse.

We call the four *ABCD* coefficients together a *rule*, and, in our model, there exists one separate rule per synapse. In line with uniform plasticity (Schmidhuber, 1993a), all the rules share the same learning rate η (we digress on the non-uniform case in Section 5.3.4.4); after preliminary experiments, we set $\eta = 0.01$. Moreover, we initialize $w_{ij}^{(0)} = 0$ for every i, j, i.e., at the beginning of the life of the VSR, every weight is set to 0. The parameters $\boldsymbol{\theta}$ that we optimize consist then in the concatenation of the *ABCD* coefficients for all the rules. The Hebbian controller has thus four times the number of free parameters of an MLP with the same architecture.

5.3.3.2 Evolutionary algorithm

We resort to EC for optimization, and, in particular, adopt the ES (Beyer and Schwefel, 2002) described in Section 3.3.5.2 as our EA. Indeed, EAs have proven competitive for reinforcement learning problems (Such et al., 2017b); in particular, ES have achieved state-of-the-art results for continuous control tasks (Salimans et al., 2017). After preliminary trials, we set $n_{\text{pop}} = 40$, $n_{\text{gen}} = 500$ (corresponding to 20 000 fitness evaluations), and $\sigma_{\text{mut}} = 0.35$.

5.3.4 Experimental analysis

We performed several experiments aimed at answering the following research questions:

RQ1 Is evolution with Hebbian learning effective? That is, are VSRs equipped with the Hebbian controller effective in the task of locomotion? Are they able to generalize to unseen conditions as new environments or malfunctions in the morphology?

RQ2 Is there any "true" learning?

RQ3 Why Hebbian learning works?

We anticipate that by "true" learning, here we mean that the agent retains the ability to accomplish the task even once Hebbian plasticity is disabled, i.e., the weights are frozen (this will become clearer below).

To answer these questions, we experimented with two different VSR shapes, see Figure 5.20. In particular, we considered a 4×3 (size of the grid enclosing the voxels) rectangle with a 2×1 rectangle of missing voxels at the bottom-center, that we call *biped*, and a 7×1 rectangle, that we call *worm*. For each shape, we considered three sensory apparatuses differing in the number and kind of sensors they are composed of, hence in their complexity. We equipped the *high*-complexity apparatus as follows: area sensors for all the voxels, touch sensors for the voxels in the bottom row of the shape, velocity sensors for the voxels in the top row of the shape, and lidar sensors for the voxels in the rightmost column of the shape. The *medium*-complexity apparatus is equivalent to the *high* apparatus, with the exception of the lidar sensors, that are absent. Finally, we equipped the *low*-complexity apparatus with just the area sensors for the two top rows in the case of the biped shape, and the three central voxels in the case of the worm shape. For the three apparatuses respectively, the input dimension is 48, 36, and 8 for the biped shape, 31, 28, and 3 for the worm shape.

We experimented with different shapes and sensory apparatuses to get a sense of the effectiveness of Hebbian learning across a wide array of morphological conditions.

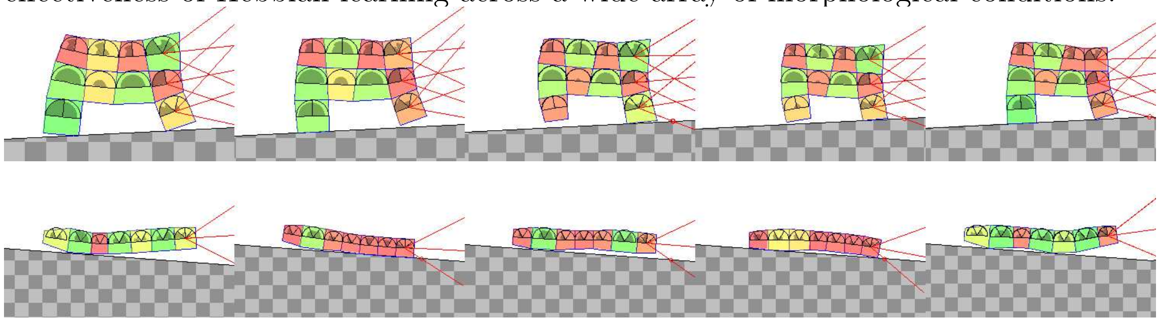


Figure 5.20: Frames of the simulation of the two shapes during locomotion on hilly terrain.

For all the experiments, we considered the task of locomotion. The goal is to travel on a terrain as fast as possible along the positive x direction, in a fixed amount of simulated time t_{final}. The fitness of a VSR is given by its average velocity $\overline{v}_x$, computed considering the x-position of the VSR center of mass at the beginning and the end of the simulation. We set $t_{\text{final}} = 60\,\text{s}$. Here, we consider the hilly terrain of Sections 4.1 and 4.2. It consists of a sequence of bumps, having an average height of 3 m and an average distance of 30 m. Additionally, the random seed for procedurally generating the bumps is different at every fitness evaluation, to prevent individuals from "overfitting" to one single terrain profile, making adaptation more challenging.

We implemented the experimental setup in Java, building on top of two frameworks: JGEA[3] for the evolutionary optimization and 2D-VSR-Sim[4] (Medvet et al., 2020b) for the simulation of VSRs. For the simulator, we set the time step to $\Delta t = 1/60\,\text{s}$ and the other parameters to the default values (as a result, all the voxels share the same mechanical properties). The code to reproduce the experiments is publicly available at `https://github.com/ndr09/VSRevo`.

For each experiment, we performed 10 evolutionary runs with different random

[3]`https://github.com/ericmedvet/jgea`
[4]`https://github.com/ericmedvet/2dhmsr`

seeds for the EA. We remark that, for a given VSR and terrain, the simulations are deterministic. After verifying the adequate hypotheses, we carried out statistical tests with the two-sided Mann-Whitney U rank test (Mann and Whitney, 1947) for independent samples using $\alpha = 0.05$ as the confidence level.

5.3.4.1 RQ1: effectiveness of evolution with Hebbian learning

As a starting point, we are interested in investigating the effectiveness of the Hebbian controller. We measure effectiveness in two different cases: in the same conditions as evolution, and on a re-assessment procedure with slightly different conditions. In both cases, the performance index is $\overline{v}_x$, which, in the first case, is the fitness function itself. The second case aims to test the generalization abilities of an evolved individual. To this end, we compute $\overline{v}_x$ across 10 unseen hilly terrains, obtained with 10 different predefined random seeds.

We compare the Hebbian model against a baseline model. In this work, we used as a baseline a "vanilla" MLP controller, with the same architecture as the Hebbian controller (see Section 5.3.3.1). The baseline controller is different from the Hebbian controller in two ways. First, the weights of the MLP of the former stay the same for the entire simulation, while in the latter they change at every time step according with Equation (5.15). Second, in the baseline controller, we optimize the weights, differently than in the Hebbian controller, where we optimize the $ABCD$ parameters. For the optimization, we use the same EA of Section 5.3.3.2. As for the Hebbian controller, we couple the baseline controller with the two shapes and the three sensory apparatuses previously outlined.

We summarize the results in Figure 5.21, which shows $\overline{v}_x$ of the best individuals at the end of evolution, and Figure 5.22, which shows $\overline{v}_x$ of the best individuals on the re-assessment terrains. For the same shape and sensory apparatus, we also report the p-value for the statistical test against the null hypothesis of equality between the medians. Figure 5.23 plots $\overline{v}_x$ in terms of median $\pm$ std. dev. over the course of

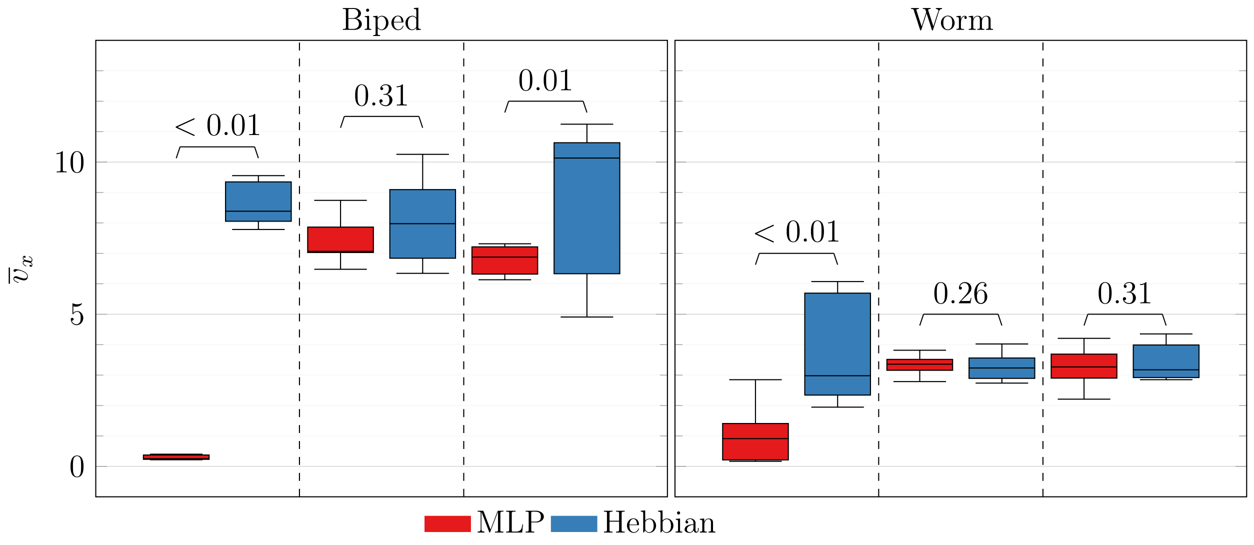


Figure 5.21: Distribution of the velocity $\overline{v}_x$ of the best individuals found in each of the 10 evolutionary runs, obtained with the two controller types on each combination of shape (the two plots) and sensory apparatus (the three columns). Hebbian learning is never worse than the baseline.
evolution.

From the figures, we find that Hebbian learning is never worse than the baseline: in fact, it is either comparable or better. In particular, if we look at the results shown in Figure 5.21, it turns out that the Hebbian controller consistently outperforms the MLP for the *low* sensory apparatus in both shapes and the *high* sensory apparatus in the biped shape, which are the configurations whose p-values are significant. Also, the performance gap seems to be particularly abysmal in the *low* sensory apparatus; we speculate the reason to be that Hebbian learning is particularly effective with "scarce" perceptual conditions. When comparing the two shapes, we find instead that the advantage of Hebbian learning over the MLP is more evident with the biped shape than with the worm. We hypothesize that this difference may be due to the fact that, having a much simpler shape, the worm can perform well already with a simple, "instinctive" controller (i.e., one that maps statically inputs to outputs, as the MLP), i.e., there is little margin for learning. On the contrary, the biped shape may benefit from learning since the controller space has an increased

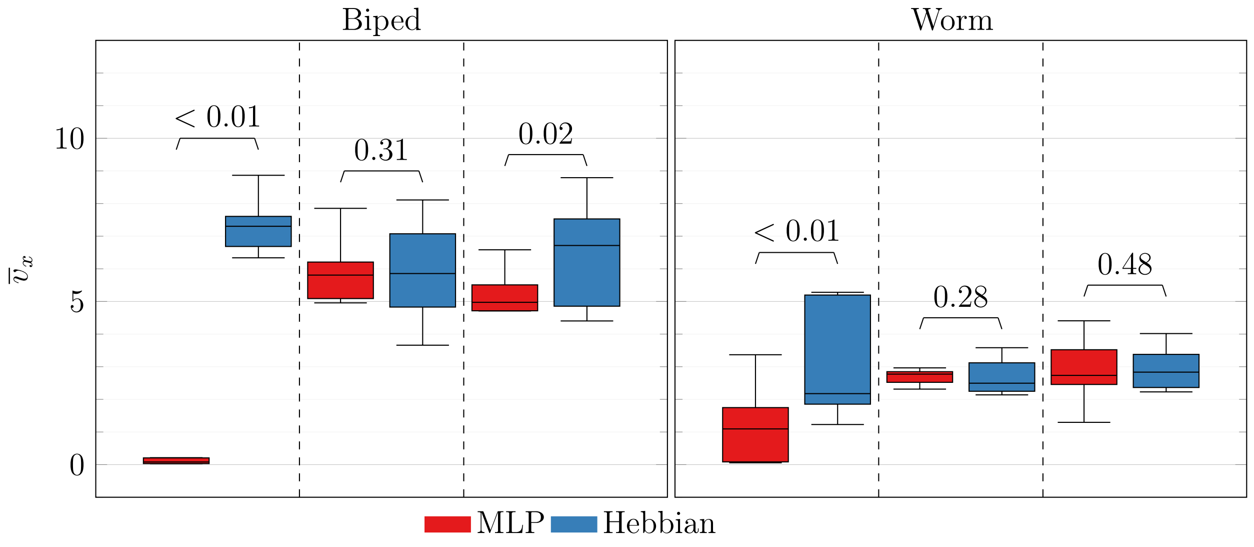


Figure 5.22: Distribution of the velocity $\overline{v}_x$ across 10 unseen hilly terrains for the best individuals found in each of the 10 evolutionary runs, obtained with the two controller types on each combination of shape (the two plots) and sensory apparatus (the three columns). Hebbian learning is never worse than the baseline.

complexity reflecting the higher complexity of the body. This somehow matches the previous observation on the increased effectiveness of Hebbian learning with simpler sensor apparatuses: with less sensors, learning can provide a better way to exploit the available perceptual information, while with more sensors an instinctive controller is enough. The results on the re-assessment, shown in Figure 5.22, corroborate and mirror these effects, indicating that the evolved Hebbian controllers also have generalization abilities. Moreover, Figure 5.23 confirms that, despite having four times the number of parameters of the MLP, the Hebbian controller succeeds in converging to a plateau in the fitness landscape.

We visually inspected the robot with the evolved Hebbian controllers and found their behaviors to be highly adapted for a locomotion task on uneven terrain; bipeds hop on their legs and worms inch forward as real caterpillars do. Visual inspection is fundamental since behaviors are a strong indicator of a possible reality gap (van Diggelen et al., 2021; Salvato et al., 2021). We made sample videos available at: `https://youtu.be/bZ2Ek9ohzXI`, `https://youtu.be/5tCaQTRXRp8`, and `https://`

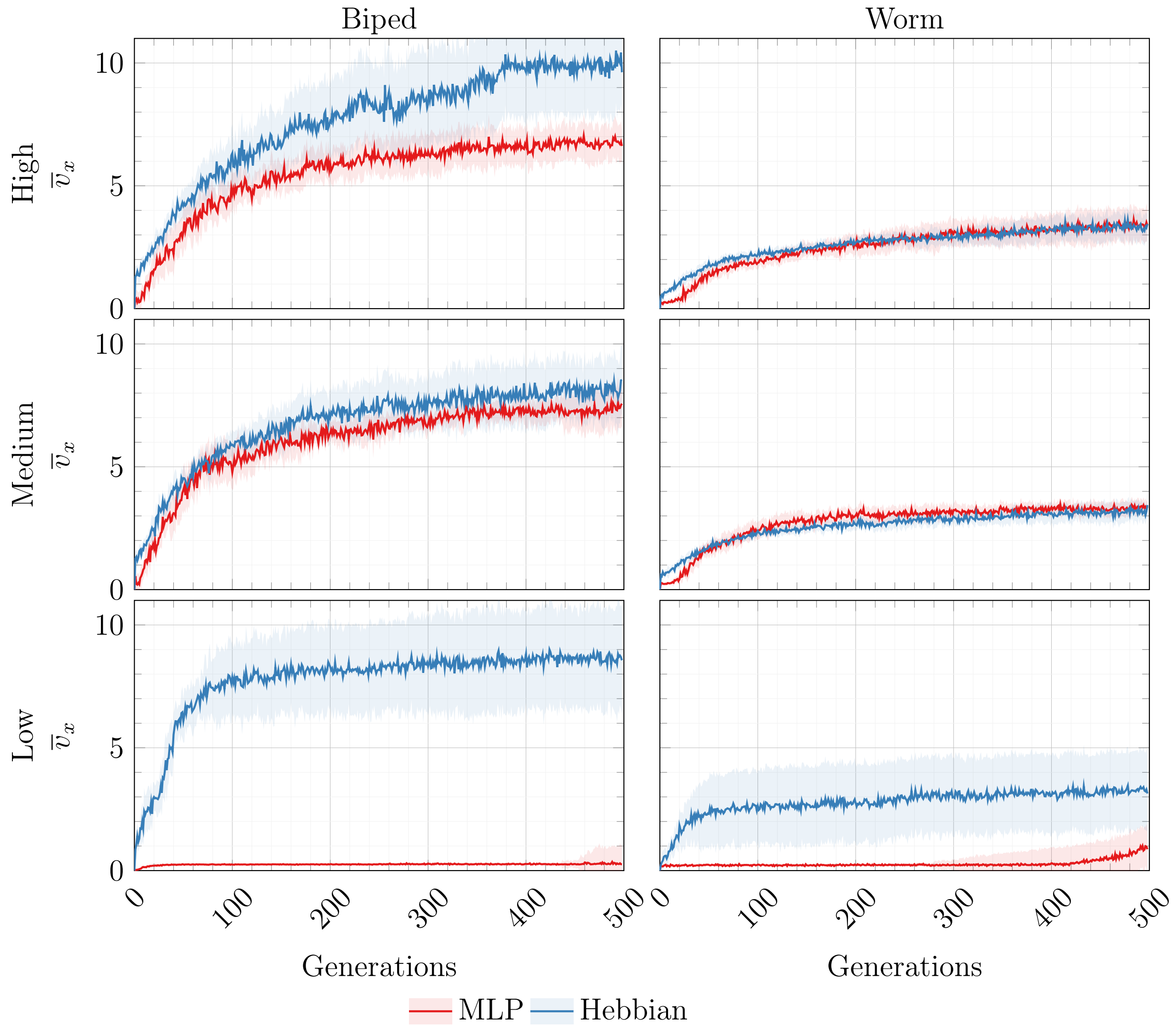


Figure 5.23: Median ± std. dev. (solid line and shaded area) of the velocity $\overline{v}_x$ of the best individuals found during each of the 10 evolutionary runs, obtained with the two controller types on each combination of shape and sensory apparatus. Hebbian learning converges as fast as the baseline even if it has four times the number of parameters.
`youtu.be/q0NrstiF9AQ`.

To further investigate the generalization abilities of the Hebbian controllers, we tested their ability to recover from unforeseen damages affecting the body of the VSR. In particular, we used the following protocol. After half of the simulation (30 s) has elapsed, the VSR experiences a trauma, with every voxel having a 0.5 probability of

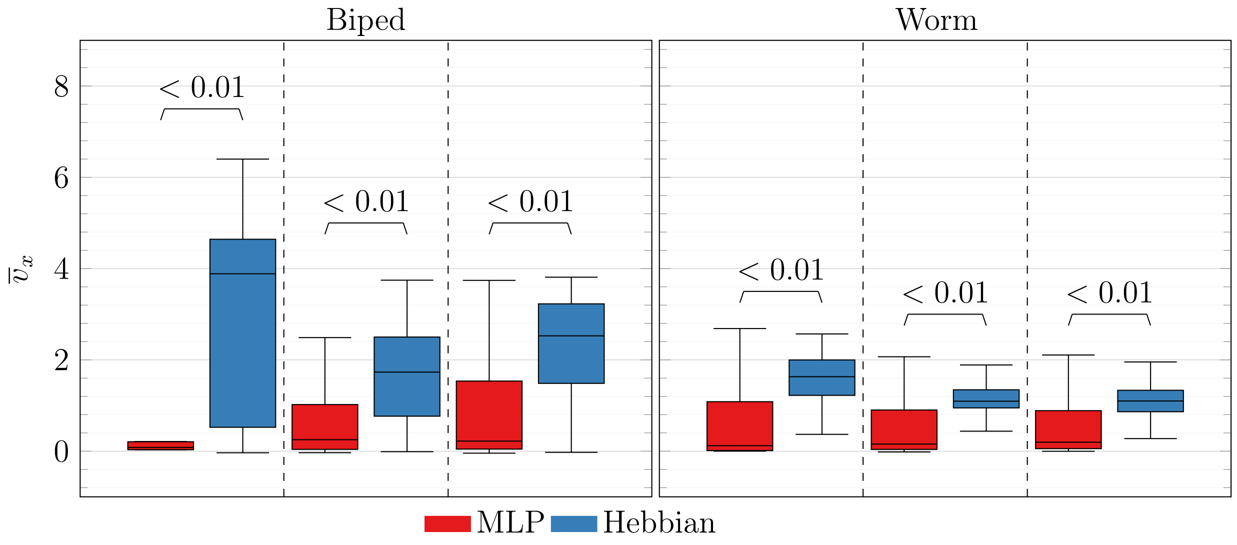


Figure 5.24: Distribution of the velocity $\overline{v}_x$ across 10 unseen hilly terrains for the best individuals found in each of the 10 evolutionary runs (with damages), obtained with the two controller types on each combination of shape (the two plots) and sensory apparatus (the three columns). Hebbian learning is decisively better than the baseline.

breaking—thus, every VSR has, on average, 50 % of the voxels broken for the second half of the simulation. Upon breakage, a voxel does not apply anymore the actuation signal it receives from the controller, and hence its area is determined only by external forces. We ran an experimental campaign of 10 evolutionary runs for the Hebbian and MLP controller types, and re-assessed the best individuals on 10 unseen hilly terrains, as already explained. Note that, during each simulation (either in evolution or re-assessment), the VSRs experience damages affecting different voxels—this makes the task of evolving a controller harder than without the damages. We found the results in these conditions to be not significantly different from those obtained without damages, and, for the sake of conciseness, we report just the re-assessment results in Figure 5.24.

From the figure, we conclude that the evolved Hebbian controller improves the systems (recovers), but it also does not make it worse, which would be a possibility if the adaptivity went in the wrong direction. Taken from this perspective, controllers with Hebbian plasticity would fit neatly into the design of an autonomous robotic ecosystem.

These findings confirm that Hebbian learning is an effective learning paradigm and that, when coupled with evolution, generalization to unseen conditions is where it can truly shine. Two questions then arise: is the Hebbian controller really learning? Furthermore, why does it work, i.e., what are the dynamics of the weights during the robot's lifetime? We set out to answer these questions in the following two subsections.

5.3.4.2 RQ2: is there any "true" learning?

We aim to verify whether, in our experiments, a VSR with Hebbian learning does indeed learn. Intuitively, we say that an individual has "learned" how to perform a task if (a) it has built some internal representation of the experience collected on the task while learning and (b) it is able to re-use this experience. That is, we assume that the outcome of learning is available even after the process of learning has ended. From this assumption, it follows that if we stop learning once enough experience has been collected, the individual should still be able to achieve the task; if, instead, we stop it too early, the individual should not be able to achieve the task. If, at some point in its life, an individual who is learning is already able to achieve the task and stop the learning, there could be two outcomes: either (a) the individual retains its ability to solve the task or (b) it loses its ability. In the latter case, we might conclude that what was happening inside the individual was not "learning" in the sense we described above but was instead some form of fast adaptation of the brain that was itself functional to the ability of the individual. Namely, it is so functional that, if you stop it, the individual loses its ability to solve the task. Stopping the learning and looking at what happens can hence be used to verify whether true learning is happening.

Based on these considerations, we performed the following experiment. Given a VSR with an evolved Hebbian controller, we take a snapshot of it, along with its weights, at every second during the simulation. For every such snapshot, we measure its average velocity $\overline{v}_x$ in a new simulation lasting 60 s over an unseen hilly terrain with Hebbian learning turned off (i.e., we fix the weights to the frozen values of the

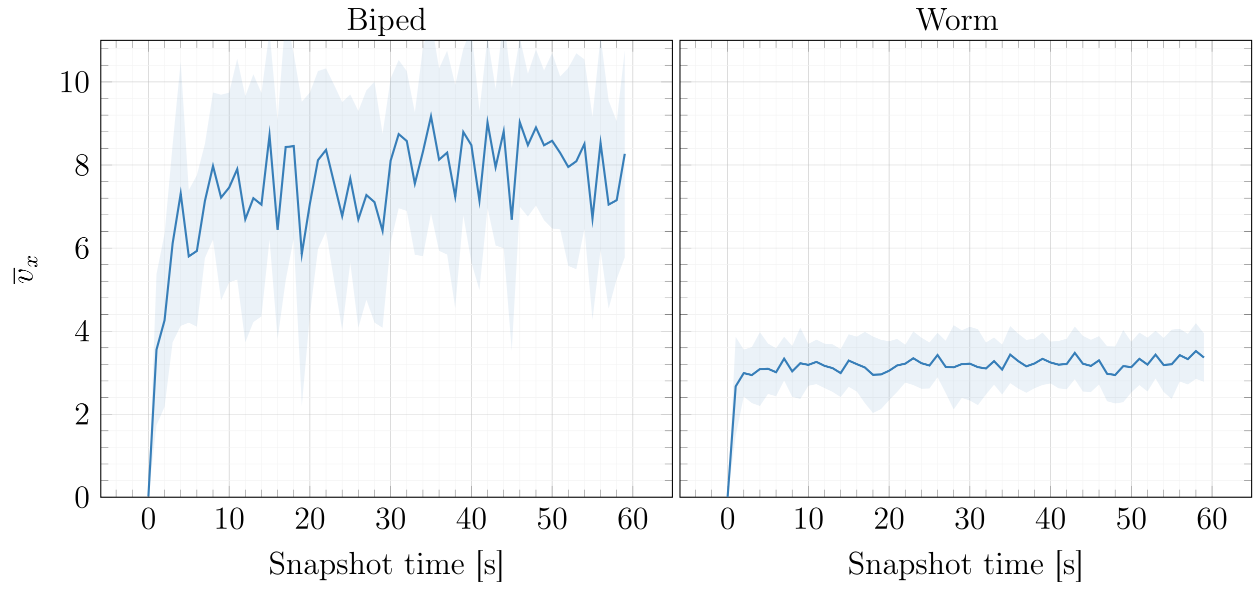


Figure 5.25: Median ± std. dev. (solid line and shaded area) of the velocity $\overline{v}_x$ of the best individuals found in each of the 10 evolutionary runs, with weights frozen at different time steps of the simulation (on x-axis, in s), obtained with the Hebbian controller on the two shapes and the *high* sensory apparatus. Hebbian learning requires just $\approx$ 10 s for the biped and $\approx$ 5 s for the worm to converge to a high-performing weight configuration.

snapshot and do not change them anymore during the simulation). We performed this procedure for the best individual (i.e., set of $ABCD$ parameters) of every evolutionary run. We report the results of such validation in Figure 5.25, in terms of median ± std. dev. across the 10 runs. On the x-axis, we report the time (in s) at which we took the corresponding snapshot. For the sake of conciseness, we report the results only for the *high* sensory apparatus since it is the configuration that delivered the best results for the Hebbian controller. The other cases are qualitatively similar.

From Figure 5.25, we observe that for both shapes the median $\overline{v}_x$ rises very steeply and then settles around a stable point. It takes about 10 s to 20 s before stabilizing for the biped shape, while for the worm it converges more rapidly in the first 5 s. This observation demonstrates that learning does indeed take place: after an initial settling period (which can be seen as the actual "learning" phase), the robots are capable of achieving high $\overline{v}_x$ even with the weights frozen—that is, they have acquired some experience through learning and they are able to exploit it to run effectively even after learning has stopped. We speculate that by the time $\overline{v}_x$ settles into a stable point, the weights have converged to a high-performing attractor in the weight space. These findings are in line with previous research on Hebbian learning for robotic agents, in particular, (Najarro and Risi, 2020).

5.3.4.3 RQ3: why Hebbian learning works

We want to understand why Hebbian learning works. In order to do so, we investigate the weight dynamics underlying locomotion and see whether they disclose some insights. Also in this case, for the sake of conciseness, we report the results only for the *high* sensory apparatus, since it performed the best in terms of effectiveness (see Section 5.3.4.1). The results are qualitatively the same for the other configurations.

Figure 5.26 plots, separately for two sample best individuals (one per shape), the histograms of the relative frequency of weights at different time steps of the simulation.

From the figure, one finding strikes us. By the end of the simulation, weights diverge to assume a bell-shaped distribution centered on their initial value of 0.0. Moreover, there are small clusters of values that accumulate on the boundaries and assume very large values. If we consider that learning takes about 10 s to happen (see Section 5.3.4.2), weights divergence is interesting since it persists over the entire simulation. We also visualized the neuron activations and found the corresponding histograms to be bi-modal, with two peaks at the boundaries (i.e., -1 and $+1$). Considering that we employ tanh as an activation function (whose output domain is indeed $[-1, +1]$), it is clear that activations become saturated. The divergence of the weights might hint that evolution is essentially performing a form of synaptic pruning (Hoefler et al., 2021), and implicitly optimizing not really the weights, but rather the topology of the neural networks. Figure 5.26 thus led us to believe that Hebbian plasticity creates some kind of underlying "Boolean" dynamics in the neural networks.

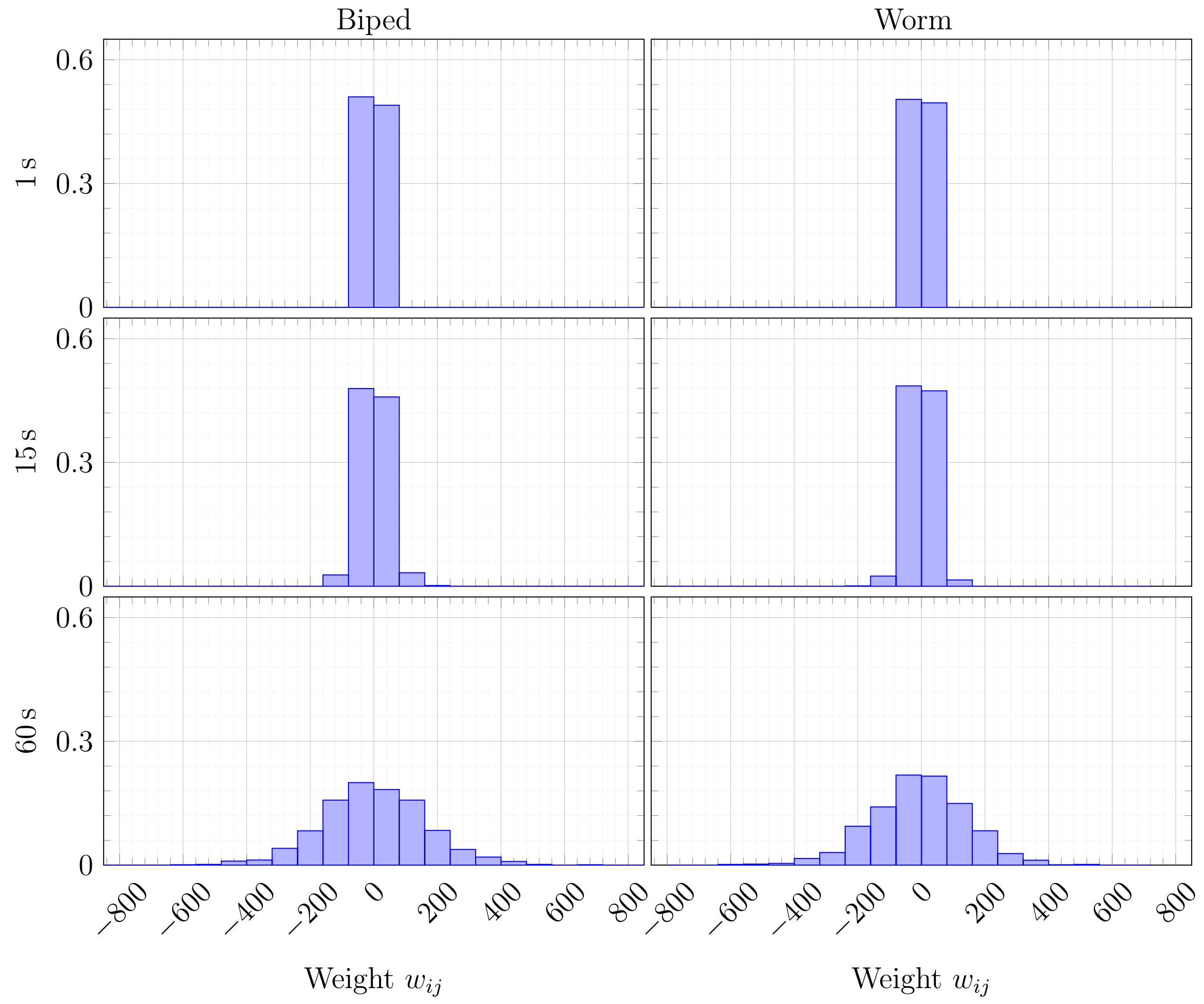


Figure 5.26: Weight distributions for two sample best individuals (one per shape, both with *high* sensory apparatus), at three different time steps of the simulation: 1 s, 15 s, and 60 s. In Hebbian learning, weights do diverge.

This fact bears similarity to what happens with weight agnostic neural networks (Gaier and Ha, 2019) and “bang-bang” controllers (Bellman et al., 1956), which are continuous-action controllers that, when optimized, degenerate to binary controllers. Inspecting the weights and activations of the other best individuals confirmed these results. Here, we plotted just two of them for the sake of conciseness.

To better understand the role that weight divergence plays, we introduce, in the form of an ablation study, a *normalized* Hebbian model. In this representation, we normalize weights to $[-1, +1]$ as $w_{ij}^{\prime(k+1)} = w_{ij}^{(k+1)} / \parallel \boldsymbol{w}_i^{(k+1)} \parallel_2$, where $\boldsymbol{w}_i^{(k+1)}$ is the vector of pre-synaptic weights for the post-synaptic i-th neuron. In other words, at each step for every neuron we divide its pre-synaptic weights by their Euclidean norm. We remark that, in the previous experiments, weights were unbounded. In doing so, we tap into the body of evidence on local synaptic competition between neurons in vivo (El-Boustani et al., 2018), according to which adjacent synapses modulate their strength so as to specialize and not be subdued by the others. Moreover, the relative relevance of weights of a given neuron does not change, and so there is no bias introduced in the EA. The rationale behind the normalized Hebbian model is clearly that, since we are preventing the weights from diverging, we want to see whether it unveils more insights about the original, non-normalized model whose weights diverged.

We ran an experimental campaign of 10 independent runs for the normalized Hebbian model, using the same parameters of the non-normalized Hebbian model. We report the results in Figure 5.27 in terms of the fitness $\overline{v}_x$ of the best individuals at the end of evolution, and compare it with the non-normalized Hebbian model results from Section 5.3.4.1. Figure 5.28 plots the histograms of the relative frequency of weights at different time steps of the simulation, for two sample best individuals.

From Figure 5.27, we first notice that normalizing the weights does not impact performance and that the normalized Hebbian model reaches fitness $\overline{v}_x$ comparable (or even better) to the non-normalized Hebbian model. Also, Figure 5.28 proves very

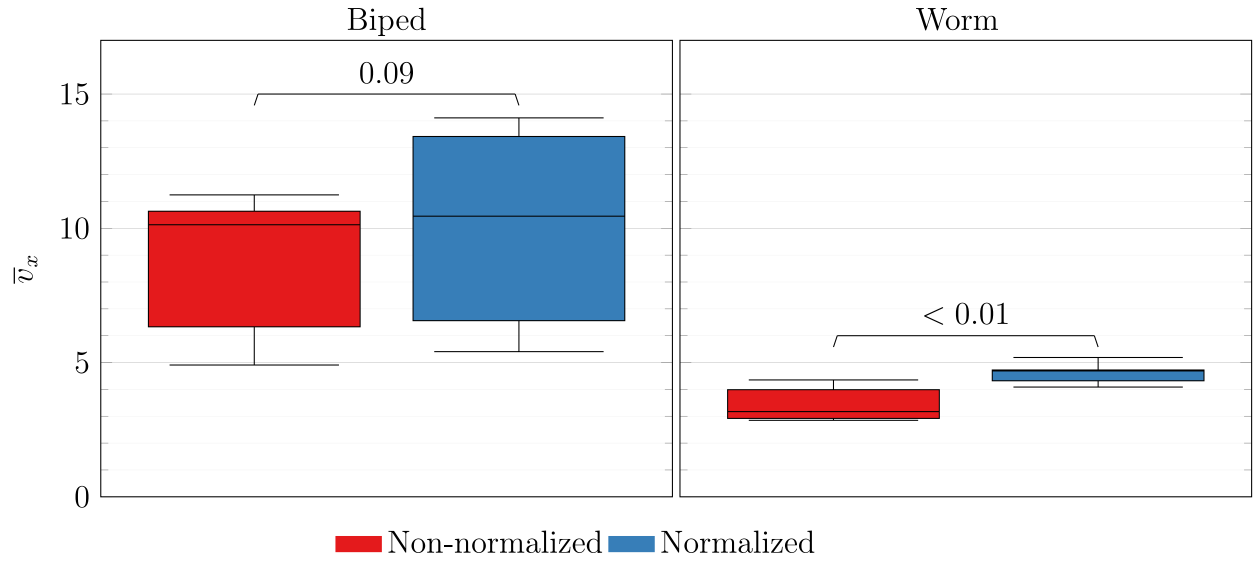


Figure 5.27: Distribution of the velocity $\overline{v}_x$ of the best individuals found in each of the 10 evolutionary runs, obtained with the Hebbian controller (with and without weight normalization to $[-1, +1]$) on the two shapes and the *high* sensory apparatus.

insightful. As expected, weights do not diverge outside of the $[-1, +1]$ range and distribute more uniformly in that range. We also visualized the neuron activations over time and, in contrast with the non-normalized Hebbian model, they display a clear recurrent cyclical pattern. We conjectured the reason for this to be that the normalized Hebbian model evolves to exploit the dynamics of the underlying soft body; intuitively, soft materials are so powerful that, in order to produce a gait, everything the controller is left to do is to instill a cyclical recurring pattern. This hypothesis appears even more grounded if we consider that the frequency of activation "cycles" corresponds to the frequency of the real gait for the two shapes, being higher and more regular for the biped (bipeds gait has a period of $\approx 1.5\,\text{s}$), whereas lower and less regular for the worm (worms gait has a period of $\approx 2\,\text{s}$). Under this light, the dynamical system of the soft body coupled with the dynamical system of the normalized Hebbian controller can be seen as an instance of morphological computation (Paul, 2006), according to which the brain offloads part of the computation to the body.

To test this hypothesis, we repeated the same validation used in Section 5.3.4.2,

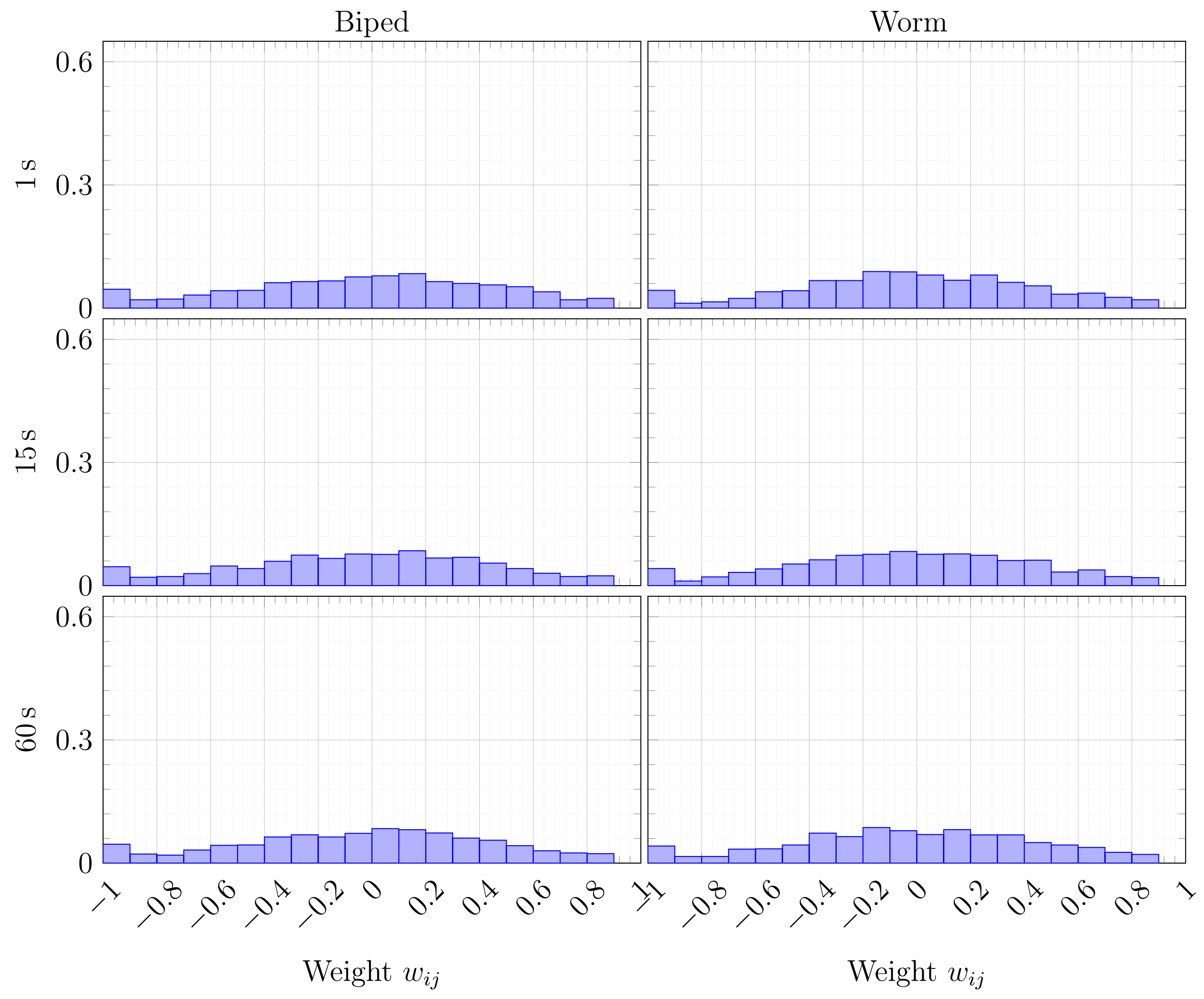


Figure 5.28: Weight distributions for two sample best individuals (one per shape, both with *high* sensory apparatus), at three different time steps of the simulation: 1 s, 15 s, and 60 s. In Hebbian learning with normalization, weights do not diverge.

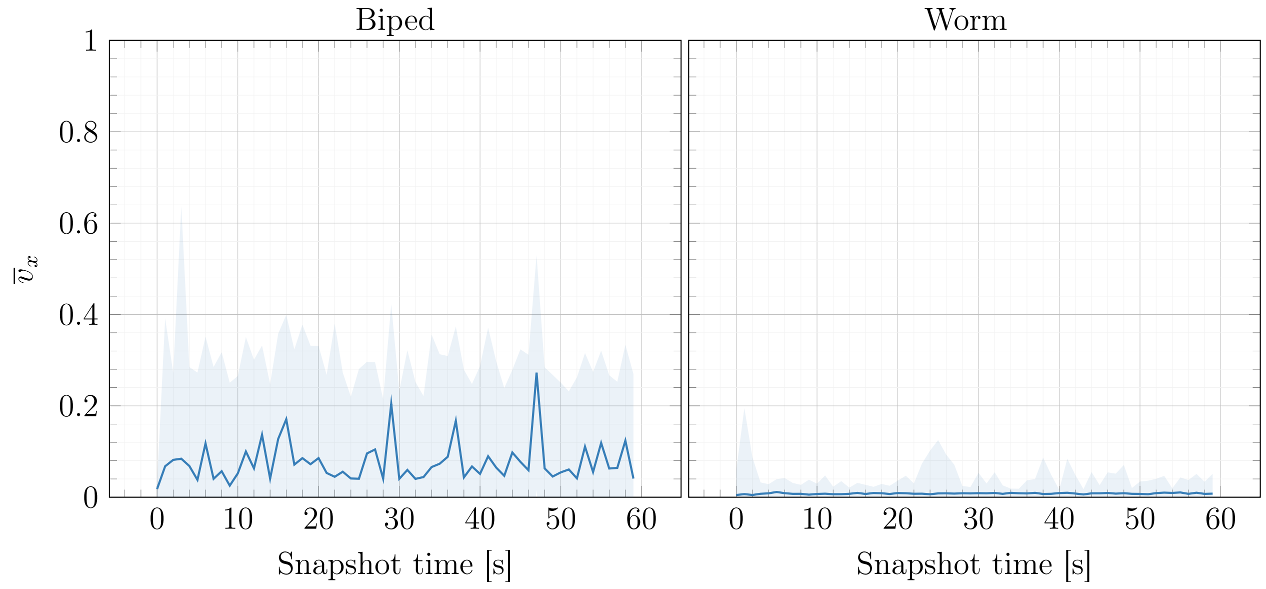


Figure 5.29: Median ± std. dev. (solid line and shaded area) of the velocity $\overline{v}_x$ of the best individuals found in each of the 10 evolutionary runs, with weights frozen at different time steps of the simulation (on x-axis, in s), obtained with the Hebbian controller on the two shapes and the *high* sensory apparatus, with weight normalization to $[-1, +1]$. Hebbian learning with weight normalization does not stabilize.

and report the results in Figure 5.29 using the same semantics of Figure 5.25. We remark that, during the simulation of a snapshot, we freeze the corresponding weights.

Surprisingly, the results are quite different from what was observed for the non-normalized Hebbian model in Section 5.3.4.2. In fact, the median average velocity $\overline{v}_x$ hovers above 0 m/s, meaning that individuals are not producing any useful locomotion at all. In the non-normalized case, see Figure 5.25, $\overline{v}_x$ climbed up very quickly. This analysis holds for both shapes.

Our interpretation is that, while it is possible to freeze Hebbian learning (after a settling period) in the non-normalized case, the same is not true for the normalized case. It is likely that the dynamical system of the Hebbian controller and the dynamical system of the soft body act in unison as a single dynamical system. Hebbian plasticity—in the normalized case—concurs to instill the correct gait dynamics for locomotion, but this is not "true" learning (as specified in Section 5.3.4.2), since freezing Hebbian learning results in extinguishing an entangled portion of the dynamical system.

We conclude that unbounded weights are a necessary component in the original formulation. We speculate the reason why weight divergence does not affect performance may be explained in two different ways. As mentioned earlier, one hypothesis is that a sort of "Boolean" dynamics emerges in the Hebbian model. This hypothesis sounds alluring if we consider that, in theory, the action space for each voxel (which is continuous) might be shrunk to a binary space with just contraction or expansion. To test this hypothesis it would be necessary, as a matter of example, to switch from the foundational task of locomotion to more complex ones that cannot be solved by a mere recurring pattern. We leave this investigation for future work. Another hypothesis is that those synapses that diverge are in fact synapses that do not contribute much to the output and we could, in theory, prune them. This hypothesis seems even more intriguing if we parallel it with the recently introduced "lottery ticket hypothesis" (Frankle and Carbin, 2018), according to which optimizing a dense neural network boils down to optimizing the most effectively initialized sub-networks. However, our setting is slightly different, as weights are initialized at the same value and develop over time through Hebbian plasticity. Future work will consider how to extend the lottery ticket hypothesis to our setting. Finally, for the sake of this study, we were not concerned with other physiological processes other than Hebbian plasticity. There is evidence that other processes, e.g., homeostatic plasticity (Turrigiano and Nelson, 2004), complement it to maintain the overall activity of a neuron within the network, and might balance unconstrained synaptic growth. We leave this investigation as future work.

5.3.4.4 Additional experiments

In the following, we list experiments that we carried out, but delivered less interesting results.

First, we tried to evolve η alongside the $ABCD$ coefficients. The motivation for this is that the joint space of η and the $ABCD$ coefficients is, potentially, a more expressive representation. In particular, it might allow evolution to find learning rates that are tailored to specific Hebbian rules, in parallel with what happens in the human brain, where plasticity is known to operate on a different timescale for the striatum and the amygdala (Costa et al., 2016). The results turned out to be not significantly different than searching in the sole $ABCD$ space. So, for the sake of this chapter, we resorted to the more compact representation between the two, i.e., the one evolving only the $ABCD$ coefficients. The reason for this might be that the EA searches more effectively in the more compact $ABCD$ space, or that differing η values do not benefit the Hebbian learning model.

Second, we experimented with different Hebbian models than the one presented in Section 5.3.3 having a different Hebbian rule for every synapse, that we will hereon label *full*. In particular, we tested:

(a) a *single* model, where all the synapses share a single Hebbian rule;

(b) a *sensors* model, where pre-synaptic connections of input neurons corresponding to sensors of the same type share the same Hebbian rule and there is a unique rule for each one of the other two layers;

(c) a *post-synaptic* model, where the post-synaptic connections of a given neuron share the same Hebbian rule;

(d) a *pre-synaptic* model, where the pre-synaptic connections of a given neuron share the same Hebbian rule.

We ran an experimental campaign for each of the models above, using the same setup of the *full* model. The *single*, *sensors*, and *input* models turned out to be ineffective for all shapes and sensory apparatuses. The *pre-synaptic* model, on the other side, managed to evolve and settle on a plateau. Nevertheless, it still underperformed the *full* model. Since what distinguishes these models is the number of parameters to optimize, they present different compactness-expressivity trade-offs. We then believe these results are a consequence of expressivity, being the latter, for this setting, beneficial.

Finally, we experimented with the initialization of the weights at the beginning of the life of the robot. In addition to the zero initialization adopted in this chapter, we considered a case with initialization of the weights from $\mathcal{U}(-1, +1)$, but we did not find any statistically significant difference. We thus resorted to zero initialization since starting from an idle posture is more sensible for a locomotion task and there is one less causal factor.

5.3.5 Concluding remarks

We observed that the evolved Hebbian controllers are never worse, and often better, than their counterparts based on MLP with evolved weights. This is true for all the tested combinations of two VSR shapes and three sensory apparatuses. Second, we found that the adopted Hebbian learning model does indeed "learn", i.e., the robots perform well even when the weight update is disabled, provided that a sufficient amount of time was previously allotted to the learning process. Third, we found that unbounded weights are a necessary element of learning in our model, in that they lead to an implicit form of pruning. On the contrary, when we normalize weights, robot performance with learning is still comparable; however, if we disable the weight update, performance drops dramatically.

Our instantiation of Hebbianism can be seen as a form of self-organization: the ANN weights collectively self-organize over the course of the robot's lifetime according to their individual Hebbian rules. Thus, we believe our work to be relevant in the study of the time scale of CI.

Chapter 6: Pressure-based soft agents: towards a novel soft robotics formalism

Biological agents have bodies that are mostly of soft tissues. Researchers have resorted to soft bodies to investigate ALife-related questions; similarly, a new era of soft-bodied robots has just begun. Nevertheless, because of their infinite degrees of freedom, soft bodies pose unique challenges in terms of simulation, control, and optimization. We considered VSRs in this thesis, and our choice was motivated; still, VSRs rely on an internal structure of rigid elements to simulate softness (see Chapter 2), limiting their shape-change potential. Here we propose a novel soft-bodied agent formalism, namely Pressure-based Soft Agents (PSAs): they are spring-mass membranes containing a pressurized medium. Pressure endows the agents with structure, while springs and masses simulate softness and allow the agents to assume a large gamut of shapes. PSAs actuate both locally, by changing the resting length of springs, and globally, by modulating global pressure. We evolve the controller of PSAs for a locomotion task on hilly terrain, an escape task from a cage, and an object manipulation task. Our results suggest that PSAs are indeed effective at the tasks, especially those requiring shape change, thus answering question **6**.

6.1 Introduction and related works

Softness is arguably one of the greatest gifts of mother nature. Every living creature on Earth possesses a body that is mostly made of soft tissues. Soft bodies can continuously bend, stretch, and twist, achieving adaptation to the environment; evolution keeps illuminating new ways to exploit softness, from the amazing manipulation feats of cephalopods (Hochner, 2012) to the protozoans of, for example, the genus *Lacrymaria* (Mast, 1911) and the genus *Erythropsidinium* (Guiry and Guiry, 2021), that can contort their soft flagellum to grasp a hard-to-reach prey, allowing for complex hunting dynamics to emerge. It is not surprising that researchers have adopted soft materials to fabricate a new generation of soft robots (Rus and Tolley, 2015). Soft robots promise to achieve shape change, recover from damage (Kriegman et al., 2019), and adapt to novel environments (Shah et al., 2021b). Soft bodies are suitable to investigate ALife-related questions (Joachimczak et al., 2016; Kriegman et al., 2018), including ER (Cheney et al., 2014b), by simulating and optimizing virtual creatures for a task.

At the same time, the simulation and optimization of soft agents pose unique challenges. No analytical methods exist, as soft bodies have infinite degrees of freedom and entail, in general, hard-to-simulate dynamics (Laschi et al., 2016). Moreover, the softness of bodies reinforces the *embodied cognition paradigm* (Pfeifer and Bongard, 2006), which posits a deep entanglement between the "brain" of an agent and the "body" that carries it (Pigozzi, 2022a). While promising in terms of morphological computation (Nakajima et al., 2015; Hauser et al., 2023), i.e., the brain offloading part of the computation to the body, such entanglement makes any co-optimization of the brain and soft body arduous (Lipson et al., 2016). Finally, how to effectively achieve shape change remains an open issue in the literature (Shah et al., 2021a).

We propose a novel formalism to study soft-bodied agents, namely Pressure-based Soft Agents (PSAs). They are bodies of gas enveloped by a chain of springs and masses, with pressure pushing on the masses from inside the body. Pressure endows the agent with structure, while springs and masses simulate softness and allow the agent morphology to assume a large gamut of shapes, modeling soft bodies' many degrees of freedom. PSAs actuate locally, by changing the resting length of springs, and globally, by modulating global pressure. We thoroughly describe the mechanical model and how to simulate it. We also equip the agent with sensors and a closed-loop controller that performs actuation by contracting or stretching the springs and modulating global pressure. See Figure 6.1a for a snapshot of the simulation.

In the realm of virtual creatures, other soft agents formalisms do exist (Milano

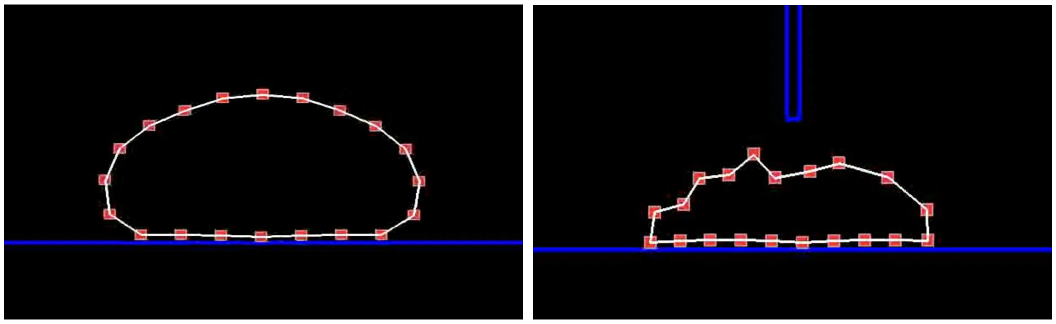

(a) a PSA (b) escaping from a cage

Figure 6.1: (Left) Pressure-based Soft Agents (PSAs) are spring-mass membranes containing a pressurized medium. Red squares are masses, white strings are springs, and blue shapes are other bodies. (Right) PSAs can effectively solve several tasks; depicted here, achieving shape change to escape from a cage.

and Nolfi, 2022), in particular, the voxel-based (Hiller and Lipson, 2012; Medvet et al., 2020b; Bhatia et al., 2021; Benureau and Tani, 2022), which achieves softness through a spring-and-masses system arranged in a grid topology, and the tensegrity-based (Rieffel et al., 2009; Zappetti et al., 2017), which achieves softness by connecting cables that are constantly in tension with rods that are constantly under compression. Albeit far-reaching they might be, these still rely on an internal structure of rigid elements for the sake of modeling softness, severely limiting their ability to change shape. Computer graphics, on the other side, employs also pressure-based soft bodies (Matyka and Ollila, 2003), that rely on internal pressure to maintain structure and can thus stretch and bend in any possible configuration. Pigozzi (2022b) ported those to the virtual creatures world and we hereon extend this work. PSAs thus are not constrained by internal rigid elements that limit the shape change capabilities of voxel-based and tensegrity-based agents. We ask ourselves whether it is possible to

(a) Attain PSAs by endowing pressure-based soft bodies with a robotic controller;

(b) Effectively exploit shape change for PSAs;

(c) Effectively solve an object manipulation task.

We experiment with a two-dimensional simulation of PSAs and carry out an extensive experimental campaign aimed at validating PSAs on three different tasks: a classic locomotion task on hilly terrain to answer (a), an escape task from within a cage to answer (b), and a ball carrier task to answer (c). The second and the third are particularly suitable to this work as they force the agent to shape-shift to escape through an aperture in the cage or grasp an object. We experiment with PSAs of three different sizes and evolve their controller with an established numerical optimizer (Hansen and Ostermeier, 2001).

Our results suggest that PSAs are indeed proficient at solving traditional tasks—locomotion—, tasks that require changing shape—escape—, and tasks that a decent level of cognition—carrying. Moreover, we also show that preventing the controller from modulating pressure (i.e., pressure is the result of only physical interactions) or preventing the controller from changing the resting length of springs (i.e., their length depends on external forces only) makes it impossible for PSAs to solve the tasks.

Looking forward, we believe PSAs can play a role in the simulation of soft-bodied agents. Indeed, many existing soft robots rely on pressure to shape change, using pumps (Kriegman et al., 2021; Shah et al., 2021b), inflatable tubes (Hawkes et al., 2017; Usevitch et al., 2020; Drotman et al., 2021), or jamming technology (Steltz et al., 2009). Finally, we envision many exciting ALife applications, including the modeling of biological cells that, similarly to PSAs, consist of a fluid, the cytoplasm, enveloped by a flexible membrane.

6.2 Proposed agent model

We propose a simple, yet expressive model of soft agents, namely Pressure-based Soft Agents (PSAs). PSAs are expressive as they can potentially assume a wide variety of shapes[1]. They are bodies of gas contained within an envelope (a chain)

[1]From now on, some elements of the notation (i.e., r, f, d, p, a, and s) appear after having been previously used in the text; with a slight abuse of notation, we repurpose them for the needs of this

of springs and masses, with pressure pushing on the masses from inside the body: actuation takes place

(a) *Locally*, by contracting or expanding the springs, and

(b) *Globally*, by changing pressure.

The harmonious execution of these two allows the PSA to assume a large gamut of shapes. By their many degrees of freedom and the co-existence of local and global actuation, PSAs are both

1. Expressive, and
2. Challenging to control.

We take inspiration from the work of Matyka and Ollila (2003) on pressure-based soft bodies for computer graphics, which was adapted for virtual creatures in Pigozzi (2022b). Such models are particularly suitable for bodies that can bend and twist in arbitrary shapes, such as balloons and cloth. To ease modeling, we work with a two-dimensional simulation in discrete time and continuous space. However, we remark that the representations and algorithms of this work are easily portable to the three-dimensional setting.

We define a PSA as the combination of an embodiment, which obeys a *mechanical model* and possesses *sensing* capacities, and a brain, which we implement with a *controller*.

6.2.1 Mechanical model

We outline the mechanical model behind the simulation of PSAs.

chapter.

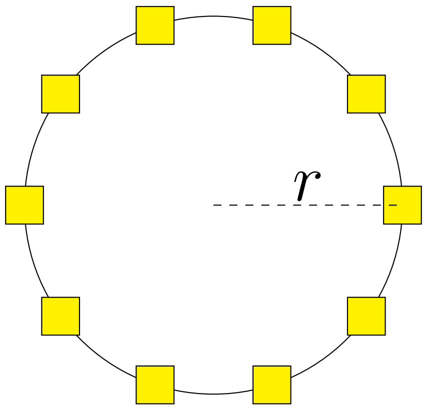


Figure 6.2: The building blocks of a PSA morphology of radius r: yellow squares are masses, and black strings are springs.

6.2.1.1 Morphology

A PSA morphology consists of a *body* of gas contained within an *envelope*. We define an envelope from a circle of radius r[2]; for simplicity, let us assume its center is the origin. We place n_{mass} rectangular masses of rigid material equispaced along the circumference, i.e., at points $r * \cos \frac{2\pi i}{n_{\text{mass}}}, r * \sin \frac{2\pi i}{n_{\text{mass}}}$, and fix their rotation. We join each mass with the previous and the following masses along the circumference with distance joints (i.e., joints fixing the relative position between two masses) of frequency f (the number of complete oscillations they undergo per unit of time), damping ratio d, maximum length l_{max}, minimum length l_{min}. Moreover, an internal pressure p (in Pa) acts on the masses; without pressure, the envelope would collapse because of gravity. The pressure thus endows the body with structure. We remark p is global, in the sense that it is the same for all masses. We summarize the building blocks of a PSA morphology in Figure 6.2.

The masses define the boundaries of the morphology and collide with external bodies. The joints, by choosing appropriate values for f and d, act as springs: they contract and expand in response to forces acting on the masses they join. As a result, the envelope is not rigid but soft, and the morphology deforms under forces acting

[2]We previously employed r in different parts of the text; with a slight abuse of notation, we repurpose it to be a radius

on the masses, either exogenous, e.g., contact with other bodies, or endogenous, i.e., changes in p.

We remark that, indeed, spring-and-damper systems are at the heart of other soft agent simulators, including voxel-based (Hiller and Lipson, 2012; Medvet et al., 2020b) and tensegrity-based (Zappetti et al., 2017). Moreover, springs allow masses to change their relative position, endowing the mechanical model with many degrees of freedom; the envelope can stretch and bend, and the body can contract or expand in limitless configurations. As a result, our model is suitable for approximating the infinite degrees of freedom of soft bodies, including soft robots. At the same time, such freedom entails that directly updating the area of the morphology is not tractable: with PSAs, we solve this problem by indirectly updating the area with p, in a way that we detail in the next paragraph.

As an aside, PSAs can be seen not only as robotic agents but also as a minimal model of a cell: the envelope constitutes the cellular membrane (Singleton et al., 2004), with masses playing the role of membrane proteins and springs the role of lipids. Being fluid, the gas effectively models the cytoplasm (Shepherd, 2006). Finally, p closely resembles turgor pressure acting on the membrane (Pritchard, 2001).

6.2.1.2 Simulation

Area a (in cm^2, because in 2D the area is that of the contour of the masses) alters because of pressure p moving the masses; p, in turn, can be the output of a controller or change according to physical laws. Since pressure is what endows PSAs with structure, we treat the latter as an ablation study in Section 6.4, and focus on PSAs that control Δp (thus affecting p).

At every time step of the simulation, we compute the total pressure acting on the side of a joint and distribute it over the masses. In detail, we:

(1) Query the controller for Δp, and sum it to the p of the previous time step.

(2) For every i-th joint, compute the total pressure acting on its side as $p_i = l_i p$, where l_i is the length of the joint, and the normalized normal vector $\hat{\boldsymbol{n}}_i \in [-1,1]^2$ pointing to the outside of the morphology.

(3) For every j-th mass, let i^- and i^+ be the joints joining it to the previous and next masses in the envelope, respectively. We transform scalar pressure into directed pressure forces $\boldsymbol{p}_{j,i^-} = \frac{p_{i-}}{2}\hat{\boldsymbol{n}}_{i^-}$ and $\boldsymbol{p}_{j,i^+} = \frac{p_{i+}}{2}\hat{\boldsymbol{n}}_{i^+}$ acting on the two joints. We divide by 2 to equally distribute pressure on the masses that anchor a joint.

(4) For every j-th mass, we compute $\boldsymbol{p}_j = \boldsymbol{p}_{j,i^-} + \boldsymbol{p}_{j,i^+}$ and apply it as a force to the mass center. We remark that $\boldsymbol{p}_j$ is indeed in N, as $\hat{\boldsymbol{n}}$ is dimensionless, l_i is in cm, and p is in Pa, with $1\,\text{Pa} = 1\,\text{Ncm}^{-1}$.

(5) Step the physics engine.

Thus, a is not a free parameter (as p), but we affect it through pressure, as higher pressure on the masses implies a larger area, and vice versa. Finally, the overall shape of the PSA morphology, i.e., the arrangement and relative positions of the masses, depends on contacts with other bodies, and changes in the resting length of springs dictated by the controller.

6.2.1.3 Parameters

Masses are squares of side 1 cm and density $2500\,\text{kg m}^{-2}$; we found results to be consistent also with other sizes and densities. After preliminary experiments and relying on our previous knowledge, we set $f = 8\,\text{Hz}$, $d = 0.3$, $l_{\text{max}} = 1.25l$, and $l_{\text{min}} = 0.75l$, where l is the initial length of a spring. As far as r and n_{mass} are concerned, they vary according to the morphology to simulate, as we shall see in the next section.

6.2.2 Sensing

To implement a closed-loop controller, we equip PSAs with sensors. Indeed, sensing is an important property for virtual creatures that interact with an environment (Talamini et al., 2019). In this work, we employ touch, pressure, position, and velocity sensors. Touch sensors perceive whether masses are touching other bodies (e.g., the ground) or not, and, for each mass, return 1 if yes, 0 otherwise. The pressure sensor perceives the current internal pressure p and is thus a proprioceptor. Position sensors perceive the relative x- and y-position of each mass from the center of mass of the morphology. Finally, velocity sensors perceive the x- and y-velocity of the center of mass of the body.

We normalize every sensor reading into $[0, 1]$, and, to introduce sensory memory, compute its average over the last t time steps (using the normalized values). We then concatenate all the sensor readings into an observation vector $\boldsymbol{o} \in [0, 1]^{3n_{\text{mass}}+3}$. After preliminary experiments, we set $t = 25$.

6.2.3 Controller

At every time step of the simulation, we feed the current observation vector $\boldsymbol{o}$ to a controller that decides two sets of actions:

(a) Local actuation: the resting length of the springs, and

(b) Global actuation: the change in pressure Δp.

As far as the former is concerned, given a control value $s \in [-1, 1]$, we instantaneously modify the resting length l of a spring as:

$$l = \begin{cases} l - s(l - l_{\min}) & \text{if } s > 0 \\ l & \text{if } s = 0 \\ l - s(l_{\max} - l) & \text{if } s < 0 \end{cases} \tag{6.1}$$

Thus, $s = -1$ corresponds to the maximum expansion, and $s = 1$ corresponds to the maximum contraction, all other values lying in between. This is the same model of actuation of other soft robotics simulators, e.g., (Medvet et al., 2020b; Bhatia et al., 2021).

Change in pressure, on the other side, requires a domain that is morphology-agnostic, given that different morphologies require different pressure ranges. As a result, we split the controller into a *pressure controller* π_p and a *springs controller* π_s. The former takes as input $\boldsymbol{o}$ and outputs Δp, that we clip to $[p_{\min}, p_{\max}]$ to remain within meaningful boundaries; the latter takes as input $\boldsymbol{o}$ and outputs $\boldsymbol{s} \in [-1, 1]^{n_{\text{mass}}+1}$ (i.e., one control value for every spring).

After preliminary experiments, we implemented π_p as a linear model of the form:

$$\Delta p = \boldsymbol{W}_p \boldsymbol{o} + b_p \tag{6.2}$$

with weights $\boldsymbol{W}_p \in \mathbb{R}^{1 \times |\boldsymbol{o}|}$ and bias $b_p \in \mathbb{R}$. As a result, $\Delta p \in \mathbb{R}$ and the model can choose the output most appropriate to its morphology. Similarly, we implemented π_s with a non-linearity to ensure the output lies in $[-1, 1]$:

$$\boldsymbol{s} = \tanh(\boldsymbol{W}_s \boldsymbol{o} + \boldsymbol{b}_s) \tag{6.3}$$

with weight matrix $\boldsymbol{W}_s \in \mathbb{R}^{(n_{\text{mass}}+1) \times |\boldsymbol{o}|}$ and bias vector $\boldsymbol{b}_s \in \mathbb{R}^{n_{\text{mass}}+1}$. Both π_p and π_s are shallow neural networks (i.e., with no hidden layers): we found this choice to be successful in line with (Mania et al., 2018), which proved linear policies to achieve performance comparable performance to deep neural networks on control tasks.

We focus on optimizing the controller of a PSA for a task. Thus, the parameters we optimize are the controller parameters $\boldsymbol{\theta} = [\boldsymbol{\theta}_p \; \boldsymbol{\theta}_s]$, where $\boldsymbol{\theta}_p = [\boldsymbol{W}_p \; b_p]$ are the pressure controller parameters, and $\boldsymbol{\theta}_s = [\boldsymbol{W}_s \; \boldsymbol{b}_s]$ are the springs controller parameters.

6.3 Experimental procedure

We performed an experimental campaign aimed at answering the following research questions:

RQ1 Is the mechanical model valid to simulate pressure-based soft bodies?

RQ2 Can we control PSAs? In other words, are PSAs capable of solving a classic locomotion task?

RQ3 Can we effectively exploit shape change for PSAs?

RQ4 Can PSAs solve an object manipulation task, that is a more complex robotic task?

We design a specific task for each question; we detail the tasks in the next sub-section.

For all tasks, we evaluate three different PSA morphologies, to get a sense of the effectiveness of PSAs across a wider array of morphological conditions. For the *large* morphology, we set $n_{\text{mass}} = 20$ and $r = 10\,\text{cm}$; for the *medium* morphology, we set $n_{\text{mass}} = 15$ and $r = 7.5\,\text{cm}$; finally, for the *small* morphology, we set $n_{\text{mass}} = 10$ and $r = 5\,\text{cm}$. For the three morphologies, the input size is 33, 48, and 63, respectively. As a result, the size of the parameter space $|\boldsymbol{\theta}|$ is 408, 833, and 1408, respectively.

Since pressure control is what endows PSAs with structure and joint control is what endows PSAs with softness, we investigate whether the two are necessary or not to accomplish the tasks and conduct the following ablation studies in RQ2, RQ3, and RQ4.

6.3.1 Ablation - w/o pressure control

As a first ablation study, we experiment with a configuration *w/o pressure control*, in contrast to the *experimental* configuration considered so far. To this end, we dispense with the pressure controller π_p and let p be the result of physical laws. In particular, the pressure of an ideal gas changes according to the ideal gas law of (Clapeyron, 1834):

$$pa = nRT \tag{6.4}$$

where p (in Pa) is the pressure value, n is the amount of substance (in mol), R is the ideal gas constant (in $\mathrm{cm^2 Pa\ mol^{-1} K^{-1}}$), and T is temperature (in K). We remark that many real gases do behave as ideal under various temperature and pressure conditions (Cengel et al., 2011). By fixing T, the right-hand side of Equation (6.4) is constant: then, p must vary to accommodate changes in a and balance the equation. At every time step, we compute a by triangulation and plug it into Equation (6.4) to compute p. The controller is then in charge only of the springs' resting length.

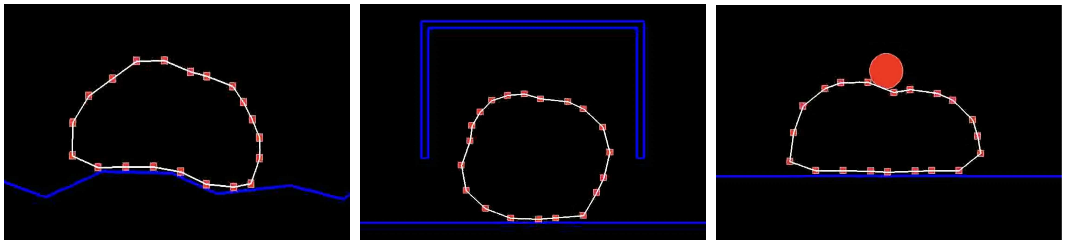
We set $T = 288.15\,\mathrm{K} = 15\,^\circ\mathrm{C}$ to simulate room temperature, and the gas (the PSA is filled with) to be N_2 (nitrogen), a cheap and common gas. n is the ratio between the gas mass m (in kg) and the molar mass (in $\mathrm{kg\ mol^{-1}}$), that is $0.028\,031\,4\,\mathrm{kg\ mol^{-1}}$ for N_2. We set $m = 0.05\,\mathrm{kg}$, $m = 0.075\,\mathrm{kg}$, and $m = 0.1\,\mathrm{kg}$ for the small, medium, and large morphologies respectively. As usual, $R = 8.314\,562\,6\,\mathrm{cm^2 Pa\ mol^{-1} K^{-1}}$ is the ideal gas constant.

For this configuration, the size of the parameter space $|\boldsymbol{\theta}|$ is 374, 784, and 1334, respectively for the three morphologies; thus, disabling pressure control results not only in (potentially) simpler actuation but also in a smaller search space that might benefit optimization.

Finally, for the configuration with pressure control, we set $p_{\max} = \frac{nRT}{\pi r^2}$, where πr^2 is the area of a perfect circle of radius r and $p_{\min} = 0.2 p_{\max}$ to prevent the PSA from collapsing.

6.3.2 Ablation - w/o spring control

As a further ablation study, we experiment with a configuration *w/o spring control*, in contrast to the *experimental* configuration described in the chapter. To this end, we dispense with the controller of the springs π_s and let the springs keep their initial resting length throughout the simulation; thus, the actual length of the



(a) locomotion (b) escape (c) carrier

Figure 6.3: The tasks considered in our experiments. The red circle on the right is a ball object.

springs depends on external forces and their oscillations only. The controller is then in charge only of the pressure.

For this configuration, the size of the parameter space $|\boldsymbol{\theta}|$ is 34, 49, and 64, respectively for the three morphologies. As with the other ablation study, disabling springs control results not only in (potentially) simpler actuation but also in a much smaller search space that might benefit optimization.

6.3.3 Tasks

We evaluate our method on three tasks: locomotion, escape, and object carrying. The tasks differ by increasing levels of environmental interaction: while locomotion can potentially be solved by an open-loop controller, escaping from a cage cannot be solved without the agent sensing its surroundings and making decisions accordingly. Finally, object carrying requires more fine-grained cognition: not only the agent must sense the object to be carried, but it also must act on it to modify its state (i.e., position). See Figure 6.3 for sample frames from these tasks.

6.3.3.1 Locomotion

Locomotion is a classic task in ER (Sims, 1994; Nolfi and Floreano, 2000), and provides a benchmark of basic control skills. It consists of walking as fast as possible over a terrain along the x direction, over an amount of simulated time t_{final}. The fitness function is the velocity $\overline{v}_x$ of the center of mass of the PSA over the simulation. We set $t_{\text{final}} = 30\,\text{s}$. We consider a hilly terrain similar to Sections 4.1, 4.2 and 5.3, with bumps of different heights and distances. For a given seed, we randomly procedurally generate the bumps by sampling distance from one bump to another from $N(\frac{r}{3}, 0.25)$ (to be then multiplied by r, to account for the size of the agent) and sampling the height of each bump $N(0, \frac{r}{3})$.

6.3.3.2 Escape

Escape is particularly suitable for soft agents (Cheney et al., 2015), as it forces the agent to change its shape to pass through an aperture radically. At the onset of each simulation, we place the PSA within a cage. The cage amounts to a roof and two walls, with one small aperture per side. The task consists of escaping as fast as possible in any direction over a maximum amount of simulated time t_{final}. The fitness function is the average velocity $\overline{v}$ of the center of mass of the PSA over the simulation, regardless of the direction. We set $t_{\text{final}} = 30\,\text{s}$. The cage is rigid, immobile, and indestructible, forcing the PSA to contort itself and squeeze through one of the apertures. After preliminary experiments, for a PSA of radius r, we set the roof height to $2r+1$, the walls $3r$ apart, and the apertures one-third of the roof height. Escape differs from locomotion in that there is a clear-cut condition for "solving" it, namely when all of the PSA masses are outside of the cage: if this is the case, we terminate the simulation.

6.3.3.3 Carrier

The third task is a relevant instance of object manipulation. Object manipulation is not only ubiquitous in robotics research, but it also relates to "fine" motor control in biology: in contrast to "gross" motor skills (e.g., locomotion and escape), fine motor skills require muscle coordination at a more fine-grained level and arguably aids in the development of intelligence (Kelly and Natale, 2020). At the onset of each simulation, we place the PSA on a flat ground and a round object (the *ball*) of radius

$\frac{r}{4}$ on top of it. The task consists in walking as fast as possible along the x direction, over an amount of simulated time t_{final}. The fitness function is the velocity $\overline{v}_x$ of the center of mass of the PSA over the simulation. We set $t_{\text{final}} = 30\,\text{s}$. If the ball touches the ground, we terminate the simulation. Given that we compute the fitness using t_{final} and not the actual simulation time, evolution favors individuals who keep the ball afloat as long as possible. After preliminary experiments, we set the ball to have a radius of $\frac{r}{4}$ and a density of $500\,\text{kg cm}^{-2}$

6.3.4 Optimization

We optimize the controller parameters $\boldsymbol{\theta}$ with Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen and Ostermeier, 2001; Hansen, 2016), an established numerical optimizer. While it is possible to use any optimization algorithm, we found CMA-ES to be stable across the different tasks, also thanks to the small size of the search space (Müller and Glasmachers, 2018). CMA-ES iteratively optimizes the solution in the form of a multivariate normal distribution, against a given fitness function. At each iteration, it samples the distribution obtaining a population of solutions, and then updates the parameters of the distribution based on the best half of the population. CMA-ES employs non-trivial heuristics while updating the distribution—we refer the reader to (Hansen and Ostermeier, 2001) for more details.

We use the default parameters suggested in (Hansen, 2006), namely the initial step size $\sigma = 0.5$ and the population size $\lambda = 3\lfloor \log |\boldsymbol{\theta}| \rfloor$. We set the initial vector of means by sampling uniformly the interval $[-1, 1]$ for each vector element. We let CMA-ES iterate until 10 000 fitness evaluations have been done.

6.3.5 Settings

For each experiment, we performed 10 evolutionary runs by varying the random seed for CMA-ES and the terrain generation in locomotion. We carried out all statistical tests with the Mann-Whitney U rank test for independent samples. We employ as physics engine the Python wrapper[3] to Box2D (Catto, 2011), a popular 2D physics library written in C++. We set the simulation frequency to 60 Hz and left all other parameters unchanged. We remark that, for a given seed and controller, all simulations are deterministic. For CMA-ES, we used the implementation of (Ha, 2017), which is a wrapper around the pycma library (Hansen et al., 2019), and, at a given iteration, parallelize fitness evaluations using multiprocessing. We made the code publicly available at `https://github.com/pigozzif/PressureSoftAgents/tree/stable-release`.

6.4 Results

We present the results of this chapter.

6.4.1 RQ1: validation of the mechanical model

We validate whether the proposed mechanical model is suitable for simulating pressure-based soft bodies. In particular, for a PSA of radius r, we verify if there exists a p such that a is that of a perfect circle of radius r; in this way, we assess whether our model can correctly simulate a balloon—an ideal pressure-based soft body.

To this end, we conduct the following experiment:

(a) We define a controller that, for a morphology of n_{mass} masses and radius r, outputs $\boldsymbol{s} \in \mathbf{0}^{n_{\text{mass}}+1}$ and $\Delta p = \frac{p_{\max}}{100}$ at every time step. In other words, it does not alter the resting length of springs, while constantly increasing the pressure.

(b) For each morphology, we simulate flat terrain using the aforementioned controller, setting $p = p_{\min} = 0$ at the beginning.

[3] `https://github.com/pybox2d/pybox2d`

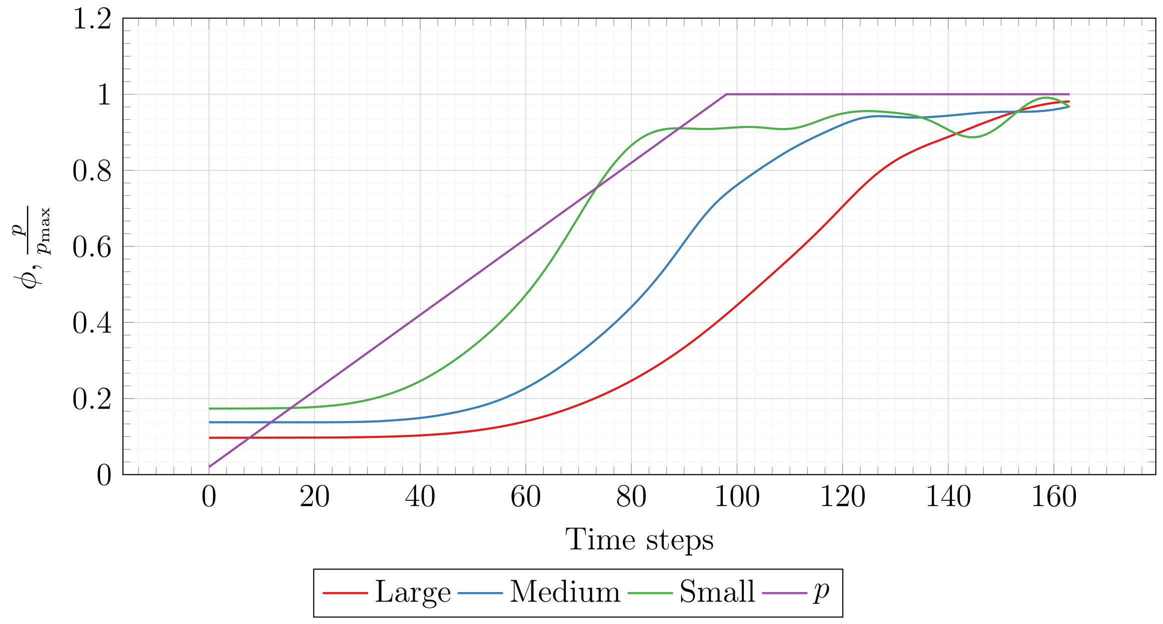


Figure 6.4: Ratio ϕ between the PSA area a and the area of a perfect circle of the same radius, together with relative pressure $\frac{p}{p_{\max}}$, obtained with three sizes. Our proposed mechanical model effectively simulates pressure-based soft bodies, as ϕ approaches 1 by constantly increasing pressure.

(c) For each time step of the simulation, we record pressure p and $\phi = \frac{a}{\pi r^2}$ as the performance indexes, πr^2 being the area of a perfect circle of radius r.

If our proposed mechanical model correctly simulates pressure-based soft bodies, there must exist a value p such that $\phi = 1$.

We report the results in Figure 6.4; being this simulation deterministic, there is no need to report error bars. According to the figure, the results are qualitatively similar for the three morphologies. The area starts just above 0, since $p = 0$ and there is no pressure supporting the envelope; it then smoothly increases throughout the simulation, before plateauing at 1 after $p = p_{\max}$.

Through that evidence, we can answer positively to RQ1: our proposed mechanical model is suitable for simulating pressure-based soft bodies.

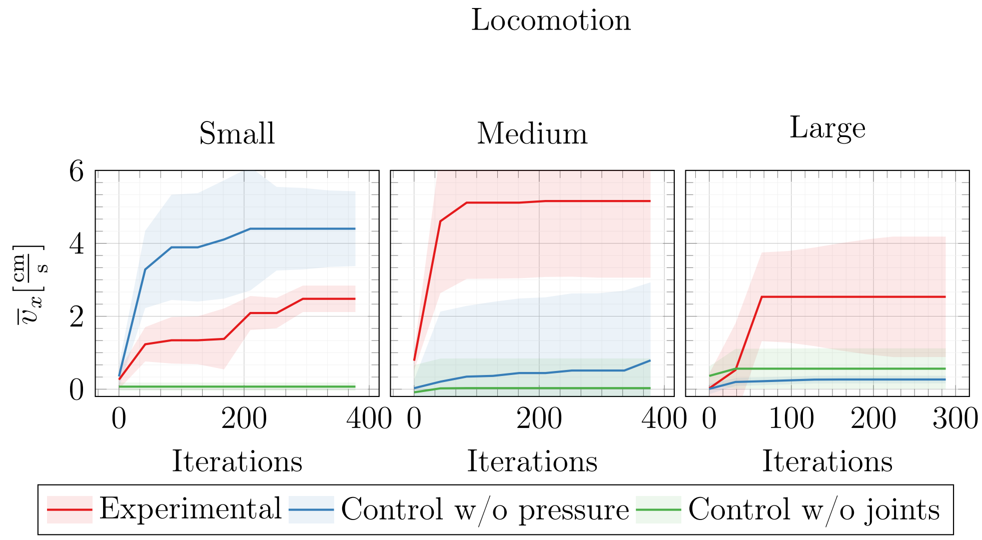


Figure 6.5: Median ± standard deviation (solid line and shaded area) of the average velocity of locomotion for the best individuals found during each evolutionary run, obtained with three sizes, both experimental and control treatments. Our agent model is effective at locomotion on hilly terrain.

6.4.2 RQ2: can we control PSAs?

To validate the effectiveness of our proposed agent model, we measure the performance of PSAs in a classic locomotion task, in three different settings: the experimental treatment, the control treatment without pressure control, and the control treatment without springs control. In all cases, we use $\overline{v}_x$ as the performance index.

We summarize the results in Figure 6.5, which plots $\overline{v}_x$ in terms of median ± standard deviation for the best individuals throughout evolution. Moreover, Figure 6.6 reports boxplots for the distribution of $\overline{v}_x$ of the best individuals. For every morphology, we also show in Table 6.1 the p-values for the statistical tests against the null hypothesis of equality between the medians for two cases: the experimental treatment against the control without pressure actuation and the experimental treatment against the control without springs actuation.

From the figures, we see that our proposed agent model is effective at the task of locomotion and succeeds in mastering it, regardless of the morphology. We visually inspected the behaviors and found them to be highly adapted for a locomotion task

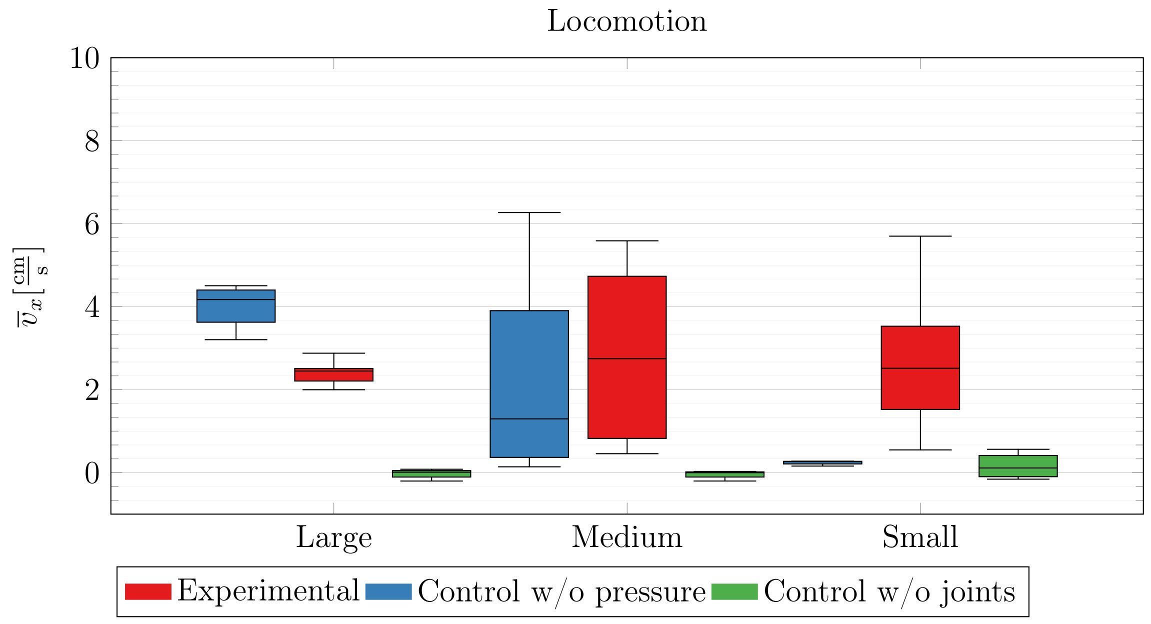


Figure 6.6: Distribution of the average velocity of locomotion for the best individuals found for each evolutionary run, obtained with three sizes, both experimental and control treatments. Dispensing with pressure control generally hampers performance in locomotion on hilly terrain, while dispensing with springs control always results in abysmal performance. *p*-values are significant.

	Locomotion		Escape		Carrier	
Size	w/o pressure	w/o joints	w/o pressure	w/o joints	w/o pressure	w/o joints
Small	n.s.	< 0.001	0.0001	< 0.0001	< 0.0001	< 0.0001
Medium	n.s.	0.002	< 0.0001	< 0.0001	< 0.0001	< 0.0001
Large	0.0001	n.s.	n.s.	< 0.0001	< 0.0001	< 0.0001

Table 6.1: *p*-values for the two-sided test of significance between the experimental treatment and the two control treatments (w/o pressure control and w/o springs control), for each size and task. n.s. stands for a non-significant result (at the 0.05 confidence level and applying Bonferroni's correction).

on hilly terrain. PSAs evolve to "roll" over the ground, sliding the masses one after the other, and modulating pressure to have the right shape at the right moment to overcome bumps: in fact, we found that decreasing pressure right before a bump allows the PSA to lower its center of gravity and generate enough momentum to walk over it. On the other side, increasing pressure on flat portions of terrain allows the PSA to bounce over it and generate enough momentum to walk faster. Interestingly, we found some individuals to show life-like behaviors: as a matter of example, when approaching bumps, some stretched out their front (masses and joints) to reach over the tip of the bump, grasp it, and finally walk over it. Others appeared to adopt the same strategy to "probe" the terrain in front of them, and plan future actions accordingly. We made videos available at `https://pressuresoftagents.github.io`.

According to the figures, PSAs evolved without pressure control are not as effective. In two morphologies out of three, $\overline{v}_x$ barely departs from its initial value, meaning that no adaptation takes place. To gain further insights into this phenomenon, we visually inspected the evolved behaviors. We found them to be not adapted to a locomotion task on hilly terrain. In particular, Medium and Large PSAs often get stuck in hollows of the terrain; other times, they unsuccessfully struggle to walk over a bump. We believe the reason is the lack of pressure control: as mentioned before, modulating pressure allows the PSA to deform according to the terrain. Surprisingly, the same is not valid for the Small morphology, which even succeeds in outperforming its experimental counterpart. We found that Small individuals without pressure control evolved to twist their springs until reaching a deformed configuration with encroaching springs; the pressure force then pushes in the wrong direction and thrusts outwards as if the robot was a "rocket". At that point, the agent starts "flying" over the terrain and unfairly achieves a very high fitness. That is just one of the ingenious examples of perverse instantiation found in the ALife world (Lehman et al., 2020). We observed the emergence of the said behavior also for some Medium individuals (explaining their large variance) and never again. From these experiments, we can speculate that our model gives rise to a rugged fitness landscape, with many local minima defined by instances of perverse instantiation that can be evolved.

Focusing on the ablation treatment without springs control, there is no evolution taking place at all: the best individuals do not manage to perform better than the initial fitness. Not surprisingly, these individuals sit idle for the entire simulation and fail to achieve any behavior due to the lack of spring control. Pressure control is thus not enough.

We remark that, as shown in Table 6.1, p-values are significant for the majority of the comparisons.

Through that evidence, we can answer positively to RQ2: we conclude that, after optimization, it is possible to control PSAs for a task requiring a basic level of cognition, considering the challenging nature of the hilly terrain. Moreover, ablating the pressure control component of the controller results in worse performance, especially for bigger (and, we believe, more realistic) morphologies, and ablating the springs control component results in no evolution at all, suggesting that global control and local control are inextricable parts of our proposed agent model.

6.4.3 RQ3: can we effectively exploit shape change?

To assess the shape-changing abilities of our proposed agent model, we measure the performance of PSAs in an escape task, in three different settings: the experimental treatment, the control treatment without pressure control, and the control treatment without springs control. In all cases, we use $\overline{v}$ as the performance index.

We summarize the results in Figure 6.7, which plots $\overline{v}$ in terms of median $\pm$ standard deviation for the best individuals throughout evolution. Moreover, Figure 6.8 reports boxplots for the distribution of $\overline{v}$ of the best individuals and, as before, Table 6.1 lists the corresponding p-values for the statistical tests against the null hypothesis of equality between the medians for two cases: the experimental treat-

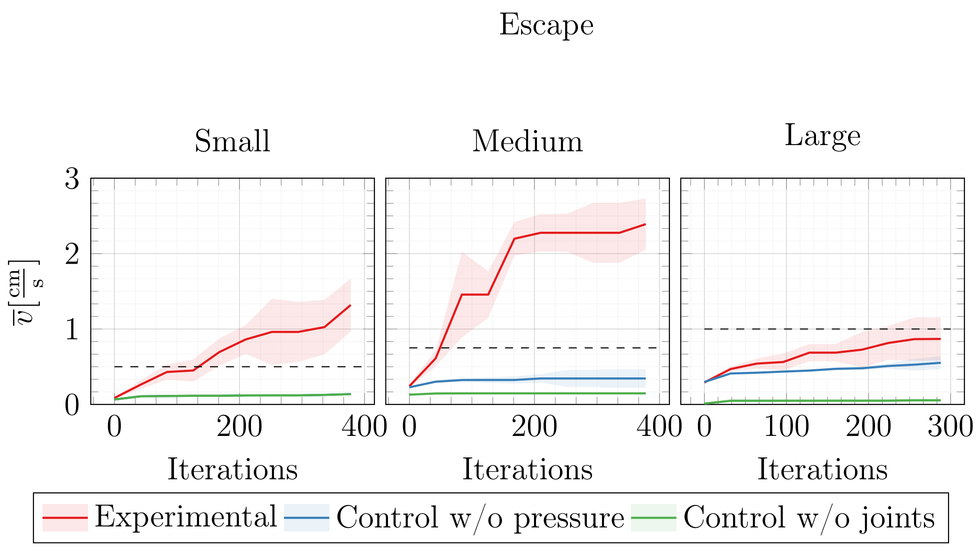


Figure 6.7: Median ± standard deviation (solid line and shaded area) of the average velocity of escape for the best individuals found during each evolutionary run, experimental, and control treatments. Dashed lines are thresholds for the task to be solved (all of the masses are outside of the cage). Our agent model is effective at shape-changing to escape from a cage.

ment against the control without pressure actuation and the experimental treatment against the control without springs actuation.

We remark that we say the task to be "solved" once all of the masses of the PSA are out of the cage; that happens when $\overline{v} \approx 1.0$ for Large, $\overline{v} \approx 0.75$ for Medium, and $\overline{v} \approx 0.5$ for Small—that is, the dashed lines in Figure 6.3b. From this consideration and the figures, we see that our proposed agent model effectively solves the task of escape from a cage for the Small and Medium morphologies and almost manages to solve it for the Large morphology. We visually inspected the behaviors and found them to be highly adapted for an escape task. Effective individuals decreased their internal pressure to reduce their area, almost flattening on the ground (see Figure 6.1b for a snapshot); then, they slithered through one of the apertures to successfully exit the cage (we remark that agents cannot evolve their initial pressure). Albeit this turned out to be a recurring pattern, we observed some variations. Some individuals, for example, evolved a repulsion for the walls: as soon as any of their touch sensors perceived a wall, they would contract themselves in the opposite

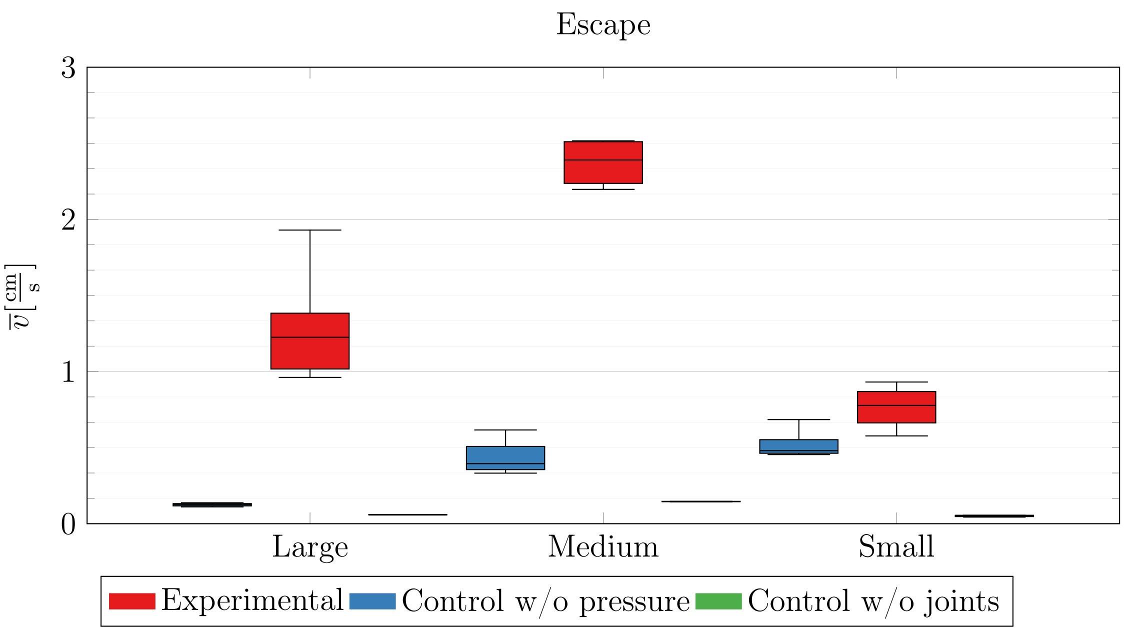


Figure 6.8: Distribution of the average velocity of escape for the best individuals found for each evolutionary run, obtained with three sizes, experimental and control treatments. Dispensing with pressure control or joint control makes it impossible to escape from a cage. p-values are significant.

direction, with a sudden wave that resembled a snail retracting its antennae upon perceiving a danger. The evolution of this trait might be since, early in the optimization, we found many individuals to become tangled up as one of the walls wedged between two of their masses (joints, having no mass, cannot oppose penetration). Other individuals, when flattened, would crawl as big cats do when approaching prey, cautiously stretching out one mass after the other. We made videos available at `https://pressuresoftagents.github.io` (after the videos for the locomotion task).

At the same time, evolution did not find effective individuals in the two ablation treatments. From the plots of Figure 6.7, we see that $\overline{v}$ barely departs from its initial value in both cases. We visually inspected the evolved individuals and found them to be not adapted at all for an escape task: all of them wobbled towards the walls to gain a little $\overline{v}$, but did not attempt squeezing through the apertures. Intuitively, the reason is their inability to control pressure, as they cannot shape change to effectively solve the task. Figure 6.8 corroborates these findings and Table 6.1 shows that p-values are significant for all the comparisons.

Through that evidence, we can answer positively to RQ3: PSAs can effectively leverage shape change to solve a task that requires squeezing through a small aperture, after optimization. Moreover, ablating the pressure control component of the controller or the springs control component results in no adaptation. To the best of our knowledge, other works on soft robots solve this task by joint optimization of morphology and control (Zardini et al., 2021; Bhatia et al., 2021), which is complex, or morphology alone (Cheney et al., 2015), that might be less feasible than control alone in a real-world setting. In the future, we envision such escape tasks to be the starting point of more interesting scenarios, like crawling inside caves with challenging terrain, as well as navigating "claustrophobic" mazes as cephalopods can do (Moriyama and Gunji, 1997).

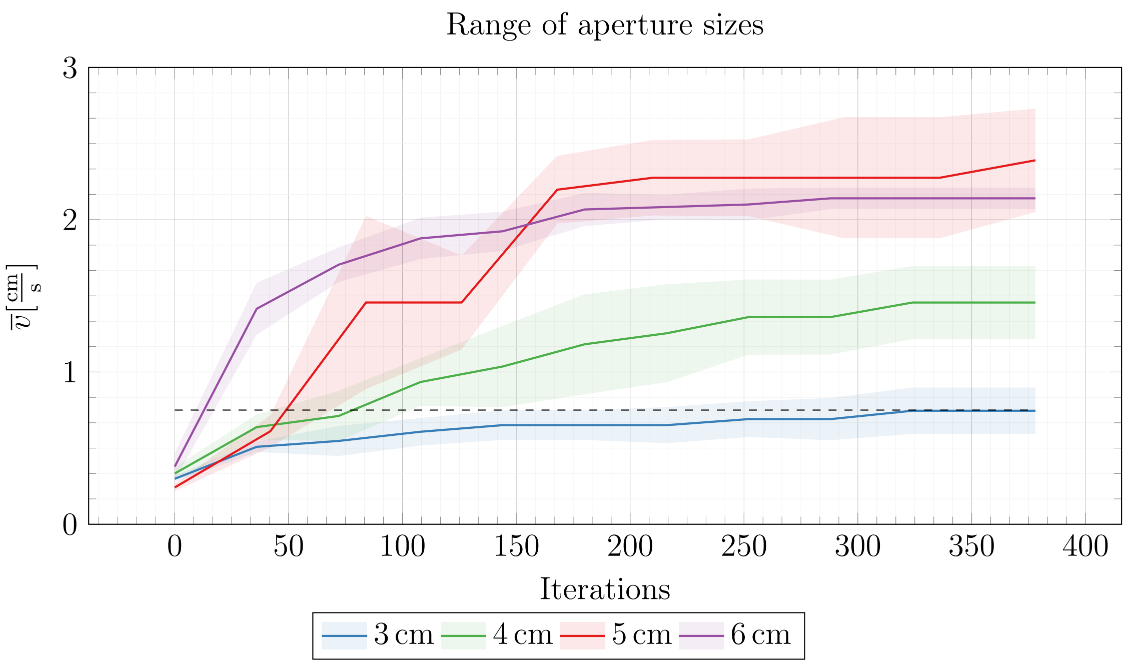


Figure 6.9: Median ± standard deviation (solid line and shaded area) of the average velocity of escape for the best medium individuals found during each evolutionary run, with different aperture sizes. The dashed line is the threshold for the task to be solved (all of the masses are outside of the cage). There exists a critical aperture size beyond which PSAs are no longer effective.

6.4.3.1 Critical aperture size

To gain deeper insights into the Escape task, we ask ourselves how the aperture size affects performance. To answer this question, we repeated the same experimental campaign of Section 6.4.3; we focused on the Medium morphology (because it is the middle and was able to accomplish the task) and experimented with three additional aperture sizes: 3 cm, 4 cm, 6 cm (we recall that 5 cm is the one employed so far for this morphology). We report the results in Figure 6.9.

PSAs can squeeze even through smaller apertures. Still, performance worsens monotonically as the aperture becomes smaller, to the point that PSAs can barely squeeze through the smallest ones, since the blue line barely reaches the threshold required for the task to be solved.

This experiment suggests that PSAs can achieve shape change for an escape task for a range of apertures, but only down to a "critical" aperture size. After inspecting the videos, we speculate the reason to be that too small an aperture requires the PSA to flatten to the point of making the "rolling" pattern of walk behavior hard to achieve.

6.4.4 RQ4: can we solve an object manipulation task?

To assess the cognitive abilities of our proposed agent model, we measure the performance of PSAs in an object manipulation task requiring the agent to carry a ball, in three different settings: the experimental treatment, the control treatment without pressure control, and the control treatment without springs control. In all cases, we use $\overline{v}_x$ as the performance index.

We summarize the results in Figure 6.10, which plots $\overline{v}_x$ in terms of median $\pm$ standard deviation for the best individuals throughout evolution. Moreover, Figure 6.11 reports boxplots for the distribution of $\overline{v}_x$ of the best individuals and, as before, Table 6.1 lists the corresponding p-values for the statistical tests against the null hypothesis of equality between the medians for two cases: the experimental treatment against the control without pressure actuation and the experimental treatment against the control without springs actuation.

The figures show that our proposed agent model manages to solve the task of carrying a ball. We visually inspected the behaviors and found the majority of them to be adapted for the carrier task. The most commonly evolved strategy worked as follows. First, the PSA waits for the ball to fall on top of it: being the robot body soft, the ball usually bounces up and down until stabilizing in a few seconds. Once the ball stabilizes, the PSA starts walking forward with its signature rolling gait (see Section 6.4.2). One possible way of holding the ball is to get it stuck between two masses; although successful at the very beginning of the simulation, this stratagem would prove fatal once the masses start rolling forward as the ball would soon touch the ground. Instead, the best individuals evolved to stretch their springs on their

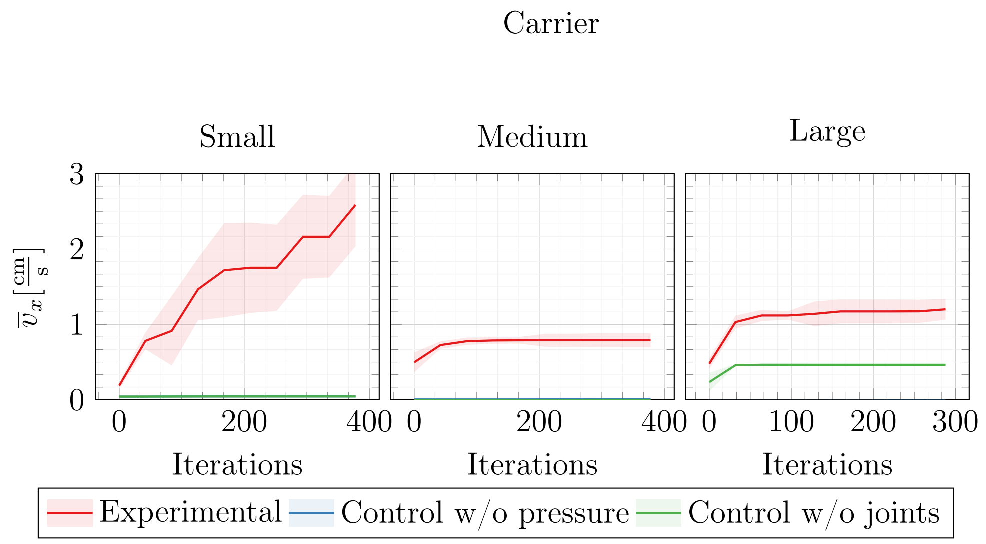


Figure 6.10: Median ± standard deviation (solid line and shaded area) of the average velocity of object carry for the best individuals found during each evolutionary run, experimental, and control treatments. Our agent model is effective at carrying an object.

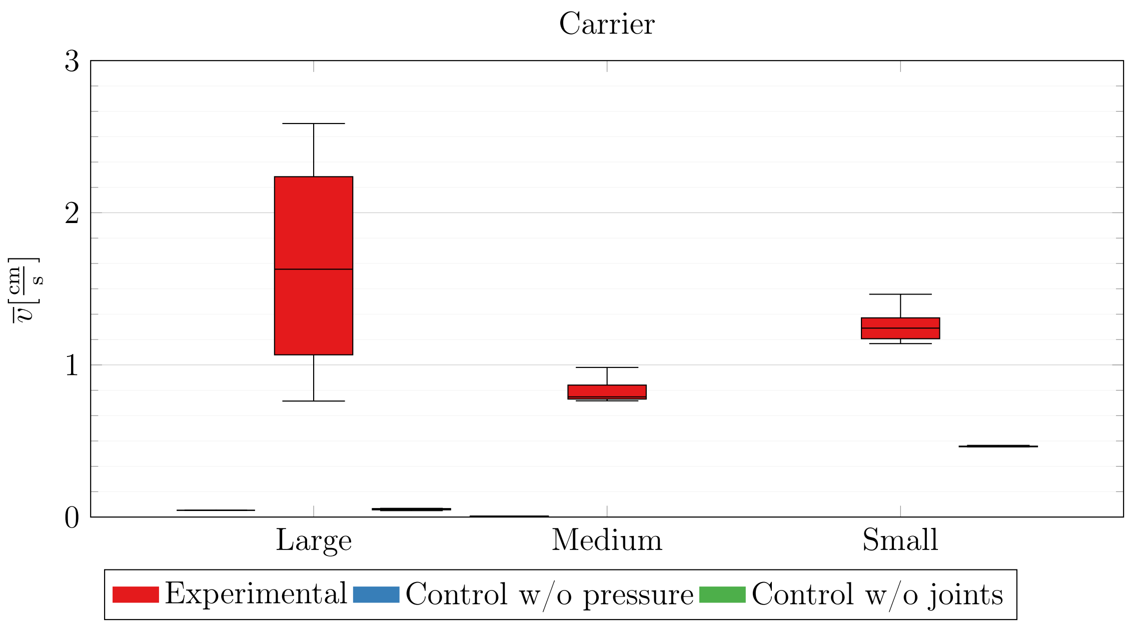


Figure 6.11: Distribution of the average velocity of object carrying for the best individuals found for each evolutionary run, obtained with three sizes, experimental and control treatments. Dispensing with pressure or joint control makes it impossible to carry an object. p-values are significant.

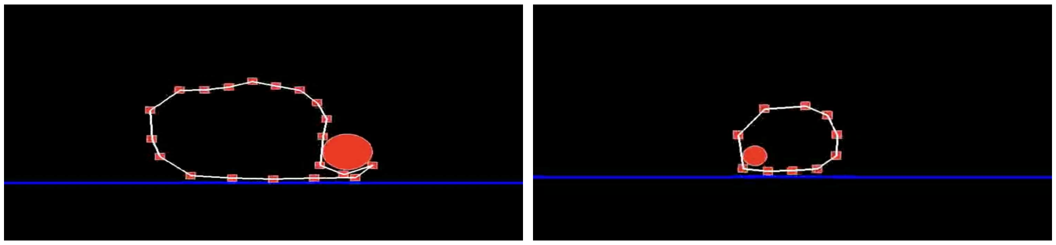

(a) (b)

Figure 6.12: Interesting behaviors evolved for the Carrier task: to prevent the ball from falling on the ground, the left individual engulfs it, while the right one holds it inside.

top as if they were a springboard: the ball bounces up and down and, as the robot moves forward, does not fall on the ground. We found this strategy to last long enough to achieve decent fitness, as shown by the figures. We made videos available at `https://pressuresoftagents.github.io` (at the bottom of the page), including a case of perverse instantiation (see next paragraph).

Still, that common strategy had its shortcomings. In particular, we witnessed very few individuals keeping the ball away from the ground until the end of the simulation. Interestingly, those that managed to last until the end developed an emergency maneuver to stave off the fall of the ball: as the ball slid away from their back, they contorted their front springs to engulf the ball as shown in Figure 6.12a: this halted their locomotion, but also prevented the ball from touching the ground. Additionally, we found a relevant case of perverse instantiation evolving among Small individuals. The most successful of them let the ball pass through a gap among their masses (springs, by definition, have no mass and are thus permeable) to hold it inside of their body in a “hamster wheel” fashion, as shown in Figure 6.12b. As can be seen in Figure 6.10 from the upward trend of the Small experimental treatment, this allowed for faster locomotion, as the ball did not risk falling off to the ground. Albeit acceptable, we consider this strategy to be perverse instantiation since it is taking advantage of our simulation formalism and we suppose PSAs to be impermeable in the real world.

There is a stark contrast between the performance of our experimental treatment and the performance of the two controls, stressed by the significant p-values of Table 6.1: PSAs that do not control their pressure or their springs do not manage to evolve at all. We visualized the behaviors of the control individuals and found them to sit idle for the entire duration of the simulation, barely keeping the ball on top of them by inertia.

Through that evidence, we can answer positively to RQ4: after optimization, it is possible to control PSAs for an object manipulation task that, as commented in Section 6.3.3, requires a finer-grained level of cognition and environmental interaction than, e.g., locomotion. Moreover, ablating the pressure control component or the springs control component of the controller results in no evolution at all, suggesting—once again—that global control and local control are inextricable parts of our proposed agent model.

6.5 Conclusion

Because of their infinite degrees of freedom, soft agents pose unique challenges in terms of simulation, control, and optimization. Here we propose a novel soft-bodied agents formalism, namely Pressure-based Soft Agents (PSAs): a spring-mass membrane containing a pressurized medium. Actuation takes place by changing the length of springs or modulating global pressure. By such a mechanical model, PSAs can assume a large gamut of shapes.

We experimentally investigate whether it is possible to control PSAs and exploit their shape change potential. That is what the chapter demonstrates:

(a) We can control PSAs, as optimization finds effective controllers for a locomotion task on hilly terrain, a task that requires a basic degree of cognition to be solved;

(b) We can effectively exploit shape change for PSAs, as optimization finds effective controllers for the task of escape from a cage, a task that requires the agent to contort itself and squeeze through a small aperture;

(c) We can quite successfully optimize PSAs for an object manipulation task of carrying a ball, a task that requires a fine-grained level of cognition and environmental interaction.

Among the limitations of this work, the manufacturability of PSAs is yet to be proven. We believe the model to be promising, as many real soft robots do rely on pressure to achieve shape change (Usevitch et al., 2020; Kriegman et al., 2021) Future work will address these issues; for the moment, we agree with (Kriegman, 2019) that virtual creatures can be "as beautiful and complex as life itself". Indeed, we believe PSAs advance reality by providing a unified framework for soft-bodied agents that rely on shape change so that several aspects can be tested before experimental implementation. Other future directions include three-dimensional simulation, as well as the simulation of phenomena related to biological cells, such as phagocytosis and mitosis. With this in mind, we may extend this model by linking several envelopes together (sharing some masses). Thus, a robot will consist of a tissue with several cavities. Opportunities are indeed many, and we have open-sourced our code with a gym (Brockman et al., 2016) interface to encourage usage by other researchers.

Chapter 7: Conclusion

This thesis delves into the realm of Evolutionary Robotics (ER) (Bongard, 2013) and presents a shift in perspective by incorporating insights from Collective Intelligence (CI) (Ha and Tang, 2022). The stagnation in ER progress, primarily due to the lack of adaptability in existing approaches, necessitates a paradigm change (Eiben, 2021a).

We focus on the case of simulated Voxel-based Soft Robots (VSRs) (Hiller and Lipson, 2012). As a preliminary, we investigate what factors impact the diversity and effectiveness of VSRs (Pigozzi et al., 2023b). We then showcase the inherent modularity of VSRs, emphasizing their potential as a substrate for CI. Through experimentation, we demonstrate a way to foster and control modularity within VSR controllers (Pigozzi and Medvet, 2022); and also, a way to embed neural networks within modules with no explicit inter-module communication, enabling our robots to only implicitly communicate through mechanical interactions (Pigozzi et al., 2022).

We also explore the challenge of enabling modules to reason about global properties based solely on local information, a previously unexplored dimension in the field (Pigozzi et al., 2023c). Furthermore, we investigate various levels of adaptation (Sipper et al., 1997) within the VSR framework. We unravel the impact that the morphology representation, when evolving robot bodies and learning robot brains, has on the speed and degree of learning (Pigozzi et al., 2023a). Additionally, we enhance the generalization abilities of VSRs in unseen environmental conditions (in the form of damage) with the integration of Hebbian learning, paving the way for more robust and versatile artificial agents (Ferigo et al., 2022a).

Looking beyond VSRs, we propose a novel soft robot formalism consisting of a pressurized medium enveloped by a chain of particles and beams, blending local and global actuation. These robots are effective at shape-changing for various tasks

(Pigozzi, 2023a).

We believe this thesis marks a milestone at the intersection of ER and CI, setting the stage for a new era of exploration and innovation. By harnessing the power of collective systems, this research opens avenues for addressing previously insurmountable challenges in ER, thereby rejuvenating the realm of ER. As we venture forward, the integration of CI principles is poised to reshape the landscape of artificial intelligence, ushering in a future where adaptive and intelligent robotic systems are not only possible but also increasingly attainable.

7.1 Broader impact

Our results can be applied in ways that may have potential negative impacts on the broader society. While the experiments of this thesis consider self-contained simulated agents, future—albeit distant—agents may evolve harmful adaptation abilities. Examples of such abilities include manipulation, physical harm, and even warfare. We believe this scenario belongs to the world of science fiction for the foreseeable future, but we still feel obliged to include such a section.

7.2 Limitations

Every research has its limitations, and this thesis is no less. In particular, we are limited by the capabilities of the main simulators employed, which are 2D and thus do not faithfully render 3D robots that operate in reality. Real-life robots must also accomplish a variety of tasks in addition to locomoting. While the locomotion tasks considered differ by the type of terrain (flat, downhill, uphill, hilly) and the difficulty (random terrain generation for hilly), and Chapter 6 even added an escape from cage and an object carrying task, more tasks will enlarge the scope of this thesis and make the results more generalizable. Examples of additional tasks include phototaxis, food harvesting, multi-agent cooperation and competition, object manipulation, and many others (Bhatia et al., 2021).

Last but not least, this thesis deals exclusively with the simulation of VSRs. Albeit virtual creatures are noteworthy of study by themselves, as argued by Kriegman (2019) for different reasons, the lack of *in vivo* validation is another limitation of this work. Indeed, robotic systems notoriously suffer from the "sim-to-real" gap, the difficulty in porting simulated results to the real world (Mouret and Chatzilygeroudis, 2017). Such difficulty creates an asymmetry between the world *in silico* and the world *in vivo*: results we obtained might not replicate well with real robots, either quantitatively (e.g., effectiveness of a controller design) or qualitatively (e.g., comparisons between different encodings). This is especially true for soft robots, whose many degrees of freedom make their behavior challenging to model and predict (Rus and Tolley, 2015), although some recent directions seem promising (Kriegman et al., 2020b). We tweaked our simulations to more closely resemble the real-world, by adding noise to the sensor readings and introducing elements of randomness in every task (with the exception of locomotion on an even surface), but, eventually, the magnitude of the sim-to-real gap in our work remains yet to be quantified.

7.3 Future work

In the future, we will set out to fabricate VSRs and validate the simulated results of this thesis in the real world. This can possibly be achieved using soft voxel actuators of Kriegman et al. (2020b), which introduced a low cost, open source, and modular soft robot design and construction kit. New developments in stretchable electronics could enable sim-to-real transfer of the sensing (Liu et al., 2021), by patterning circuits that have light or contact sensors that would act as floor or neighbor sensors, while IMUs could calculate velocity.

Works Cited

Eric Aaron, Joshua Hawthorne-Madell, Ken Livingston, and John H. Long. Morphological Evolution: Bioinspired Methods for Analyzing Bioinspired Robots. *Frontiers in Robotics and AI*, 8, 2022.

Francesca Arese Lucini, Flaviano Morone, Maria Silvina Tomassone, and Hernán A Makse. Diversity increases the stability of ecosystems. *PloS one*, 15(4): e0228692, 2020.

Joshua E Auerbach and Josh C Bongard. Environmental influence on the evolution of morphological complexity in machines. *PLoS Comput Biol*, 10(1): e1003399, 2014.

Bruno B Averbeck and Vincent D Costa. Motivational neural circuits underlying reinforcement learning. *Nature Neuroscience*, 20(4):505–512, 2017.

Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. *Advances in Neural Information Processing Systems*, 29:4331–4339, 2016.

Jaume Baguna, E Saló, and Carme Auladell. Regeneration and pattern formation in planarians. iii. that neoblasts are totipotent stem cells and the cells. *Development*, 107(1):77–86, 1989.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

J Mark Baldwin. Organic selection. *Science*, 5(121):634–636, 1897.

James Mark Baldwin. A new factor in evolution. *American naturalist*, pages 536–553, 2018.

James Mark Baldwin et al. A new factor in evolution. *Diacronia*, (7):1–13, 2018.

Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Giovanni Squillero. Multi-level diversity promotion strategies for Grammar-guided Genetic Programming. *Applied Soft Computing*, 83:105599, 2019.

Randall D Beer, Hillel J Chiel, Roger D Quinn, Kenneth S Espenschied, and Patrik Larsson. A distributed neural network architecture for hexapod robot locomotion. *Neural Computation*, 4(3):356–365, 1992.

Randall D Beer et al. Toward the evolution of dynamical neural networks for minimally cognitive behavior. *From animals to animats*, 4:421–429, 1996.

Richard Bellman, Irving Glicksberg, and Oliver Gross. On the "bang-bang" control problem. *Quarterly of Applied Mathematics*, 14(1):11–18, 1956.

Fabien CY Benureau and Jun Tani. Morphological development at the evolutionary timescale: Robotic developmental evolution. *Artificial Life*, 28(1): 3–21, 2022.

Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies–a comprehensive introduction. *Natural computing*, 1(1):3–52, 2002.

Jagdeep Bhatia, Holly Jackson, Yunsheng Tian, Jie Xu, and Wojciech Matusik. Evolution gym: A large-scale benchmark for evolving soft robots. *Advances in Neural Information Processing Systems*, 34, 2021.

Douglas Blackiston, Emma Lederer, Sam Kriegman, Simon Garnier, Joshua Bongard, and Michael Levin. A cellular platform for the development of synthetic living machines. *Science Robotics*, 6(52):eabf1571, 2021.

Paul Bloom. *Descartes' Baby: How the Science of Child Development Explains What Makes Us Human.* 01 2004.

J. Bongard. Spontaneous evolution of structural modularity in robot neural network controllers: artificial life/robotics/evolvable hardware. In *GECCO '11*, 2011.

Josh Bongard, Victor Zykov, and Hod Lipson. Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121, 2006.

Josh C Bongard. Evolutionary robotics. *Communications of the ACM*, 56(8): 74–83, 2013.

Josh C. Bongard, Anton Bernatskiy, Kenneth R. Livingston, Nicholas Livingston, John H. Long Jr., and Marc L. Smith. Evolving robot morphology facilitates the evolution of neural modularity and evolvability. In Sara Silva and Anna Isabel Esparcia-Alcázar, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015*, pages 129–136. ACM, 2015. doi: 10.1145/2739480.2754750. URL `https://doi.org/10.1145/2739480.2754750`.

Jürgen Branke. Evolutionary algorithms for neural network design and liraining. 1995.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Rodney A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6(1):3–15, 1990. ISSN 0921-8890. doi: https://doi.org/10.1016/S0921-8890(05)80025-9. URL `https://www.sciencedirect.com/science/article/pii/S0921889005800259`. Designing Autonomous Agents.

Thomas H Brown, Edward W Kairiss, and Claude L Keenan. Hebbian synapses: biophysical mechanisms and algorithms. *Annual review of neuroscience*, 13(1): 475–511, 1990.

Edgar Buchanan, Léni K Le Goff, Wei Li, Emma Hart, Agoston E Eiben, Matteo De Carlo, Alan F Winfield, Matthew F Hale, Robert Woolley, Mike Angus, et al. Bootstrapping Artificial Evolution to Design Robots for Autonomous Fabrication. *Robotics*, 9(4):106, 2020.

Wilhelm Burger, Mark James Burge, Mark James Burge, and Mark James Burge. *Principles of digital image processing*, volume 111. Springer, 2009.

Zack Butler and Daniela Rus. Distributed planning and control for modular robots with unit-compressible modules. *The International Journal of Robotics Research*, 22(9):699–715, 2003.

Natalia Caporale and Yang Dan. Spike timing–dependent plasticity: a Hebbian learning rule. *Annu. Rev. Neurosci.*, 31:25–46, 2008.

C. Cappelle, A. Bernatskiy, K. Livingston, N. Livingston, and J. Bongard. Morphological modularity can enable the evolution of robot behavior to scale linearly with the number of environmental features. *Frontiers Robotics AI*, 3: 59, 2016.

Erin Catto. Box2d: A 2d physics engine for games. *URL: http://www. box2d. org*, 2011.

Yunus A Cengel, Michael A Boles, and Mehmet Kanoğlu. *Thermodynamics: an engineering approach*, volume 5. McGraw-hill New York, 2011.

Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath. An attentive survey of attention models. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 12(5):1–32, 2021.

Boyuan Chen, Robert Kwiatkowski, Carl Vondrick, and Hod Lipson. Fully body visual self-modeling of robot morphologies. *Science Robotics*, 7(68), 2022.

Nicholas Cheney, Jeff Clune, and Hod Lipson. Evolved electrophysiological soft robots. In *Artificial Life Conference Proceedings 14*, pages 222–229. MIT Press, 2014a.

Nick Cheney, Robert MacCurdy, Jeff Clune, and Hod Lipson. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 167–174. ACM, 2013.

Nick Cheney, Robert MacCurdy, Jeff Clune, and Hod Lipson. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. *ACM SIGEVOlution*, 7(1):11–23, 2014b.

Nick Cheney, Josh Bongard, and Hod Lipson. Evolving soft robots in tight spaces. In *Proceedings of the 2015 annual conference on Genetic and Evolutionary Computation*, pages 935–942. ACM, 2015.

Nick Cheney, Josh Bongard, Vytas SunSpiral, and Hod Lipson. Scalable co-optimization of morphology and control in embodied machines. *Journal of The Royal Society Interface*, 15(143):20170937, 2018.

Jinyoung Choi, Beom-Jin Lee, and Byoung-Tak Zhang. Multi-focus attention network for efficient deep reinforcement learning. In *Workshops at the thirty-first AAAI conference on artificial intelligence*, 2017.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. Back to basics: Benchmarking canonical evolution strategies for playing atari. *arXiv preprint arXiv:1802.08842*, 2018.

David Johan Christensen, Ulrik Pagh Schultz, and Kasper Stoy. A distributed and morphology-independent strategy for adaptive locomotion in self-reconfigurable modular robots. *Robotics and Autonomous Systems*, 61(9):1021–1035, 2013.

Émile Clapeyron. Mémoire sur la puissance motrice de la chaleur. *Journal de l'École polytechnique*, 14:153–190, 1834.

Jeff Clune, Benjamin E Beckmann, Philip K McKinley, and Charles Ofria. Investigating whether hyperneat produces modular neural networks. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 635–642, 2010.

Jeff Clune, Jean-Baptiste Mouret, and Hod Lipson. The evolutionary origins of modularity. *Proceedings. Biological sciences / The Royal Society*, 280: 20122863, 07 2013. doi: 10.1098/rspb.2012.2863.

Asher Cohen and Robert D Rafal. Attention and feature integration: Illusory conjunctions in a patient with a parietal lobe lesion. *Psychological science*, 2 (2):106–110, 1991.

Oliver J Coleman and Alan D Blair. Evolving plastic neural networks for online learning: review and future directions. In *Australasian Joint Conference on Artificial Intelligence*, pages 326–337. Springer, 2012.

Samuel Frazer Cooke and Timothy VP Bliss. Plasticity in the human central nervous system. *Brain*, 129(7):1659–1673, 2006.

James W Cooley and John W Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

Francesco Corucci, Nick Cheney, Francesco Giorgio-Serchi, Josh Bongard, and Cecilia Laschi. Evolving soft locomotion in aquatic and terrestrial environments: effects of material properties and environmental transitions. *Soft robotics*, 5(4):475–495, 2018.

Vincent D Costa, Olga Dal Monte, Daniel R Lucas, Elisabeth A Murray, and Bruno B Averbeck. Amygdala and ventral striatum make distinct contributions to reinforcement learning. *Neuron*, 92(2):505–517, 2016.

Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)*, 45(3):1–33, 2013.

Antoine Cully. Autonomous skill discovery with quality-diversity and unsupervised descriptors. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 81–89, 2019.

Antoine Cully and Yiannis Demiris. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22(2):245–259, 2017.

Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

Sylvain Cussat-Blanc, Kyle Harrington, and Wolfgang Banzhaf. Artificial gene regulatory networks—a review. *Artificial life*, 24(4):296–328, 2019.

Charles Darwin. On the origin of species, 1859, 1859.

Matteo De Carlo, Daan Zeeuwe, Eliseo Ferrante, Gerben Meynen, Jacintha Ellers, and AE Eiben. Influences of Artificial Speciation on Morphological Robot Evolution. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2272–2279. IEEE, 2020.

Matteo De Carlo, Eliseo Ferrante, Daan Zeeuwe, Jacintha Ellers, Gerben Meynen, and AE Eiben. Heritability in morphological robot evolution. *arXiv preprint arXiv:2110.11187*, 2021.

Dr. Hugo de Garis. Genetic programming - building artificial nervous systems with genetically programmed neural network modules. *Proceedings of the 7th International Conference on Machine Learning*, 08 1998.

Kenneth A De Jong. *Evolutionary Computation: A Unified Approach.* MIT Press, 2006.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Stephane Doncieux, Nicolas Bredeche, Jean-Baptiste Mouret, and Agoston E Eiben. Evolutionary robotics: what, why, and where to. *Frontiers in Robotics and AI*, 2:4, 2015.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Hubert L Dreyfus. Alchemy and artificial intelligence. Technical report, RAND CORP SANTA MONICA CA, 1965.

Jon Driver. A selective review of selective attention research from the past century. *British Journal of Psychology*, 92(1):53–78, 2001.

Dylan Drotman, Saurabh Jadhav, David Sharp, Christian Chan, and Michael T Tolley. Electronics-free pneumatic circuits for controlling soft-legged robots. *Science Robotics*, 6(51):eaay2627, 2021.

A. E. Eiben and Emma Hart. If It Evolves It Needs to Learn. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '20, page 1383–1384, New York, NY, USA, 2020a. Association for Computing Machinery. ISBN 9781450371278.

A.E. Eiben. Real-world robot evolution: Why would it (not) work? *Frontiers in Robotics and AI*, 8:243, 2021a. ISSN 2296-9144. doi: 10.3389/frobt.2021.696452. URL `https://www.frontiersin.org/article/10.3389/frobt.2021.696452`.

AE Eiben. Real-world robot evolution: Why would it (not) work? *Frontiers in Robotics and AI*, page 243, 2021b.

AE Eiben and Emma Hart. If it evolves it needs to learn. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pages 1383–1384, 2020b.

Agoston E Eiben and Jim Smith. From evolutionary computation to the evolution of things. *Nature*, 521(7553):476–482, 2015.

Aguston E Eiben, Nicolas Bredeche, Mark Hoogendoorn, Jürgen Stradner, Jon Timmis, Andy Tyrrell, and Alan Winfield. The triangle of life: Evolving robots in real-time and real-space. In *Proceedings of the European Conference on Artificial Life (ECAL-2013)*, pages 1–8, 2013.

Sami El-Boustani, Jacque PK Ip, Vincent Breton-Provencher, Graham W Knott, Hiroyuki Okuno, Haruhiko Bito, and Mriganka Sur. Locally coordinated synaptic plasticity of visual cortex neurons in vivo. *Science*, 360(6395):1349–1354, 2018.

Kai Olav Ellefsen, Jean-Baptiste Mouret, and J. Clune. Neural modularity helps organisms evolve to learn new skills without forgetting old skills. *PLoS Computational Biology*, 11, 2015.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Carlos Espinosa-Soto and Andreas Wagner. Specialization can drive the evolution of modularity. *PLOS Computational Biology*, 6(3):1–10, 03 2010. doi: 10.1371/journal.pcbi.1000719. URL `https://doi.org/10.1371/journal.pcbi.1000719`.

A. Faiña, F. Bellas, F. Orjales, D. Souto, and R. Duro. An evolution friendly modular architecture to produce feasible robots. *Robotics Auton. Syst.*, 63: 195–205, 2015.

Andres Faiña. Evolving modular robots: Challenges and opportunities. In *ALIFE 2021: The 2021 Conference on Artificial Life*. MIT Press, 2021.

Andrea Ferigo, Giovanni Iacca, and Eric Medvet. Beyond Body Shape and Brain: Evolving the Sensory Apparatus of Voxel-based Soft Robots. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 2021.

Andrea Ferigo, Giovanni Iacca, Eric Medvet, and Federico Pigozzi. Evolving hebbian learning rules in voxel-based soft robots. *IEEE Transactions on Cognitive and Developmental Systems*, pages 1–1, 2022a. doi: 10.1109/TCDS.2022.3226556.

Andrea Ferigo, LB Soros, Eric Medvet, and Giovanni Iacca. On the Entanglement between Evolvability and Fitness: an Experimental Study on Voxel-based Soft Robots. In *ALIFE 2022: The 2022 Conference on Artificial Life*. MIT Press, 2022b.

Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The journal of machine learning research*, 15(1):3133–3181, 2014.

Dario Floreano and Joseba Urzelai. Evolutionary robots with on-line self-organization and behavioral fitness. *Neural Networks*, 13(4-5):431–443, 2000.

Dario Floreano and Joseba Urzelai. Evolution of plastic control networks. *Autonomous robots*, 11(3):311–317, 2001.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

Steven E Franklin and Oumer S Ahmed. Deciduous tree species classification using object-based analysis and machine learning with unmanned aerial vehicle multispectral data. *International Journal of Remote Sensing*, 39(15-16):5236–5245, 2018.

R. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3:128–135, 1999.

Stefano Furlan, Eric Medvet, Giorgia Nadizar, and Federico Pigozzi. On the mutual influence of human and artificial life: an experimental investigation. In *Artificial Life Conference Proceedings 34*, volume 2022, page 14. MIT press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info ..., 2022.

Adam Gaier and David Ha. Weight agnostic neural networks. *Advances in neural information processing systems*, 32, 2019.

Andrea Galassi, Marco Lippi, and Paolo Torroni. Attention in natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Karunesh Ganguly and Mu-ming Poo. Activity-dependent neural plasticity from bench to bedside. *Neuron*, 80(3):729–741, 2013.

L. Gentile, F. Cebrià, and Kerstin Bartscherer. The planarian flatworm: an in vivo model for stem cell biology and nervous system regeneration. *Disease Models & Mechanisms*, 4:12 – 19, 2011.

John Gerhart and Marc Kirschner. The theory of facilitated variation. *Proceedings of the National Academy of Sciences*, 104(suppl 1):8582–8589, 2007.

James J Gibson. The theory of affordances. *Hilldale, USA*, 1(2):67–82, 1977.

Michelle Girvan and Mark Newman. Community structure in social and biological networks. *proc natl acad sci*, 99:7821–7826, 11 2001.

Léni K Le Goff and Emma Hart. On the challenges of jointly optimising robot morphology and control using a hierarchical optimisation scheme. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1498–1502, 2021.

O. Goldschmidt and D. Hochbaum. Polynomial algorithm for the k-cut problem. *[Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science*, pages 444–451, 1988.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

S J Gould and R C Lewontin. The spandrels of san marco and the panglossian paradigm: a critique of the adaptationist programme. *Proceedings of the Royal Society of London Series B, Biological Sciences*, pages 581–598, 1979.

Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Learning graph cellular automata. *Advances in Neural Information Processing Systems*, 34, 2021.

Djordje Grbic and Sebastian Risi. Safer Reinforcement Learning through Transferable Instinct Networks. In *ALIFE 2021: The 2021 Conference on Artificial Life*. MIT Press, 2021.

T Ryan Gregory, James A Nicol, Heidi Tamm, Bellis Kullman, Kaur Kullman, Ilia J Leitch, Brian G Murray, Donald F Kapraun, Johann Greilhuber, and Michael D Bennett. Eukaryotic genome size databases. *Nucleic acids research*, 35(suppl_1):D332–D338, 2007.

Roderich Groß, Elio Tuci, Marco Dorigo, Michael Bonani, and Francesco Mondada. Object transport by modular robots that self-assemble. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2558–2564. IEEE, 2006.

MD Guiry and GM Guiry. Algaebase, world-wide electron. publ., nat. univ. ireland, galway, 2018, 2021.

Agrim Gupta, Silvio Savarese, Surya Ganguli, and Li Fei-Fei. Embodied Intelligence via Learning and Evolution. *arXiv preprint arXiv:2102.02202*, 2021.

A.R. Gutai and T.E. Gorochowski. How biological concepts and evolutionary theories are inspiring advances in machine intelligence. *Preprints*, 2021. doi: 10.20944/preprints202109.0234.v1.

David Ha. Evolving stable strategies. *blog.otoro.net*, 2017. URL `http://blog.otoro.net/2017/11/12/evolving-stable-strategies/`.

David Ha. Reinforcement learning for improving agent design. *Artificial life*, 25(4):352–365, 2019.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

David Ha and Yujin Tang. Collective intelligence for deep learning: A survey of recent developments. *arXiv preprint arXiv:2111.14377*, 2021.

David Ha and Yujin Tang. Collective intelligence for deep learning: A survey of recent developments. *Collective Intelligence*, 1(1):26339137221114874, 2022.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. In *International Conference on Learning Representations*, 2017.

Sehoon Ha, Peng Xu, Zhenyu Tan, Sergey Levine, and Jie Tan. Learning to walk in the real world with minimal human effort. In *Conference on Robot Learning*, pages 1110–1120. PMLR, 2021.

M Hale, Edgar Buchanan Berumen, Alan Winfield, Jon Timmis, Emma Hart, Gusz Eiben, Wei Li, and Andy Tyrrell. The are robot fabricator: How to (re) produce robots that can evolve in the real world. In *International Society for Artificial Life: ALIFE2019*, pages 95–102. York, 2019a.

Matthew Hale, Edgar Buchanan, Alan Winfield, Jon Timmis, Emma Hart, A. Eiben, Mike Angus, Frank Veenstra, Wei Li, Robert Woolley, Matteo De Carlo, and Andy Tyrrell. The are robot fabricator: How to (re)produce robots that can evolve in the real world. In *International Society for Artificial Life: ALIFE2019*, pages 95–102, 01 2019b. doi: 10.1162/isal_a_00147.

Ahmed Hallawa, Thorsten Born, Anke Schmeink, Guido Dartmann, Arne Peine, Lukas Martin, Giovanni Iacca, AE Eiben, and Gerd Ascheid. Evo-RL: evolutionary-driven reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 153–154, 2021.

Nikolaus Hansen. The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation*, pages 75–102. Springer, 2006.

Nikolaus Hansen. The CMA Evolution Strategy: A Tutorial, 2016.

Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of IEEE international conference on evolutionary computation*, pages 312–317. IEEE, 1996.

Nikolaus Hansen and Andreas Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2):159–195, 2001. URL `https://doi.org/10.1162/106365601750190398`.

Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. Cma-es/pycma on github. *Zenodo, doi*, 10, 2019.

Emma Hart and Léni K Le Goff. Artificial evolution of robot bodies and control: on the interaction between evolution, learning and culture. *Philosophical Transactions of the Royal Society B*, 377(1843):20210117, 2022.

Helmut Hauser, Auke J Ijspeert, Rudolf M Füchslin, Rolf Pfeifer, and Wolfgang Maass. Towards a theoretical foundation for morphological computation with compliant bodies. *Biological cybernetics*, 105(5):355–370, 2011.

Helmut Hauser, Thrishantha Nanayakkara, and Fulvio Forni. Leveraging morphological computation for controlling soft robots: Learning from nature to control soft robots. *IEEE Control Systems Magazine*, 43(3):114–129, 2023.

Elliot W Hawkes, Laura H Blumenschein, Joseph D Greer, and Allison M Okamura. A soft robot that navigates its environment through growth. *Science Robotics*, 2(8):eaan3028, 2017.

Donald Olding Hebb. *The organization of behavior: A neuropsychological theory.* Psychology Press, 2005.

Jonathan Hiller and Hod Lipson. Automatic design and manufacture of soft robots. *IEEE Transactions on Robotics*, 28(2):457–466, 2011.

Jonathan Hiller and Hod Lipson. Automatic design and manufacture of soft robots. *IEEE Transactions on Robotics*, 28(2):457–466, 2012.

Jonathan Hiller and Hod Lipson. Dynamic Simulation of Soft Multimaterial 3D-printed Objects. *Soft Robotics*, 1(1):88–101, 2014.

Geoffrey E Hinton, Steven J Nowlan, et al. How learning can guide evolution. *Complex systems*, 1(3):495–502, 1987.

Geoffrey E Hinton, Steven J Nowlan, et al. How learning can guide evolution. *Adaptive individuals in evolving populations: models and algorithms*, 26:447–454, 1996.

Binyamin Hochner. An embodied view of octopus neurobiology. *Current biology*, 22(20):R887–R892, 2012.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. *J. Mach. Learn. Res.*, 22(241):1–124, 2021.

Kazuya Horibe, Kathryn Walker, and Sebastian Risi. Regenerating soft robots through neural cellular automata. In *EuroGP*, pages 36–50, 2021.

Gregory S Hornby, Jordan B Pollack, et al. Body-brain co-evolution using L-systems as a generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 868–875, 2001.

Toby Howison, Simon Hauser, Josie Hughes, and Fumiya Iida. Reality-assisted evolution of soft robots through large-scale physical experimentation: a review. *Artificial Life*, 26(4):484–506, 2020.

Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pages 4455–4464. PMLR, 2020.

Matthew Hutson. Has artificial intelligence become alchemy?, 2018.

Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural networks*, 21(4):642–653, 2008.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Milan Jelisavcic, Kyrre Glette, Evert Haasdijk, and AE Eiben. Lamarckian evolution of simulated modular robots. *Frontiers in Robotics and AI*, 6:9, 2019.

Michał Joachimczak, Reiji Suzuki, and Takaya Arita. Artificial Metamorphosis: Evolutionary Design of Transforming, Soft-Bodied Robots. *Artificial Life*, 22 (3):271–298, 2016.

Brian R Johnson and S. Lam. Self-organization, natural selection, and evolution: Cellular hardware and genetic software. 2010.

Jakob Jordan, Maximilian Schmidt, Walter Senn, and Mihai A Petrovici. Evolving interpretable plasticity for spiking networks. *Elife*, 10, 2021.

A. Joven, A. Elewa, and A. Simon. Model systems for regeneration: salamanders. *Development*, 146, 2019.

Daniel Kahneman. Thinking fast and slow. *New York: Farrar, Straus and Giroux*, 2011.

Shadi Tasdighi Kalat, Siamak G Faal, and Cagdas D Onal. A decentralized, communication-free force distribution method with application to collective object manipulation. *Journal of Dynamic Systems, Measurement, and Control*, 140(9):091012, 2018.

A. Kamimura, H. Kurokawa, E. Yoshida, K. Tomita, S. Kokaji, and S. Murata. Distributed adaptive locomotion by a modular robotic system, m-tran ii. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*

(IROS) (IEEE Cat. No.04CH37566), volume 3, pages 2370–2377 vol.3, 2004. doi: 10.1109/IROS.2004.1389763.

Akiya Kamimura, Haruhisa Kurokawa, E Toshida, Kohji Tomita, Satoshi Murata, and Shigeru Kokaji. Automatic locomotion pattern generation for modular robots. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 1, pages 714–720. IEEE, 2003.

DP Kelly and MJ Natale. Neurodevelopmental and executive function and dysfunction. *Nelson Textbook of Pediatrics. 21st ed. Philadelphia, PA: Elsevier*, 2020.

Scott A Kelly, Tami M Panhuis, and Andrew M Stoehr. Phenotypic plasticity: molecular mechanisms and adaptive significance. *Compr Physiol*, 2(2):1417–1439, 2012.

Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *arXiv preprint arXiv:2101.01169*, 2021.

M. Kimura. The neutral theory of molecular evolution. *Scientific American*, 241 5:98–100, 102, 108 passim, 1979.

Hiroaki Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Syst.*, 4, 1990.

Julian Kiverstein and Matt Sims. Is free-energy minimisation the mark of the cognitive? *Biology & Philosophy*, 36(2):25, 2021.

Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. Empowerment: A universal agent-centric measure of control. In *2005 ieee congress on evolutionary computation*, volume 1, pages 128–135. IEEE, 2005.

Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail`, 2018.

J. Koza. Genetic programming - on the programming of computers by means of natural selection. In *Complex adaptive systems*, 1993.

Sam Kriegman. Why virtual creatures matter. *Nature Machine Intelligence*, 1 (10):492–492, 2019.

Sam Kriegman, Nick Cheney, and Josh Bongard. How morphological development can guide evolution. *Scientific reports*, 8(1):13934, 2018.

Sam Kriegman, Stephanie Walker, Dylan Shah, Michael Levin, Rebecca Kramer-Bottiglio, and Josh Bongard. Automated shapeshifting for function recovery in damaged robots. In *15th Robotics: Science and Systems, RSS 2019.* MIT Press Journals, 2019.

Sam Kriegman, Douglas Blackiston, Michael Levin, and Josh Bongard. A scalable pipeline for designing reconfigurable organisms. *Proceedings of the National Academy of Sciences*, 117(4):1853–1859, 2020a.

Sam Kriegman, Amir Mohammadi Nasab, Dylan Shah, Hannah Steele, Gabrielle Branin, Michael Levin, Josh Bongard, and Rebecca Kramer-Bottiglio. Scalable sim-to-real transfer of soft robot designs. *2020 3rd IEEE International Conference on Soft Robotics, RoboSoft 2020*, pages 359–366, may 2020b. doi: 10.1109/ROBOSOFT48309.2020.9116004.

Sam Kriegman, Amir Mohammadi Nasab, Dylan Shah, Hannah Steele, Gabrielle Branin, Michael Levin, Josh Bongard, and Rebecca Kramer-Bottiglio. Scalable sim-to-real transfer of soft robot designs. In *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*, pages 359–366, 2020c. doi: 10.1109/RoboSoft48309.2020.9116004.

Sam Kriegman, Amir Mohammadi Nasab, Douglas Blackiston, Hannah Steele, Michael Levin, Rebecca Kramer-Bottiglio, and Josh Bongard. Scale invariant robot behavior with fractals. *arXiv preprint arXiv:2103.04876*, 2021.

Gongjin Lan, Matteo De Carlo, Fuda van Diggelen, Jakub M Tomczak, Diederik M Roijers, and Agoston E Eiben. Learning directed locomotion in modular robots with evolvable morphologies. *Applied Soft Computing*, 111:107688, 2021.

Cecilia Laschi, Barbara Mazzolai, and Matteo Cianchetti. Soft robotics: Technologies and systems pushing the boundaries of robot abilities. *Science Robotics*, 1(1):eaah3690, 2016.

John Lawrence. *Arm loss and regeneration in Asteroidea (Echinodermata)*, pages 39–52. 07 2020. ISBN 9781003077572. doi: 10.1201/9781003077572-7.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.

Julie Legrand, Seppe Terryn, Ellen Roels, and Bram Vanderborght. Reconfigurable, multi-material, voxel-based soft robots. *IEEE Robotics and Automation Letters*, 2023.

Joel Lehman and Kenneth O Stanley. Exploiting Open-Endedness to Solve Problems Through the Search for Novelty. volume 11, page 329, 2008.

Joel Lehman and Kenneth O Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 211–218, 2011.

Joel Lehman, Jeff Clune, Dusan Misevic, Christoph Adami, Lee Altenberg, Julie Beaulieu, Peter J Bentley, Samuel Bernard, Guillaume Beslon, David M

Bryson, et al. The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. *Artificial life*, 26(2):274–306, 2020.

Jakub Limanowski and Felix Blankenburg. Minimal self-models and the free energy principle. *Frontiers in human neuroscience*, 7:547, 2013.

Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *AI Open*, 2022.

Bruce G Lindsay. Mixture models: theory, geometry and applications. In *NSF-CBMS regional conference series in probability and statistics*, pages i–163. JSTOR, 1995.

Hod Lipson and Jordan B Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974–978, 2000.

Hod Lipson, Vytas Sunspiral, Josh Bongard, and Nicholas Cheney. On the difficulty of co-optimizing morphology and control in evolved virtual creatures. In *Artificial Life Conference Proceedings 13*, pages 226–233. MIT Press, 2016.

Jinguo Liu, Xin Zhang, and G. Hao. Survey on research and development of reconfigurable modular robots. *Advances in Mechanical Engineering*, 8, 2016.

Shanliangzi Liu, Dylan S. Shah, and Rebecca Kramer-Bottiglio. Highly stretchable multilayer electronic circuits using biphasic gallium-indium. *Nature Materials*, pages 1–8, feb 2021. ISSN 14764660. doi: 10.1038/s41563-021-00921-8. URL `https://doi.org/10.1038/s41563-021-00921-8`.

Sida Liu, David Matthews, Sam Kriegman, and Josh Bongard. Voxcraft-sim, a gpu-accelerated voxel-based physics engine. `https://github.com/voxcraft/voxcraft-sim`, 2020. URL `https://github.com/voxcraft/voxcraft-sim`.

Stuart Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

E. López, R. Poli, A. Kattan, M. O'Neill, and A. Brabazon. Neutrality in evolutionary algorithms... what do we know? *Evolving Systems*, 2:145–163, 2011.

Sean Luke. *Essentials of metaheuristics.* 2009.

Jie Luo, Aart Stuurman, Jakub M Tomczak, Jacintha Ellers, and Agoston E Eiben. The effects of learning in morphologically evolving robot systems. *arXiv preprint arXiv:2111.09851*, 2021.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

Anne E Magurran. *Measuring biological diversity.* John Wiley & Sons, 2013.

Horia Mania, Aurelia Guy, and Benjamin Recht. Simple linear search of static policies is competitive for reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

H. B. Mann and D. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18: 50–60, 1947.

Georg Martius, Ralf Der, and Nihat Ay. Information driven self-organization of complex robotic behaviors. *PloS one*, 8(5):e63400, 2013.

Peter V Massey and Zafar I Bashir. Long-term depression: multiple forms and implications for brain function. *Trends in neurosciences*, 30(4):176–184, 2007.

Samuel Ottmar Mast. Habits and reactions of the ciliate, lacrymaria. *Journal of Animal Behavior*, 1(4):229, 1911.

David Matthews, Andrew Spielberg, Rus Daniela, Sam Kriegman, and Josh Bongard. Efficient automatic design of robots. *Proceedings of the National Academy of Sciences*, 2023.

Claudio Mattiussi and Dario Floreano. Analog genetic encoding for the evolution of circuits and networks. *IEEE Transactions on evolutionary computation*, 11(5):596–607, 2007.

Maciej Matyka and Mark Ollila. Pressure model of soft body simulation. In *The Annual SIGRAD Conference. Special Theme-Real-Time Simulations. Conference Proceedings from SIGRAD2003*, number 010, pages 29–33. Citeseer, 2003.

RL Mayden. A hierarchy of species concepts: the denouement of the specie problem. *The Units of Biodiversity Species in Practice.(eds MF Claridge, HA*, 1997.

Giles Mayley. Landscapes, learning costs, and genetic assimilation. *Evolutionary Computation*, 4(3):213–234, 1996.

Alan McIntyre, Matt Kallada, Cesar G. Miguel, and Carolina Feher de Silva. neat-python.

Eric Medvet, Alberto Bartoli, Andrea De Lorenzo, and Giulio Fidel. Evolution of distributed neural controllers for voxel-based soft robots. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 112–120, 2020a.

Eric Medvet, Alberto Bartoli, Andrea De Lorenzo, and Stefano Seriani. 2D-VSR-Sim: A simulation tool for the optimization of 2-D voxel-based soft robots. *SoftwareX*, 12, 2020b.

Eric Medvet, Alberto Bartoli, Federico Pigozzi, and Marco Rochelli. Biodiversity in evolved voxel-based soft robots. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 129–137, 2021.

Eric Medvet, Giorgia Nadizar, and Federico Pigozzi. On the impact of body material properties on neuroevolution for embodied agents: the case of voxel-based soft robots. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 2122–2130, 2022.

Thomas Miconi, Kenneth Stanley, and Jeff Clune. Differentiable plasticity: training plastic neural networks with backpropagation. In *International Conference on Machine Learning*, pages 3559–3568. PMLR, 2018.

Nicola Milano and Stefano Nolfi. Phenotypic complexity and evolvability in evolving robots. *Frontiers in Robotics and AI*, 9:994485, 2022.

Julian Francis Miller and Simon L Harding. Cartesian genetic programming. In *Proceedings of the 10th annual conference companion on Genetic and evolutionary computation*, pages 2701–2726, 2008.

Karine Miras, Matteo De Carlo, Sayfeddine Akhatou, and AE Eiben. Evolving-controllers versus learning-controllers for morphologically evolvable robots. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pages 86–99. Springer, 2020a.

Karine Miras, Eliseo Ferrante, and AE Eiben. Environmental influences on evolvable robots. *PloS one*, 15(5):e0233848, 2020b.

Melanie Mitchell. Why ai is harder than we think. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '21, page 3, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383509. doi: 10.1145/3449639.3465421. URL https://doi.org/10.1145/3449639.3465421.

Tom M Mitchell. *The need for biases in learning generalizations.* Department of Computer Science, Laboratory for Computer Science Research . . . , 1980.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

David S Moore. *The Dependent Gene: The Fallacy of "Nature Vs. Nurture"*. Macmillan, 2003.

Alexander Mordvintsev, Ettore Randazzo, Eyvind Niklasson, and Michael Levin. Growing neural cellular automata. *Distill*, 2020. doi: 10.23915/distill.00023. https://distill.pub/2020/growing-ca.

Rodrigo Moreno, Frank Veenstra, David Silvera, Julian Franco, Oscar Gracia, Ernesto Cordoba, Jonatan Gomez, and Andres Faina. Automated reconfiguration of modular robots using robot manipulators. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 884–891, 2018. doi: 10.1109/SSCI.2018.8628628.

Tohru Moriyama and Yukio-Pegio Gunji. Autonomous learning in maze solution by octopus. *Ethology*, 103(6):499–513, 1997.

J-B Mouret and Stéphane Doncieux. Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evolutionary computation*, 20(1):91–133, 2012.

Jean-Baptiste Mouret and Konstantinos Chatzilygeroudis. 20 years of reality gap: a few thoughts about simulators in evolutionary robotics. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1121–1124, 2017.

Jean-Baptiste Mouret and S. Doncieux. Evolving modular neural-networks through exaptation. *2009 IEEE Congress on Evolutionary Computation*, pages 1570–1577, 2009.

Nils Müller and Tobias Glasmachers. Challenges in high-dimensional reinforcement learning with evolution strategies. In *International Conference on Parallel Problem Solving from Nature*, pages 411–423. Springer, 2018.

Elias Najarro and Sebastian Risi. Meta-learning through Hebbian plasticity in random networks. *Advances in Neural Information Processing Systems*, 33: 20719–20731, 2020.

Kohei Nakajima, Helmut Hauser, Tao Li, and Rolf Pfeifer. Information processing via physical soft body. *Scientific reports*, 5(1):1–11, 2015.

Emre O Neftci and Bruno B Averbeck. Reinforcement learning in artificial and biological systems. *Nature Machine Intelligence*, 1(3):133–143, 2019.

John Von Neumann and Arthur W. Burks. *Theory of Self-Reproducing Automata*. University of Illinois Press, USA, 1966.

Scott Niekum, Andrew G Barto, and Lee Spector. Genetic programming for reward function search. *IEEE Transactions on Autonomous Mental Development*, 2(2):83–90, 2010.

Geoff Nitschke and David Howard. AutoFac: The Perpetual Robot Machine. *IEEE Transactions on Artificial Intelligence*, 2021.

Yael Niv, Daphna Joel, Isaac Meilijson, and Eytan Ruppin. Evolution of reinforcement learning in uncertain environments: Emergence of risk-aversion and matching. In *European Conference on Artificial Life*, pages 252–261. Springer, 2001.

Stefano Nolfi. *Behavioral and Cognitive Robotics: An adaptive perspective.* Stefano Nolfi, 2021.

Stefano Nolfi and Dario Floreano. *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines.* MIT press, 2000.

Stefano Nolfi and Domenico Parisi. Learning to adapt to changing environments in evolving neural networks. *Adaptive behavior*, 5(1):75–98, 1996.

Jørgen Nordmoen, Frank Veenstra, Kai Olav Ellefsen, and Kyrre Glette. MAP-Elites enables Powerful Stepping Stones and Diversity for Modular Robotics. *Frontiers in Robotics and AI*, 8, 2021.

Tønnes F Nygaard, Charles P Martin, Eivind Samuelsen, Jim Torresen, and Kyrre Glette. Real-world evolution adapts robot morphology and control to hardware limitations. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 125–132, 2018.

OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. 2019. URL `https://arxiv.org/abs/1912.06680`.

Arthur O'Sullivan and Steven M Sheffrin. Economics: Principles in action. Upper Saddle River, New Jersey 07458: Pearson Prentice Hall. 2003.

Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

Dai Owaki, Shun-ya Horikiri, Jun Nishii, and Akio Ishiguro. Tegotae-based control produces adaptive inter-and intra-limb coordination in bipedal walking. *Frontiers in neurorobotics*, 15:47, 2021.

Paolo Pagliuca and Stefano Nolfi. The Dynamic of Body and Brain Co-Evolution. *arXiv preprint arXiv:2011.11440*, 2020.

Giuseppe Paolo, Alban Laflaquiere, Alexandre Coninx, and Stephane Doncieux. Unsupervised learning and exploration of reachable outcome space. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2379–2385. IEEE, 2020.

Deepak Pathak, Chris Lu, Trevor Darrell, Phillip Isola, and Alexei A Efros. Learning to control self-assembling morphologies: a study of generalization via modularity. *arXiv preprint arXiv:1902.05546*, 2019.

Anna R Patten, Suk Yu Yau, Christine J Fontaine, Alicia Meconi, Ryan C Wortman, and Brian R Christie. The benefits of exercise on structural and functional plasticity in the rodent hippocampus of different disease models. *Brain Plasticity*, 1(1):97–127, 2015.

Chandana Paul. Morphological computation: A basis for the analysis of morphology and control requirements. *Robotics and Autonomous Systems*, 54(8): 619–630, 2006.

Joachim Winther Pedersen and Sebastian Risi. Evolving and merging hebbian learning rules: increasing generalization by decreasing the number of rules. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 892–900, 2021.

Jaume Pellicer and Ilia J Leitch. The plant dna c-values database (release 7.1): an updated online repository of plant genome size data for comparative studies. *[" New Phytologist"]*, 2019.

Rolf Pfeifer and Josh Bongard. *How the body shapes the way we think: a new view of intelligence.* MIT press, 2006.

Federico Pigozzi. Robots: the century past and the century ahead, 2022a.

Federico Pigozzi. Shape Change and Control of Pressure-based Soft Agents. In *ALIFE 2022: The 2022 Conference on Artificial Life*, 07 2022b. doi: 10.1162/isal_a_00520. URL `https://doi.org/10.1162/isal_a_00520`. 37.

Federico Pigozzi. Pressure-based soft agents. *Artificial life*, 2023a.

Federico Pigozzi. Of typewriters and pcs. In *ALIFE 2023: Ghost in the Machine: Proceedings of the 2023 Artificial Life Conference.* MIT Press, 2023b.

Federico Pigozzi and Eric Medvet. Evolving modularity in soft robots through an embodied and self-organizing neural controller. *Artificial Life*, 2022.

Federico Pigozzi, Eric Medvet, and Laura Nenzi. Mining road traffic rules with signal temporal logic and grammar-based genetic programming. *Applied Sciences*, 11(22):10573, 2021.

Federico Pigozzi, Yujin Tang, Eric Medvet, and David Ha. Evolving Modular Soft Robots without Explicit Inter-Module Communication using Local Self-Attention. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '22, page 148–157, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392372. doi: 10.1145/3512290.3528762. URL `https://doi.org/10.1145/3512290.3528762`.

Federico Pigozzi, Federico Julian Camerota Verdù, and Eric Medvet. How the morphology encoding influences the learning ability in body-brain co-optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1045–1054, 2023a.

Federico Pigozzi, Eric Medvet, Alberto Bartoli, and Marco Rochelli. Factors impacting diversity and effectiveness of evolved modular robots. *ACM Transactions on Evolutionary Learning*, 3(1):1–33, 2023b.

Federico Pigozzi, Stephanie Woodman, Eric Medvet, Rebecca Kramer-Bottiglio, and Josh Bongard. Morphology choice affects the evolution of affordance detection in robots. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 211–219, 2023c.

Sidney Pontes-Filho and Stefano Nichele. Towards a framework for the evolution of artificial general intelligence. *arXiv:1903.10410*, 2019.

Manu Prakash and Matthew Bull. Mobile defects born from an energy cascade shape the locomotive behavior of a headless animal. In *APS March Meeting Abstracts*, volume 2022, pages D03–009, 2022.

Trevor D Price, Anna Qvarnström, and Darren E Irwin. The role of phenotypic plasticity in driving genetic evolution. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 270(1523):1433–1440, 2003.

Jeremy Pritchard. Turgor pressure. 2001.

Xinghua Qu, Zhu Sun, Yew-Soon Ong, Abhishek Gupta, and Pengfei Wei. Minimalistic attacks: How little it takes to fool deep reinforcement learning policies. *IEEE Transactions on Cognitive and Developmental Systems*, 13(4): 806–817, 2020.

J Pena Queralta, Cassandra McCord, Tuan Nguyen Gia, Hannu Tenhunen, and Tomi Westerlund. Communication-free and index-free distributed formation control algorithm for multi-robot systems. *Procedia Computer Science*, 151: 431–438, 2019.

A Rahimi. Machine learning has become alchemy. In *Thirsty-first Conference on Neural Information Processing Systems*, 2017.

John Rieffel, Francisco Valero-Cuevas, and Hod Lipson. Automated discovery and optimization of large irregular tensegrity structures. *Computers & Structures*, 87(5-6):368–379, 2009.

Sebastian Risi and Kenneth O Stanley. Indirectly encoding neural plasticity as a pattern of local rules. In *International Conference on Simulation of Adaptive Behavior*, pages 533–543. Springer, 2010.

Sebastian Risi and Kenneth O Stanley. Deep neuroevolution of recurrent and discrete world models. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 456–462, 2019.

Franz Rothlauf. On the locality of representations. *None*, 2003.

Franz Rothlauf. Representations for genetic and evolutionary algorithms. In *Representations for Genetic and Evolutionary Algorithms*, pages 9–32. Springer, 2006.

Franz Rothlauf and David E Goldberg. Redundant representations in evolutionary computation. *Evolutionary Computation*, 11(4):381–415, 2003.

Stephane Roy and Samuel Gatien. Regeneration in axolotls: A model to aim for! *Experimental gerontology*, 43:968–73, 10 2008. doi: 10.1016/j.exger.2008.09.003.

David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning internal representations by back-propagating errors. *Nature*, 323(99):533–536, 1986.

Daniela Rus and Michael T Tolley. Design, fabrication and control of soft robots. *Nature*, 521(7553):467, 2015.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Erica Salvato, Gianfranco Fenu, Eric Medvet, and Felice Andrea Pellegrino. Crossing the Reality Gap: a Survey on Sim-to-Real Transferability of Robot Controllers in Reinforcement Learning. *IEEE Access*, 2021.

Eivind Samuelsen and Kyrre Glette. Some distance measures for morphological diversification in generative evolutionary robotics. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 721–728, 2014.

Hiroki Sayama. *Introduction to the modeling and analysis of complex systems.* Open SUNY Textbooks, 2015.

Anton Maximilian Schäfer and Hans Georg Zimmermann. Recurrent neural networks are universal approximators. In *International Conference on Artificial Neural Networks*, pages 632–640. Springer, 2006.

Jürgen Schmidhuber. Reducing the ratio between learning complexity and number of time varying variables in fully recurrent nets. In *International Conference on Artificial Neural Networks*, pages 460–463. Springer, 1993a.

Jürgen Schmidhuber. A 'self-referential'weight matrix. In *International Conference on Artificial Neural Networks*, pages 446–450. Springer, 1993b.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Franziska Schrodt, Joseph J Bailey, W Daniel Kissling, Kenneth F Rijsdijk, Arie C Seijmonsbergen, Derk Van Ree, Jan Hjort, Russell S Lawley, Christopher N Williams, Mark G Anderson, et al. Opinion: To advance sustainable stewardship, we must document not only biodiversity but geodiversity. *Proceedings of the National Academy of Sciences*, 116(33):16155–16158, 2019.

Jacob Schrum and Risto Miikkulainen. Evolving multimodal behavior with modular neural networks in ms. pac-man. *GECCO 2014 - Proceedings of the 2014 Genetic and Evolutionary Computation Conference*, 07 2014. doi: 10.1145/2576768.2598234.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Hans-Paul Schwefel. Cybernetic evolution as strategy for experimental research in fluid mechanics. *Master's Thesis, Technical University of Berlin*, 1965.

Andrew Seary and William Richards. Partitioning networks by eigenvectors. *Proc. Int. Conf. on Social Networks*, 1, 01 1996.

Idan Segev and Elad Schneidman. Axons as computing devices: Basic insights gained from models. *Journal of Physiology-Paris*, 93:263–270, 1999.

Dylan Shah, Bilige Yang, Sam Kriegman, Michael Levin, Josh Bongard, and Rebecca Kramer-Bottiglio. Shape changing robots: bioinspiration, simulation, and physical realization. *Advanced Materials*, 33(19):2002882, 2021a.

Dylan S Shah, Joshua P Powers, Liana G Tilton, Sam Kriegman, Josh Bongard, and Rebecca Kramer-Bottiglio. A soft robot that adapts to environments through shape change. *Nature Machine Intelligence*, 3(1):51–59, 2021b.

Lawrence Shapiro. *Embodied cognition.* Routledge, 2019.

Asheesh Sharma, Sabine Hauert, and Helmut Hauser. Morphological communication for swarms. In *Artificial Life Conference Proceedings 32*, pages 549–557. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , 2020.

Christopher Ariel Shaw, Jill C McEachern, and Jill McEachern. *Toward a theory of neuroplasticity.* Psychology Press, 2001.

Virginia A Shepherd. The cytomatrix as a cooperative system of macromolecular and water networks. *Current topics in developmental biology*, 75:171–223, 2006.

Bruno Siciliano, Oussama Khatib, and Torsten Kröger. *Springer handbook of robotics*, volume 200. Springer, 2008.

Fernando Silva, Miguel Duarte, Luís Correia, Sancho Moura Oliveira, and Anders Lyhne Christensen. Open issues in evolutionary robotics. *Evolutionary computation*, 24(2):205–236, 2016.

S. Silva and E. Costa. Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. *Genetic Programming and Evolvable Machines*, 10:141–179, 2008.

Edward H Simpson. Measurement of diversity. *Nature*, 163(4148):688–688, 1949.

Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM, 1994.

Paul Singleton et al. *Bacteria in biology, biotechnology and medicine.* Number Ed. 6. John Wiley & Sons, 2004.

Moshe Sipper, Eduardo Sanchez, Daniel Mange, Marco Tomassini, Andrés Pérez-Uribe, and André Stauffer. A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems. *IEEE Transactions on Evolutionary Computation*, 1(1):83–97, 1997.

Andrew C Slocum, Douglas C Downey, and Randall D Beer. Further experiments in the evolution of minimally cognitive behavior: From perceiving affordances to selective attention. In *From animals to animats 6: Proceedings of the sixth international conference on simulation of adaptive behavior*, pages 430–439. Citeseer, 2000.

Tom Smith, Andrew Philippides, Phil Husbands, and Michael O'Shea. Neutrality and ruggedness in robot landscapes. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, pages 1348–1353. IEEE, 2002.

Andrea Soltoggio, Peter Durr, Claudio Mattiussi, and Dario Floreano. Evolving neuromodulatory topologies for reinforcement learning-like problems. In *2007 IEEE Congress on Evolutionary Computation*, pages 2471–2478. IEEE, 2007.

Andrea Soltoggio, Kenneth O Stanley, and Sebastian Risi. Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks. *Neural Networks*, 108:48–67, 2018.

Giovanni Squillero and Alberto Tonda. Divergence of character and premature convergence: A survey of methodologies for promoting diversity in evolutionary optimization. *Information Sciences*, 329:782–799, 2016.

Giovanni Squillero and Alberto Tonda. Promoting diversity in evolutionary optimization: why and how. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 998–1016, 2018.

Kenneth Stanley, David D'Ambrosio, and Jason Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial life*, 15:185–212, 02 2009. doi: 10.1162/artl.2009.15.2.15202.

Kenneth O Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2): 131–162, 2007.

Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

Erik Steltz, Annan Mozeika, Nick Rodenberg, Eric Brown, and Heinrich M Jaeger. Jsel: Jamming skin enabled locomotion. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5672–5677. IEEE, 2009.

Thomas A Stoffregen. Affordances as properties of the animal-environment system. In *How shall affordances be refined? Four perspectives*, pages 115–134. Routledge, 2018.

Jörg Stork, Martin Zaefferer, Nils Eisler, Patrick Tichelmann, Thomas Bartz-Beielstein, and AE Eiben. Behavior-based neuroevolutionary training in reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1753–1761, 2021.

F. Such, Vashisht Madhavan, Edoardo Conti, J. Lehman, Kenneth O. Stanley, and J. Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *ArXiv*, abs/1712.06567, 2017a.

Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017b.

Shyam Sudhakaran, Djordje Grbic, Siyan Li, Adam Katona, Elias Najarro, Claire Glanois, and Sebastian Risi. Growing 3d artefacts and functional machines with neural cellular automata. *arXiv preprint arXiv:2103.08737*, 2021.

Xin Sui, Hegao Cai, Dongyang Bie, Yu Zhang, Jie Zhao, and Yanhe Zhu. Automatic generation of locomotion patterns for soft modular reconfigurable robots. *Applied Sciences*, 10(1):294, 2020.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Michael A Tabak, Mohammad S Norouzzadeh, David W Wolfson, Steven J Sweeney, Kurt C VerCauteren, Nathan P Snow, Joseph M Halseth, Paul A Di Salvo, Jesse S Lewis, Michael D White, et al. Machine learning to classify animal species in camera trap images: Applications in ecology. *Methods in Ecology and Evolution*, 10(4):585–590, 2019.

Jacopo Talamini, Eric Medvet, Alberto Bartoli, and Andrea De Lorenzo. Evolutionary Synthesis of Sensing Controllers for Voxel-based Soft Robots. In *Artificial Life Conference Proceedings*, pages 574–581. MIT Press, 2019.

Jacopo Talamini, Eric Medvet, and Stefano Nichele. Criticality-Driven Evolution of Adaptable Morphologies of Voxel-Based Soft-Robots. *Frontiers in Robotics and AI*, 8:172, 2021.

Yujin Tang and David Ha. The sensory neuron as a transformer: Permutation-invariant neural networks for reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Yujin Tang, Duong Nguyen, and David Ha. Neuroevolution of self-interpretable agents. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 414–424, 2020.

Danesh Tarapore, Jeff Clune, Antoine Cully, and Jean-Baptiste Mouret. How do different encodings influence the performance of the MAP-Elites algorithm? In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 173–180, 2016.

Paul Templier, Emmanuel Rachelson, and Dennis G Wilson. A geometric encoding for neural network evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 919–927, 2021.

Robin Thandiackal, Kamilo Melo, Laura Paez, Johann Herault, Takeshi Kano, Kyoichi Akiyama, Frédéric Boyer, Dimitri Ryczko, Akio Ishiguro, and Auke J Ijspeert. Emergence of robust self-organized undulatory swimming based on local hydrodynamic force sensing. *Science robotics*, 6(57):eabf6354, 2021.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

David Tilman, Michael Clark, David R Williams, Kaitlin Kimmel, Stephen Polasky, and Craig Packer. Future threats to biodiversity and pathways to their prevention. *Nature*, 546(7656):73–81, 2017.

Anne Treisman, Alfred Vieira, and Amy Hayes. Automaticity and preattentive processing. *The American journal of psychology*, pages 341–362, 1992.

Leonardo Trujillo, Gustavo Olague, Evelyne Lutton, Francisco Fernandez de Vega, León Dozal, and Eddie Clemente. Speciation in behavioral space for evolutionary robotics. *Journal of Intelligent & Robotic Systems*, 64(3):323–351, 2011.

Peter D Turney. Myths and legends of the baldwin effect. *arXiv preprint cs/0212036*, 2002.

Gina G Turrigiano and Sacha B Nelson. Homeostatic plasticity in the developing nervous system. *Nature reviews neuroscience*, 5(2):97–107, 2004.

Nathan S Usevitch, Zachary M Hammond, Mac Schwager, Allison M Okamura, Elliot W Hawkes, and Sean Follmer. An untethered isoperimetric soft robot. *Science Robotics*, 5(40):eaaz0492, 2020.

Fuda van Diggelen, Eliseo Ferrante, Nihed Harrak, Jie Luo, Daan Zeeuwe, and AE Eiben. The Influence of Robot Traits and Evolutionary Dynamics on the Reality Gap. *IEEE Transactions on Cognitive and Developmental Systems*, 2021.

Vassiiis Vassiliades and Jean-Baptiste Mouret. Discovering the elite hypervolume by leveraging interspecies correlation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 149–156, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Frank Veenstra, Andres Faina, Sebastian Risi, and Kasper Stoy. Evolution and morphogenesis of simulated modular robots: a comparison between a direct and generative encoding. In *European Conference on the Applications of Evolutionary Computation*, pages 870–885. Springer, 2017.

Conrad H Waddington. Canalization of development and the inheritance of acquired characters. *Nature*, 150(3811):563–565, 1942.

Conrad Hal Waddington. *The strategy of the genes.* Routledge, 2014.

G. Wagner, M. Pavlicev, and J. Cheverud. The road to modularity. *Nature Reviews Genetics*, 8:921–931, 2007.

Günter P. Wagner and L. Altenberg. Complex adaptations and the evolution of evolvability. 2005.

Günter P Wagner, Jason Mezey, and Raffaele Calabretta. *Natural selection and the origin of modules.* na, 2001.

Michael Wainberg, Babak Alipanahi, and Brendan J Frey. Are random forests truly the best classifiers? *The Journal of Machine Learning Research*, 17(1): 3837–3841, 2016.

Kathryn Walker and Helmut Hauser. Evolution of morphology through sculpting in a voxel based robot. In *ALIFE 2021: The 2021 Conference on Artificial Life*. MIT Press, 2021.

Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *International Conference on Learning Representations*, 2018.

Bruce H Weber and David J Depew. *Evolution and learning: The Baldwin effect reconsidered.* Mit Press, 2003.

L Darrell Whitley. Fundamental principles of deception in genetic search. In *Foundations of genetic algorithms*, volume 1, pages 221–241. Elsevier, 1991.

Annie Wong, Thomas Bäck, Anna V Kononova, and Aske Plaat. Multiagent deep reinforcement learning: Challenges and directions towards human-like approaches. *arXiv preprint arXiv:2106.15691*, 2021.

Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision. *arXiv preprint arXiv:2006.03677*, 2020.

Anil Yaman, Giovanni Iacca, Decebal Constantin Mocanu, Matt Coler, George Fletcher, and Mykola Pechenizkiy. Evolving plasticity for autonomous learning under changing environmental conditions. *Evolutionary computation*, 29(3): 391–414, 2021.

Yuichi Yamashita and Jun Tani. Emergence of functional hierarchy in a multiple timescale neural network model: A humanoid robot experiment. *PLoS computational biology*, 4:e1000220, 12 2008. doi: 10.1371/journal.pcbi.1000220.

Mark Yim, Ying Zhang, John Lamping, and Eric Mao. Distributed control for 3d metamorphosis. *Autonomous Robots*, 10:41–56, 2001.

Mark Yim, Wei-Min Shen, Behnam Salemi, Daniela Rus, Mark Moll, Hod Lipson, Eric Klavins, and Gregory S Chirikjian. Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):43–52, 2007.

Crawford Young. *The politics of cultural pluralism.* Univ of Wisconsin Press, 1979.

Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Deep reinforcement learning with relational inductive biases. In *International Conference on Learning Representations*, 2018.

Davide Zappetti, Stefano Mintchev, Jun Shintake, and Dario Floreano. Bio-inspired tensegrity soft modular robots. In *Conference on Biomimetic and Biohybrid Systems*, pages 497–508. Springer, 2017.

Enrico Zardini, Davide Zappetti, Davide Zambrano, Giovanni Iacca, and Dario Floreano. Seeking quality diversity in evolutionary co-design of morphology and control of soft tensegrity modular robots. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 189–197, 2021.

Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and evolutionary computation*, 1(1): 32–49, 2011.

Karl Zilles. Neuronal plasticity as an adaptive property of the central nervous system. *Annals of Anatomy-Anatomischer Anzeiger*, 174(5):383–391, 1992.

Vita

Federico Pigozzi was born in Italy in 1995. He received a Bachelors degree in Economics *cum laude* from the University of Trieste, Italy, and a Masters degree in Data Science and Scientific Computing *cum laude* from the University of Trieste, Italy. He is graduating with a PhD degree in Computer Engineering at the University of Trieste, Italy, and has been a visiting research fellow at the University of Vermont, USA. His research interests include artificial intelligence and synthetic biology. In his free time, he enjoys stand-up comedy.

Address: pigozzife@gmail.com

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.