How the Morphology Encoding Influences the Learning Ability in Body-Brain Co-Optimization

Federico Pigozzi
University of Trieste
Trieste, Italy
federico.pigozzi@phd.units.it

Federico Julian Camerota Verdù
University of Trieste
Trieste, Italy
federicojulian.camerotaverdu@phd.units.it

Eric Medvet
University of Trieste
Trieste, Italy
emedvet@units.it

ABSTRACT

Embedding the learning of controllers within the evolution of morphologies has emerged as an effective strategy for the co-optimization of agents' bodies and brains. Intuitively, that is how nature shaped animal life on Earth. Still, the design of such co-optimization is a complex endeavor; one issue is the choice of the genetic encoding for the morphology. Such choice can be crucial for the effectiveness of learning, i.e., how fast and to what degree agents adapt, through learning, during their life. Here we evolve the morphologies of voxel-based soft agents 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 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.

CCS CONCEPTS

• **Computing methodologies** → **Evolutionary robotics**; *Reinforcement learning*.

KEYWORDS

Evolutionary reinforcement learning, Encoding, Plasticity, Embodiment

1 INTRODUCTION

Starting from the seminal work of Sims [45], researchers have resorted to evolution for co-optimizing embodied agents' bodies and brains. Still, the embodied cognition paradigm [39], 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 [9]. On the other side, learning alone is myopic since adapting the morphology is also essential [17]. As a result, co-optimization, by embedding the learning of controllers within the evolution of morphologies [15, 27], 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 [5, 54].

In this work, we evolve the morphologies of voxel-based soft agents using Evolutionary Algorithms (EAs) [8] while learning the controllers with Reinforcement Learning (RL) [50]. We resort to voxel-based soft agents as they provide so many more degrees of freedom and flexibility than traditional robots, providing ideal testbeds for theories related to embodied cognition [26]. 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 [41]. The most straightforward instance of plasticity is lifetime learning [24], also known as the epigenetic timescale of adaptation [46]. 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 1.

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 [57]. 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* [1, 2]. We remark that the Baldwin effect is not an instance of Lamarckism [53].

We experiment with three tasks from the EvoGym benchmark [3], consisting of bridge walking, beam toppling, and cave crawling, and analyze how the direct and indirect morphology encodings affect

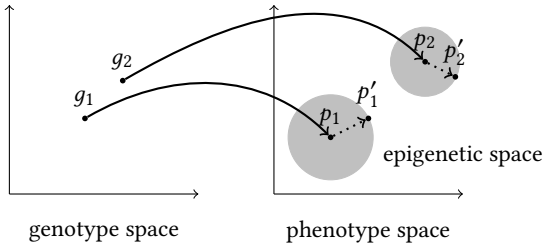


Figure 1: The intuition of plasticity. As a matter of example, we show a two-dimensional genotype space together with a two-dimensional phenotype space. A genotype g_i maps to a phenotype p_i, which is plastic: through learning, it can visit any point in its epigenetic space (the shaded circle) during its lifetime, to finally settle to a learned phenotype p_i'. The learning trajectories (the dotted lines) need not be straight.

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, the direct encoding is better suited at facilitating it. We attribute the reason to the lower locality [43] and heritability [7] 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.

2 RELATED WORK

Hinton et al. [19] 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 [35], population-based training [21], evolving instinctive behaviors for RL [16], and optimizing neural networks [49].

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 [15], inherited controller archives [13], inner evolutionary optimization [30], adaptive weights [10, 33, 40], or even Lamarckian evolution [22], 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. [15] detected signs of a Baldwin effect in tree-based robots and Luo et al. [27] in modular robots. Kriegman et al. [26] 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.

There are, however, other studies that compared different encodings for the morphology (and, in some cases, for the controller too) for robotic agents. Ferigo et al. [11] studied the interplay between evolvability and fitness for voxel-based soft robots that are evolved with four different encodings: similarly to our study, they compared direct and indirect encodings, but they did not consider learning, nor any other form of plasticity. Nadizar et al. [32] did, instead, consider a form of plasticity: they studied morphological development schedules for voxel-based soft robots and they experimentally compared a few encodings for both the controller and the morphology. Less recently, Veenstra et al. [54] compared a direct and a generative encoding for a different kind of modular robots, in particular in terms of their ability to foster the evolution of robots composed of different number of modules.

3 AGENT MODEL

We employ the open-source discrete-time and continuous-space simulator Evolution Gym (EvoGym) [3], which also provides a number of benchmark tasks. In EvoGym, agents are 2D voxel-based soft robots, 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*:

(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 2 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 $\boldsymbol{M} \in \{\varnothing, \text{pass-rigid}, \text{pass-soft}, \text{act-h}, \text{act-w}\}^{w \times h}$, 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 $\boldsymbol{a}^{(k)} \in [0.6, 1.6]^{n_{\text{act-h}}+n_{\text{act-v}}}$ 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

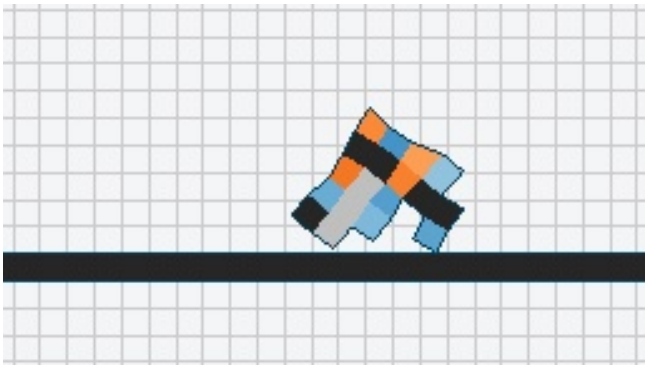


Figure 2: A voxel-based soft agent. The voxel color stands for the material: black is rigid inactive, gray is soft inactive, orange is a horizontal actuator, and cyan is a vertical actuator.

provides a reward signal $r^{(k)} \in \mathbb{R}$ that is task-specific and capture 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 a Multi-Layer Perceptron (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 [51]. Following [3], 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 $[0.6, 1.6]$).

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 [8], as well as generating emergent patterns for the morphology of virtual creatures [5, 6]. 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 [31] and there is evidence it mimics learning in the animal brain [34].

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 "offline" at the end of RL:

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

We iterate until $n_{\text{f-evals}}$ fitness evaluations have been done. Figure 3 is a schematic view of our co-optimization.

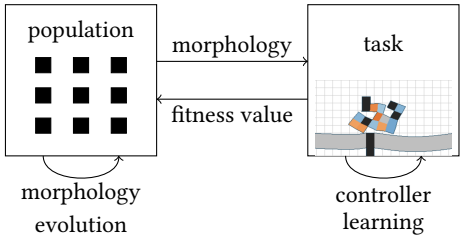


Figure 3: Overview of the co-optimization.

To investigate how the 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.

4.1 Evolution

For the morphology, evolutionary robotics 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 [5]); moreover, they strive to emulate the biological genome [37, 52], which is capable of compressing a whole phenotype with just a "few" genes (generally, in the order of thousands [14, 38]) while still fostering phenotypic variation [12]. We experiment with one variant for each type.

4.1.1 Direct encoding. Our direct encoding represents a morphology with the matrix $\boldsymbol{M}$ of Section 3 as genotype; then, each matrix entry encodes one voxel of the morphology.

For evolving $\boldsymbol{M}$, we employ a simple genetic algorithm [8]. It evolves a fixed-size population of n_{pop} individuals iterating the following two steps until $n_{\text{f-evals}}$ 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 [3].

4.1.2 Indirect encoding. Several indirect encodings exist in the literature that are suitable for voxel-based soft agents, e.g., L-systems

[20], gene regulatory networks [23], tree-based [32], and Gaussian mixtures [18], but we rely on the work of Cheney et al. [5] as it proved very effective at evolving morphologies specifically for multi-material voxel-based soft agents.

The genotype is a Compositional Pattern Producing Network (CPPN) [47], a feed-forward neural network with three input neurons and four output neurons. To map a CPPN to a 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) [48] 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 [4] 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 [48] for details. We used elitist selection as with GA. We used the implementation of [29] and the same hyperparameters of [3], with the exceptions of n_{pop} and $n_{\text{f-evals}}$, for which we use the same values as for the direct encoding case.

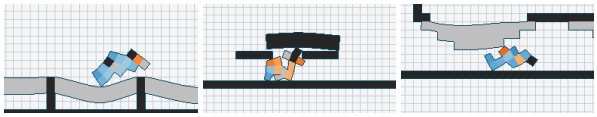
4.2 Learning

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 [5].

In particular, we employ the Proximal Policy Optimization (PPO) [44] algorithm, a policy gradient method that has reached state-of-the-art performance in many continuous control tasks [36] 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 [25] and the same hyperparameter settings of [3]: in particular, we set $n_{\text{RL-episodes}} = 32$ and $n_{\text{sim-RL-iter}} = 128$.



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

Figure 4: Three frames of agents performing the three tasks we considered in this paper.

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 voxel-based soft agents, 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 $n_{\text{f-evals}} = 500$; for `CaveCrawler`, which turned out more difficult to solve, we set $n_{\text{f-evals}} = 1000$. 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 [3].

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. Each run took approximately 8 h on a Linux virtual machine with 96 cores at 2.3 GHz and 48 GB of RAM.

5.1 Tasks

In the following sections, we briefly describe the environments used in our experiments, we refer the reader to [3, Appendix B] for more details.

5.1.1 `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:

$$r^{(k)} = x_a^{(k)} - x_a^{(k-1)}, \tag{2}$$

and a final reward of 1 upon reaching the end of the bridge. Figure 4a 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.

5.1.2 `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:

$$r^{(k)} = r_1^{(k)} + r_2^{(k)} + r_3^{(k)}, \tag{3}$$

with:

$$r_1^{(k)} = |x_b^{(k-1)} - x_a^{(k-1)}| - |x_b^{(k)} - x_a^{(k)}| \tag{4}$$

$$r_2^{(k)} = |x_b^{(k-1)} - x_b^{(k)}| + 3|y_b^{(k-1)} - y_b^{(k)}| \tag{5}$$

$$r_2^{(k)} = y_b^{(k-1)} - y_b^{(k-1)}, \tag{6}$$

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 4b 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.

5.1.3 `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 4c is a sample frame from the task. We adopted this task as it is among those classified as challenging in [3].

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 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 (by means of a few morphological descriptors) vary over the course of 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 the fitness for the best individual of each generation over the course of evolution, in terms of median ± standard deviation across the evolutionary runs.

Figure 5 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 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 6 for the direct encoding and Figure 7 for the indirect one. As observed in [5] and [11], 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 might be interpreted as a poor convergence or, from another 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.

In order 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 over the course of evolution.

5.2.1 Qualitative analysis: learning curves. To have an overview on how individuals learn at different stages of the 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, at $\frac{1}{3}n_{\text{f-evals}}$, at $\frac{2}{3}n_{\text{f-evals}}$, and at the end of the evolution. Then, for each individual in the population, we instrumented the PPO learning procedure to compute

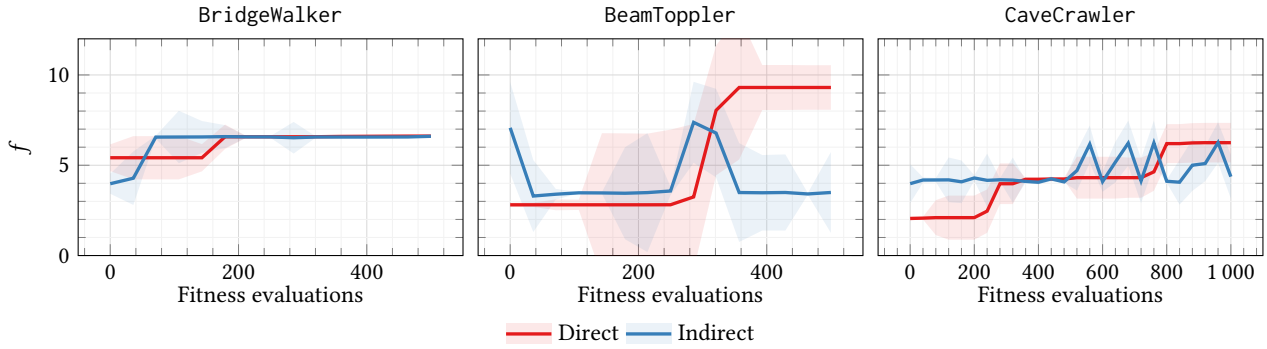
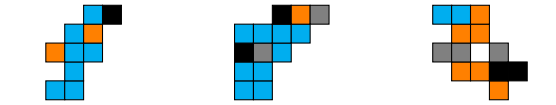
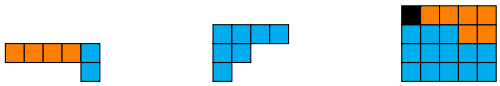


Figure 5: Median ± standard deviation across the evolutionary runs for the fitness f of the best individual over the course of evolution. Both encodings optimize effective solutions.



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

Figure 6: Sample morphologies evolved over three tasks with direct encoding.



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

Figure 7: Sample morphologies evolved over three tasks with indirect encoding.

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 8 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 8, 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 this task 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 $\approx$ 100 PPO iterations; in a previous stage, namely at $\frac{2}{3} n_{\text{f-evals}}$ fitness evaluations (green line), the learning is slower at the beginning but then slowly continues for all the 1000 PPO iterations.

Summarizing, 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.

5.2.2 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 9 visualizes the results as the distribution of ρ across evolutionary runs for the three tasks and the 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 9 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 in Section 5.2.1 is not the result of shorter trajectories for the agents.

5.2.3 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

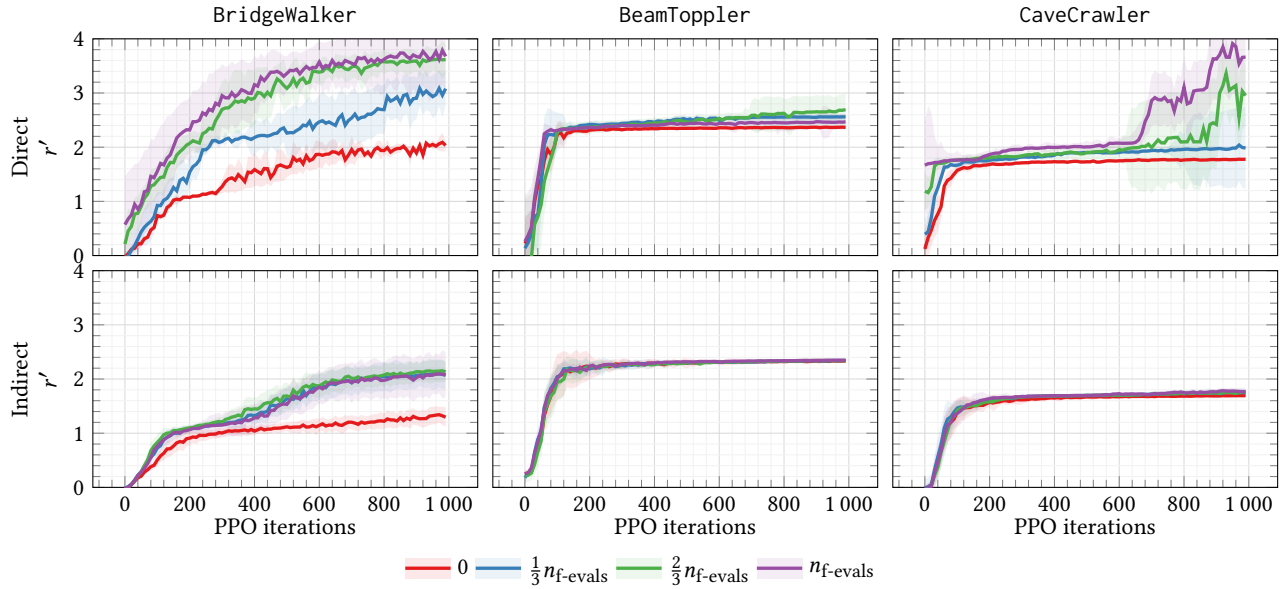


Figure 8: Moving average $r^{(k)}$ as a function of PPO iterations, taken at different generations, for three tasks and two encodings. Median ± standard deviation across evolutionary runs. Speed of learning increases for the direct encoding, while the same is not always true for the indirect.

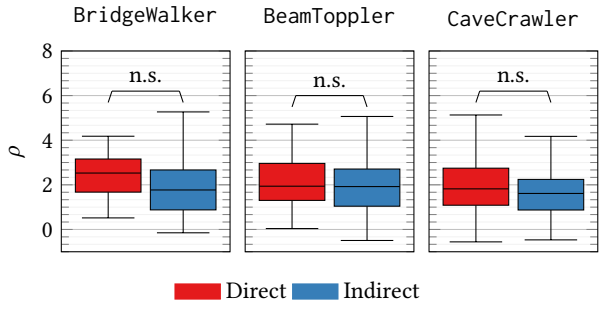


Figure 9: Distribution of learning radius ρ across evolutionary runs. There is no significant difference between the two encodings.

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 10, in terms of median ± standard deviation across evolutionary runs.

From Figure 10, we find that the two encodings radically differ in the trend of the descriptors. Morphological descriptors for direct encoding mostly follow monotonous paths, signaling there is a convergence 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 10 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 in order to topple the beam.

Moreover, Figure 10 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. [11] 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 8 and discussed in Section 5.2.1.

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,

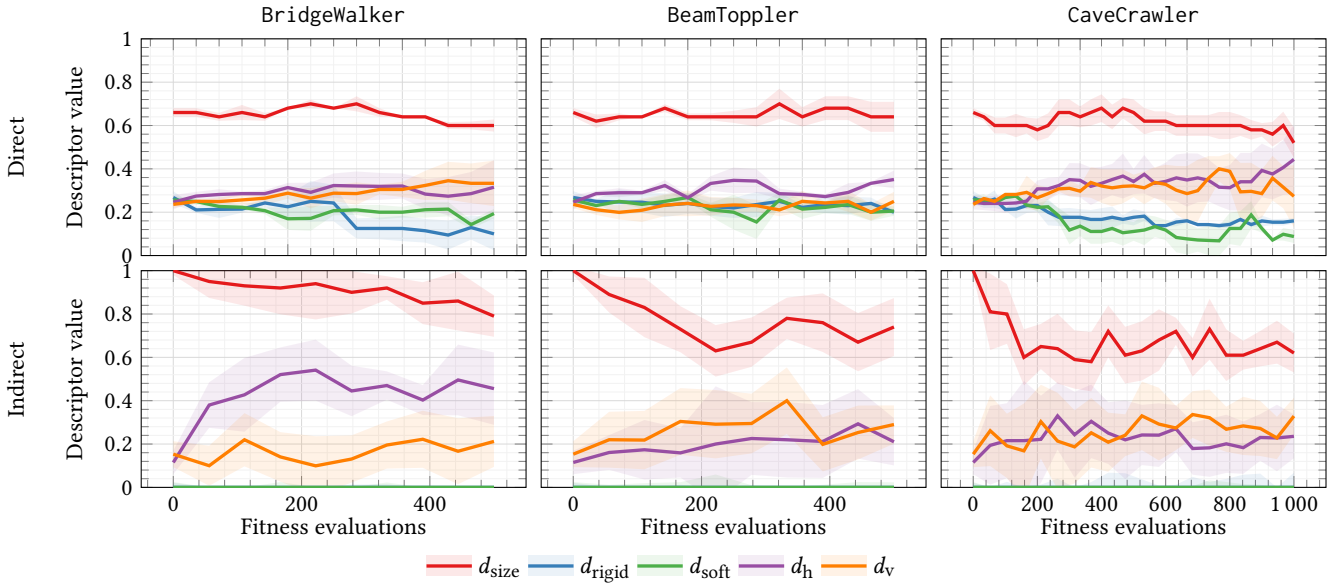


Figure 10: Median ± standard deviation across evolutionary runs for the morphological descriptors (fraction of non-empty voxels, rigid material voxels, soft material voxels, horizontal actuators, vertical actuators) of the population over the course of evolution. Paths of morphological descriptors are mostly monotonous for direct encoding, but erratic for indirect encoding.

we find that, in the case of direct encodings, effective controllers emerge more and more quickly and morphologies converge. In other words, morphologies "canalize" [55] towards designs that allow for learning to take place more swiftly [56]. Recalling Figure 1, we interpret this canalization as evolution that selects for morphologies that allow for phenotypes to "roll" over canals in the epigenetic space [56].

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* [42], one of the properties of genetic encodings [43], 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* [7], 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 [28] 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.

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 voxel-based soft agents—whose many degrees of freedom makes them ideal testbeds—with Evolutionary Algorithms (EAs) and, for each morphology, learn a controller through Reinforcement Learning (RL). For the EA, we test two encodings, direct and indirect. We experiment with three tasks from the Evolution Gym [3] 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 [42] and lower heritability [7] 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.

ACKNOWLEDGMENTS

The experiments in this work were conducted using virtual machines provided by Google Cloud Platform granted to F.P..

REFERENCES

[1] J Mark Baldwin. 1897. Organic selection. *Science* 5, 121 (1897), 634–636.
[2] James Mark Baldwin et al. 2018. A new factor in evolution. *Diacronia* 7 (2018), 1–13.
[3] Jagdeep Bhatia, Holly Jackson, Yunsheng Tian, Jie Xu, and Wojciech Matusik. 2021. Evolution gym: A large-scale benchmark for evolving soft robots. *Advances in Neural Information Processing Systems* 34 (2021), 2201–2214.
[4] Jürgen Branke. 1995. Evolutionary Algorithms for Neural Network Design and liraining. (1995).
[5] Nick Cheney, Robert MacCurdy, Jeff Clune, and Hod Lipson. 2013. 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*. ACM, 167–174.
[6] Francesco Corucci, Nick Cheney, Francesco Giorgio-Serchi, Josh Bongard, and Cecilia Laschi. 2018. Evolving soft locomotion in aquatic and terrestrial environments: effects of material properties and environmental transitions. *Soft robotics* 5, 4 (2018), 475–495.
[7] Matteo De Carlo, Eliseo Ferrante, Daan Zeeuwe, Jacintha Ellers, Gerben Meynen, and AE Eiben. 2021. Heritability in Morphological Robot Evolution. *arXiv preprint arXiv:2110.11187* (2021).
[8] Kenneth A De Jong. 2006. *Evolutionary Computation: A Unified Approach*. MIT Press.
[9] AE Eiben and Emma Hart. 2020. If it evolves it needs to learn. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*. 1383–1384.
[10] Andrea Ferigo, Giovanni Iacca, Eric Medvet, and Federico Pigozzi. 2022. Evolving Hebbian Learning Rules in Voxel-based Soft Robots. *IEEE Transactions on Cognitive and Developmental Systems* (2022), 1–1. https://doi.org/10.1109/TCDS.2022.3226556
[11] Andrea Ferigo, LB Soros, Eric Medvet, and Giovanni Iacca. 2022. 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.
[12] John Gerhart and Marc Kirschner. 2007. The theory of facilitated variation. *Proceedings of the National Academy of Sciences* 104, suppl 1 (2007), 8582–8589.
[13] Léni K Le Goff and Emma Hart. 2021. 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*. 1498–1502.
[14] 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. 2007. Eukaryotic genome size databases. *Nucleic acids research* 35, suppl_1 (2007), D332–D338.
[15] Agrim Gupta, Silvio Savarese, Surya Ganguli, and Li Fei-Fei. 2021. Embodied Intelligence via Learning and Evolution. *arXiv preprint arXiv:2102.02202* (2021).
[16] Ahmed Hallawa, Thorsten Born, Anke Schmeink, Guido Dartmann, Arne Peine, Lukas Martin, Giovanni Iacca, AE Eiben, and Gerd Ascheid. 2021. Evo-RL: evolutionary-driven reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 153–154.
[17] Emma Hart and Léni K Le Goff. 2022. Artificial evolution of robot bodies and control: on the interaction between evolution, learning and culture. *Philosophical Transactions of the Royal Society B* 377, 1843 (2022), 20210117.
[18] Jonathan Hiller and Hod Lipson. 2012. Automatic design and manufacture of soft robots. *IEEE Transactions on Robotics* 28, 2 (2012), 457–466.
[19] Geoffrey E Hinton, Steven J Nowlan, et al. 1987. How learning can guide evolution. *Complex systems* 1, 3 (1987), 495–502.
[20] Gregory S Hornby, Jordan B Pollack, et al. 2001. Body-brain co-evolution using L-systems as a generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*. 868–875.
[21] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. 2017. Population based training of neural networks. *arXiv preprint arXiv:1711.09846* (2017).
[22] Milan Jelisavcic, Kyrre Glette, Evert Haasdijk, and AE Eiben. 2019. Lamarckian evolution of simulated modular robots. *Frontiers in Robotics and AI* 6 (2019), 9.
[23] Michał Joachimczak, Reiji Suzuki, and Takaya Arita. 2016. Artificial Metamorphosis: Evolutionary Design of Transforming, Soft-Bodied Robots. *Artificial Life* 22, 3 (2016), 271–298. https://doi.org/10.1162/ARTL_a_00207 PMID: 27139940.
[24] Scott A Kelly, Tami M Panhuis, and Andrew M Stoehr. 2012. Phenotypic plasticity: molecular mechanisms and adaptive significance. *Compr Physiol* 2, 2 (2012), 1417–1439.
[25] Ilya Kostrikov. 2018. PyTorch Implementations of Reinforcement Learning Algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail.
[26] Sam Kriegman, Nick Cheney, and Josh Bongard. 2018. How morphological development can guide evolution. *Scientific reports* 8, 1 (2018), 13934.
[27] Jie Luo, Aart Stuurman, Jakub M Tomczak, Jacintha Ellers, and Agoston E Eiben. 2021. The Effects of Learning in Morphologically Evolving Robot Systems. *arXiv preprint arXiv:2111.09851* (2021).
[28] Giles Mayley. 1996. Landscapes, learning costs, and genetic assimilation. *Evolutionary Computation* 4, 3 (1996), 213–234.
[29] Alan McIntyre, Matt Kallada, Cesar G. Miguel, and Carolina Feher de Silva. [n. d.]. neat-python.
[30] Karine Miras, Matteo De Carlo, Sayfeddine Akhatou, and AE Eiben. 2020. Evolving-controllers versus learning-controllers for morphologically evolvable robots. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 86–99.
[31] 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. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.
[32] Giorgia Nadizar, Eric Medvet, and Karine Miras. 2022. On the schedule for morphological development of evolved modular soft robots. In *Genetic Programming: 25th European Conference, EuroGP 2022, Held as Part of EvoStar 2022, Madrid, Spain, April 20–22, 2022, Proceedings*. Springer, 146–161.
[33] Giorgia Nadizar, Eric Medvet, Hola Huse Ramstad, Stefano Nichele, Felice Andrea Pellegrino, and Marco Zullich. 2022. Merging pruning and neuroevolution: towards robust and efficient controllers for modular soft robots. *The Knowledge Engineering Review* 37 (2022).
[34] Emre O Neftci and Bruno B Averbeck. 2019. Reinforcement learning in artificial and biological systems. *Nature Machine Intelligence* 1, 3 (2019), 133–143.
[35] Scott Niekum, Andrew G Barto, and Lee Spector. 2010. Genetic programming for reward function search. *IEEE Transactions on Autonomous Mental Development* 2, 2 (2010), 83–90.
[36] 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. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. (2019). arXiv:1912.06680 https://arxiv.org/abs/1912.06680
[37] Joachim Winther Pedersen and Sebastian Risi. 2021. Evolving and merging Hebbian learning rules: increasing generalization by decreasing the number of rules. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 892–900.
[38] Jaume Pellicer and Ilia J Leitch. 2019. The Plant DNA C-values database (release 7.1): an updated online repository of plant genome size data for comparative studies. *[" New Phytologist"]* (2019).
[39] Rolf Pfeifer and Josh Bongard. 2006. *How the body shapes the way we think: a new view of intelligence*. MIT press.
[40] Federico Pigozzi, Yujin Tang, Eric Medvet, and David Ha. 2022. Evolving Modular Soft Robots without Explicit Inter-Module Communication using Local Self-Attention. In *Proceedings of the 24th annual conference on Genetic and Evolutionary Computation*. ACM.
[41] Trevor D Price, Anna Qvarnström, and Darren E Irwin. 2003. The role of phenotypic plasticity in driving genetic evolution. *Proceedings of the Royal Society of London. Series B: Biological Sciences* 270, 1523 (2003), 1433–1440.
[42] Franz Rothlauf. 2003. On the locality of representations. *None* (2003).
[43] Franz Rothlauf. 2006. Representations for genetic and evolutionary algorithms. In *Representations for Genetic and Evolutionary Algorithms*. Springer, 9–32.
[44] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).
[45] Karl Sims. 1994. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM, 15–22.
[46] Moshe Sipper, Eduardo Sanchez, Daniel Mange, Marco Tomassini, Andrés Pérez-Uribe, and André Stauffer. 1997. A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems. *IEEE Transactions on Evolutionary Computation* 1, 1 (1997), 83–97.
[47] Kenneth O Stanley. 2007. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines* 8, 2 (2007), 131–162.
[48] Kenneth O Stanley and Risto Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation* 10, 2 (2002), 99–127.
[49] Jörg Stork, Martin Zaefferer, Nils Eisler, Patrick Tichelmann, Thomas Bartz-Beielstein, and AE Eiben. 2021. Behavior-based neuroevolutionary training in reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1753–1761.
[50] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.
[51] Jacopo Talamini, Eric Medvet, Alberto Bartoli, and Andrea De Lorenzo. 2019. Evolutionary Synthesis of Sensing Controllers for Voxel-based Soft Robots. In

Artificial Life Conference Proceedings. MIT Press, 574–581.

[52] Yujin Tang, Duong Nguyen, and David Ha. 2020. Neuroevolution of self-interpretable agents. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 414–424.

[53] Peter D Turney. 2002. Myths and legends of the Baldwin effect. *arXiv preprint cs/0212036* (2002).

[54] Frank Veenstra, Andres Faina, Sebastian Risi, and Kasper Stoy. 2017. Evolution and morphogenesis of simulated modular robots: a comparison between a direct and generative encoding. In *European Conference on the Applications of Evolutionary Computation*. Springer, 870–885.

[55] Conrad H Waddington. 1942. Canalization of development and the inheritance of acquired characters. *Nature* 150, 3811 (1942), 563–565.

[56] Conrad Hal Waddington. 2014. *The strategy of the genes.* Routledge.

[57] Bruce H Weber and David J Depew. 2003. *Evolution and learning: The Baldwin effect reconsidered.* Mit Press.