On the Schedule for Morphological Development of Evolved Modular Soft Robots

Giorgia Nadizar[1][0000-0002-3535-9748], Eric Medvet[2][0000-0001-5652-2113], and Karine Miras[3][0000-0003-4942-3488]

[1] Department of Mathematics and Geosciences, University of Trieste, Italy
[2] Department of Engineering and Architecture, University of Trieste, Italy
[3] Artificial Intelligence Department, Rijksuniversiteit Groningen, Netherlands

Abstract. Development is fundamental for living beings. As robots are often designed to mimic biological organisms, development is believed to be crucial for achieving successful results in robotic agents, as well. What is not clear, though, is the most appropriate scheduling for development. While in real life systems development happens mostly during the initial growth phase of organisms, it has not yet been investigated whether such assumption holds also for artificial creatures. In this paper, we employ a evolutionary approach to optimize the development—according to different representations—of Voxel-based Soft Robots (VSRs), a kind of modular robots. In our study, development consists in the addition of new voxels to the VSR, at fixed time instants, depending on the development schedule. We experiment with different schedules and show that, similarly to living organisms, artificial agents benefit from development occurring at early stages of life more than from development lasting for their entire life.

Keywords: Adaptation · Evolutionary Robotics · Embodied cognition · Development · Body-brain evolution

1 Introduction and related works

Phenotypic development is pervasive in nature and it can happen in different dimensions, e.g., lifetime body adaptations to cope with environmental seasonal changes [1], brain plasticity through learning [2], body training [3], behavioral environmental regulation [4], etc. Additionally to these forms of development, there is a very fundamental one: growth. Notably, growth starts during morphogenesis and may continue for a long period during the lifetime of a creature, according to its species. In humans, the fluctuations of phenotypic growth depend on genetic and environmental factors during prepubertal and pubertal development [5]. Because of the complexity of these factors, the difficulty of establishing growth standards has been discussed [6]. Moreover, different body traits develop at distinct ages. For instance, while the significant height growth happens until adolescence [7], male muscle mass peaks around their 20 to 30 years old [8].

Curiously, while body growth in animals is rapid in early life, it then progressively slows, and the reasons for this are not yet quite understood [7], though its association to possible advantages of delaying fertility maturation has been discussed [9]. Furthermore, animal brain development is lifelong, happening prenatally, during infancy and adolescence, and even in adulthood [10]. In fact, because of its complexity, the human frontal cortex is not fully developed until the mid twenties [4]. Note that this developmental maturation does not necessarily mean growth in the brain, but a better organization of the neural structures, requiring some form of pruning [4]. Interestingly, synaptic pruning—i.e., removing some of the connections in the neural network constituting the brain—has been shown to be potentially beneficial also for evolved artificial agents [11,12].

While the dynamics of growth, including the interplay between body and brain development, is unclear, we could hardly doubt that this complex dynamics is fundamental for the behavioral complexity observed in diverse species. Therefore, the field of Evolutionary Robotics has great motivation to study growth development. Nevertheless, not only is the field focused mostly on evolving the controller without evolving the body [13], but also has development received relatively little attention.

Some developmental representations have become popular [14,15], but they have been mostly used for morphogenesis only. One instance of a development study has demonstrated the benefits of environmental regulation for lifetime phenotypic plasticity so that bodies and brains of robots could adapt in response to environmental changes [16,17]. Another approach experimented with reconfigurable robots that relied on manually designed bodies [18]. Furthermore, a field of research called morphogenetic engineering has been introduced to promote models of complex self-architecture systems [19]. Some other examples of development studies are the use of pre-programmed lifetime changes that could be compared to growth: investigating the impact of development on evolvability [20,21], and exploring the effects of using stages of morphological development as a way of scaffolding behavior [22].

Although the aforementioned works represent an important step for the investigations of development within artificial life, there is still a lot to be explored. There is a pressure for an increase in this type of investigation, with the purpose of expanding our perspective about how development can be carried out, and in which conditions determined effects shall be observed.

Aiming at furnishing the literature with new insights, in this paper we address a specific question related to growth development: does the development schedule impact on the effectiveness of evolved agents? Namely, is a continuous, lifelong development better or worse than a development that occurs mostly at the beginning of the life of the agent? To this extent, we design various development representations for 2-D simulated modular Voxel-based Soft Robots (VSRs) [23] which can be optimized via suitable evolutionary algorithms (EAs): we hence combine development with evolution, allowing robots to undergo alterations on different timescales. Due to their expressive power, VSRs are ideal for experimenting with morphological development and they have already been used in [21,24]: differently from the present paper, the two cited works do not study the development schedule, but the overall impact of development on evolution, in [21], and the possibility of exploiting environmental feedback for determining the development, in [24].

Even though our work focuses on morphological development, involving only the body of the agent, the robot controller is tightly coupled with its morphology: therefore, we also design evolvable brains which can effectively control different bodies. To assess the effects of development, we evaluate the performance of robots in a locomotion task. For providing more context, our study encompasses also agents which do not undergo development, considered as a baseline. Our results show that, for all representations, the most appropriate scheduling of development for artificial agents resembles that of living organisms. Namely, we find that early development yields to better performing robots than those which experience continuous growth. Moreover, the comparison with non-developing robots confirms the potentially beneficial effects of development for artificial agents.

2 Background: Voxel-based Soft Robots

We experiment with Voxel-based Soft Robots (VSRs), a kind of modular soft robots. Each VSR module consists of a deformable cube (*voxel*), which can vary in volume in order to achieve movement. The final volume of each voxel is determined by two factors: (a) external forces acting on the voxel, which is deformable by construction, and (b) a control signal, regulating active expansion/contraction of the voxel. In this study, we consider a 2-D variant of said robots that are simulated in discrete time [25]. Working in 2-D reduces the computational cost of the simulation, while yielding results that are conceptually portable to the 3-D case.

A VSR is defined by its *morphology* and its *controller*, describing respectively the arrangement of voxels in a 2-D grid and the law which determines each voxel control signal.

2.1 VSR morphology

The morphology of a VSR describes the voxel arrangement in a 2-D grid. Each voxel is a soft deformable square, modeled with (a) four masses at the corners, to prevent excessive deformations and rigidly connect neighbors, (b) spring-damper systems, to ensure softness and elasticity, and (c) ropes, to avoid uncontrolled expansion. We refer the reader to [26] for further details on the voxel mechanical model utilized.

VSRs accomplish movement similarly to biological muscles, thanks to the contraction and expansion of individual voxels. The behavior of each voxel is determined by a control signal and by the interaction with other bodies, exerting their forces on it, e.g., the ground or other voxels. At each simulation time step k, the controller feeds every i-th voxel with a control signal $a_i^{(k)} \in [-1, 1]$, -1 corresponding to maximum requested expansion, and 1 corresponding to maximum requested contraction. In the simulator employed [25], contraction and expansions are modeled as linear variations of the rest-length of the spring-damper system, proportional to the control signal received.

Voxels are equipped with sensors. We use three types of sensors, whose readings can be exploited by the VSR controller: (a) *area* sensors, perceiving the ratio between the current area of the voxel and its rest area, (b) *touch* sensors, sensing if the voxel is in contact with the ground or not, and (c) *velocity* sensors, which perceive the velocity of the center of mass of the voxel along the x- and y-axes (thus corresponding to the union of a v_x and a v_y sensor). We normalize sensor readings in such a way that, at each simulation time step, the readings $\boldsymbol{s}_i^{(k)}$ of the i-th voxel are defined in $[-1,1]^4$.

2.2 VSR controller

At each simulation time step k, the VSR controller is fed with the sensor readings $\boldsymbol{s}^{(k)} = [\boldsymbol{s}_1^{(k)}\ \boldsymbol{s}_1^{(k)} \dots]$ and outputs the control signals $\boldsymbol{a}^{(k)} = (a_1^{(k)}, a_2^{(k)}, \dots)$ for all voxels.

In this study, we consider two different kinds of controllers: the phase controller [23,27,21] and the neural controller [28,29]. For both controllers, in this work, even though we compute the control signal at each simulation time step, we actually apply it to the voxel every $t_\text{step} = 0.5\,\text{s}$ and keep it constant in between variations. In other words, we employ a step like control signal derived by the original control signal. We do this because we aim at preventing vibrating behaviors, which have been found to be a strong attractor in evolution of VSRs [30].

Phase controller. In the phase controller, each control signal is computed from the current time, according to a sinusoidal function. Namely, the control signal of the i-th voxel at simulation time step k is computed as $a_i^{(k)} = \sin\left(2 f \pi k \Delta t + \phi_i\right)$, where f is the sine wave frequency, Δt is the simulation time interval, and ϕ_i is the voxel phase. In most works where they have been used, these controllers have been optimized only in the phases ϕ_i, whereas f is set a priori to the same value for each voxel: for this reason, these are called phase controllers.

Note that this is an open-loop type of controller, which does not exploit sensor readings $\boldsymbol{s}^{(k)}$.

Neural controller. VSR neural controllers are based on Artificial Neural Networks (ANNs), which are employed to process sensor readings and produce voxel control signals. Neural controllers have been demonstrated beneficial for the achievement of good performance in VSRs [28,31], due to their sensing abilities.

We use the *distributed* neural controller presented in [29], consisting of a number of fully-connected feed-forward ANNs, i.e., multi-layer perceptrons (MLPs), one per voxel. At every simulation time step k, each MLP processes the local sensor readings $\boldsymbol{s}_i^{(k)}$ together with the information coming from the neighboring voxels, in order to produce the local control signal $a_i^{(k)}$ and the information to be passed to neighboring voxels. The information passed between neighboring voxels consists of a vector of n_signal values, or a zero-vector of the same size for voxels at the boundaries. Such vector is processed with one time step of delay, i.e., every MLP processes the vectors produced by neighboring MLPs at the previous simulation time step. The control signal of the i-th voxel is hence determined as $\left[a_i^{(k)}\ \boldsymbol{m}_{i,N}^{(k)}\ \boldsymbol{m}_{i,E}^{(k)}\ \boldsymbol{m}_{i,S}^{(k)}\ \boldsymbol{m}_{i,W}^{(k)}\right] = \text{MLP}_{\boldsymbol{\theta}}\left(\left[\boldsymbol{s}_i^{(k)}\ \boldsymbol{m}_{i_N,S}^{(k-1)}\ \boldsymbol{m}_{i_E,W}^{(k-1)}\ \boldsymbol{m}_{i_S,N}^{(k-1)}\ \boldsymbol{m}_{i_W,E}^{(k-1)}\right]\right)$, where $\boldsymbol{m}_{i,N}^{(k)} \in [-1,1]^{n_\text{signal}}$ is the information output by the i-th voxel for its neighbor at north (the same for the other three neighbors), $\boldsymbol{m}_{i_N,S}^{(k-1)} \in [-1,1]^{n_\text{signal}}$ is the information output by the neighbor at north for the i-th voxel (that is its neighbor at south) at previous time step $k-1$, and $\boldsymbol{\theta} \in \mathbb{R}^p$ is the vector of the parameters (or weights) of the MLP.

In this study, we utilize an identical MLP in each voxel, both in terms of architecture and weights $\boldsymbol{\theta}$. This design choice arises from the fact that this controller can be employed for a variable amount of voxels without any changes, so it is particularly suitable for a developing body. Moreover, [30] showed experimentally that using the same MLP in each voxel is not worse than using MLPs with different weights.

3 Development of VSRs

We consider morphological development, i.e., a mechanism according to which, at given time instants during the life of the VSR, new voxels are added to the VSR body.

For the purpose of this study, we say that the development of a VSR is completely described by a schedule and a development function. We define the *schedule* as a sequence $S = (t_j)_j$ of time instants when the addition of a new voxel occurs. We define the *development function* as a function d that, given a number of voxels n, outputs a VSR $d(n)$ consisting of at most n voxels. We impose the requirements for the function d that, for all n, (a) the morphology of the VSR $d(n)$ differs from the morphology of the VSR $d(n+1)$ for at most one voxel and (b) $d(n)$ has no more voxels than $d(n+1)$.

Given a starting size n_0, a schedule S, and a development function d, we can perform a simulation of a VSR $d(n_0)$ that starts with a morphology of (at most) n_0 voxels at $t=0$ and, at each $t_j \in S$, develops to a VSR $d(n_0+j)$ that is not smaller than the previous one.

The main goal of this study is to gain insights into the impact of the schedule on the effectiveness of evolved developing VSRs. To achieve this goal, we set the schedule to a few predefined sequences, let the evolution optimize the developing function, and compare the outcomes.

3.1 Representations for the development function

For broadening the generality of our experimental findings, we consider four different ways of representing the developing function in a way that allows its optimization by the means of evolution.

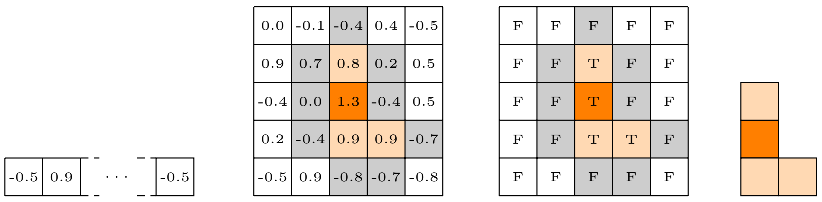
For the ease of presentation, we describe the development function d in terms of a function d_{morph}, that determines the morphology of the VSR, and a function $d_{\text{controller}}$, that determines the controller of the VSR. Moreover, we directly describe how the outputs of $d_{\text{morph}}(n, g)$ and $d_{\text{controller}}(n, g)$ are computed, given a genotype g and a number n.

Vector-based morphology representation. Given a real vector $\boldsymbol{v} \in \mathbb{R}^{n_{\text{side}}^2}$, we obtain a morphology of n voxels as follows.

We denote by $\boldsymbol{M} = d_{\text{morphology}}(n, \boldsymbol{v})$ the Boolean matrix describing the obtained morphology, where the voxel at position i, j is present if and only if the corresponding element $m_{i,j}$ is set. First, we reshape the vector $\boldsymbol{v}$ to a matrix $\boldsymbol{V}$ of n_{side}^2 real values. Second, we determine the greatest element of $\boldsymbol{V}$ and set the corresponding element of $\boldsymbol{M}$. Then, we repeat the following two steps until $\min(n, n_{\text{side}}^2)$ elements of $\boldsymbol{M}$ have been set: (1) we consider the subset of $\boldsymbol{M}$ unset elements that are adjacent to set elements and (2) we set the element of the subset with the largest corresponding value in $\boldsymbol{V}$.

Note that, with this representation, it is guaranteed that the morphology will have exactly n voxels, provided that n_{side}, a parameter of the representation, is large enough.

Figure 1 provides a schematic representation of an example of application of this function with $n_{\text{side}} = 5$ and $n = 4$.



0.0	-0.1	-0.4	0.4	-0.5
0.9	0.7	0.8	0.2	0.5
-0.4	0.0	1.3	-0.4	0.5
0.2	-0.4	0.9	0.9	-0.7
-0.5	0.9	-0.8	-0.7	-0.8

F	F	F	F	F
F	F	T	F	F
F	F	T	F	F
F	F	T	T	F
F	F	F	F	F

(a) Real vec. $\boldsymbol{v}$. (b) Real matrix $\boldsymbol{V}$. (c) Bool. mat. $\boldsymbol{M}$. (d) VSR.

Fig. 1: Schematic view of the grid-based morphology representation $d_{\text{morph}}(n, \boldsymbol{v})$, with $n_{\text{side}} = 5$, $n = 4$, and an example $\boldsymbol{v} \in \mathbb{R}^{25}$. Dark orange is used to highlight the first element chosen (the one with highest value), whereas lighter orange is used to indicate the other chosen elements. The gray area indicates the candidate voxels for a possible future development (i.e., $n = 5$ with this same $\boldsymbol{v}$).

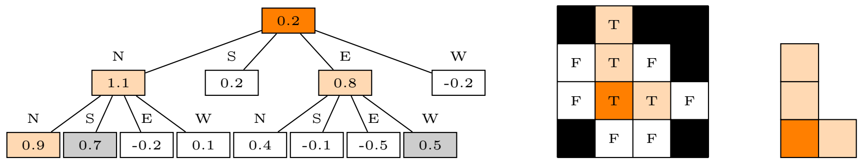
Tree-based morphology representation. Given an ordered tree T in which each node is a number in $\mathbb{R}$ and has either 0 or 4 child nodes, we obtain a morphology of up to n voxels as follows.

Each node of the tree corresponds to an element of the matrix $\boldsymbol{M}$ describing the morphology and the four children of a node correspond to the four neighboring elements at north, south, east, and west. Given a node corresponding to the $m_{i,j}$ element, the first child corresponds to $m_{i,j-1}$, the second to $m_{i,j+1}$, etc.

First, we transform T into a tree T' by mapping each node of T to a node in T' being a pair (v, u), where $v \in \mathbb{R}$ is the node real value and $u \in \{\textsc{set}, \textsc{unset}, \textsc{used}\}$—initially, $u = \textsc{unset}$ for every node in T'. Second, we set the root u to SET. Then, we repeat the following three steps until n nodes in T' have $u = \textsc{set}$ or there are no more T' nodes with $u = \textsc{unset}$: (1) we consider the subset of nodes with $u = \textsc{unset}$ and whose parent node has $u = \textsc{set}$, (2) we choose the node in the subset with the largest v, and (3) set $u = \textsc{set}$ for the chosen node and $u = \textsc{used}$ for all the other nodes representing the same position and with $u = \textsc{unset}$. Finally, we obtain the morphology $\boldsymbol{M}$ by setting the element $m_{0,0}$—for convenience, we assume that the indexes of the elements of $\boldsymbol{M}$ can assume values in $\mathbb{Z}$—and setting every other element i, j for which there is a SET node in T' whose relative position to the root is i, j.

Note that, with this representation, a morphology with less than n voxels could be obtained for a given tree T. In the extreme case, if T consists of the root node only, then the morphology will have just one voxel. On the other hand, the representation is parameter-free and does not impose an upper bound on the number of voxels in the VSR. Note also that, for each position of the matrix $\boldsymbol{M}$ there are up to four nodes in the tree T, but at most one is used depending on ancestors of the node: i.e., this representation is redundant [32] and exhibits epistasis [33].

Figure 2 provides a schematic representation of an example of application of this function with $n = 4$.



(a) Tree T'. (b) Bool. matr. $\boldsymbol{M}$. (c) VSR.

Fig. 2: Schematic view of the tree-based morphology representation $d_{\text{morph}}(n, T)$, with $n = 4$. Colors in T' nodes represent the value of u, while numbers are the ones of T (not shown here for brevity). Dark orange is used to highlight the root of the tree, whereas lighter orange is for $u = \textsc{set}$, white for $u = \textsc{unset}$, and gray for $u = \textsc{used}$. The same colors are used in the Boolean matrix $\boldsymbol{M}$ and in the obtained VSR morphology. Black cells in $\boldsymbol{M}$ correspond to nodes that are not present in the tree, hence such cells could never be used with this T, regardless of n.

Vector-based phase controller representation. Given a real vector $v \in \mathbb{R}^{n_{\text{side}}^2}$, we obtain a phase controller for a VSR whose morphology $\boldsymbol{M}$ can be contained in a $n_{\text{side}} \times n_{\text{side}}$ grid of voxels as follows.

First, we reshape the vector $\boldsymbol{v}$ to a matrix $\boldsymbol{V}$ of $n_{\text{side}} \times n_{\text{side}}$ real values. Second we build a phase controller in which the phase ϕ of the voxel at position

i, j, if any, is given by the corresponding element in $\boldsymbol{V}$. If there is no voxel at i, j, the corresponding element in $\boldsymbol{V}$ does not contribute to the controller.

Vector-based neural controller representation. Given a real vector $v \in \mathbb{R}^p$ and a description of the topology of an MLP consisting on the numbers $l_1, \dots, l_m$ of neurons for each layer (with $p = \sum_{i=1}^{i<m} l_{i+1}(l_i + 1)$), we obtain a neural controller by simply setting the parameters vector $\boldsymbol{\theta}$ of the MLP to $\boldsymbol{v}$.

Note that, since the MLP is the same for all voxels, this controller is applicable to any VSR, regardless of its morphology, provided that (a) l_1 is compatible with the dimension $n_\text{sensor} = |\boldsymbol{s}_i^{(k)}|$ of sensor readings in the voxels and with the value of n_signal and (b) l_m is compatible with the value of n_signal. More precisely, given n_sensor and n_signal, $l_1 = n_\text{sensor} + 4n_\text{signal}$ and $l_m = 1 + 4n_\text{signal}$ must hold. Since n_sensor is determined by the morphology ($n_\text{sensor} = 4$ in this work, see Section 2.1), it follows that the free parameters for this representations are n_signal, m, and the values of l_j for $2 \leq j < m$.

Tree-based phase controller representation. This controller representation is tightly coupled with the tree-based morphology representation: in fact, we only use it in combination with that representation. Given an ordered tree T in which each node is a pair of numbers $v, \phi \in \mathbb{R}^2$ and has either 0 or 4 child nodes, we obtain a phase controller for the VSR mapped from the v-part of T according to the tree-based morphology representation, as follows. Let M be the morphology obtained as described above, we associate with each voxel in the morphology the ϕ value of the corresponding element in T.

The rationale for this representation, is to tightly couple v values, determining the morphology, and ϕ values, determining the controller, by embedding them in the same tree. Together with appropriate genetic operators, this link should prevent destructive effects resulting from the misalignment between the part of the genotype describing the morphology and the one describing the brain [34].

Full development function representations. Summarizing, we consider the four representations resulting from the following combinations of a d_morph and a $d_\text{controller}$ representation:

- *Grid-phase*, in which the genotype is a vector $\boldsymbol{v} \in \mathbb{R}^{2n_\text{side}^2}$: we obtain the robot by mapping the leading half of $\boldsymbol{v}$ with the vector-based morphology representation and the trailing half with the vector-based phase controller representation.
- *Grid-neural*, in which the genotype is a vector $\boldsymbol{v} \in \mathbb{R}^{n_\text{side}^2+p}$: we obtain the robot by mapping the leading n_side^2 elements of $\boldsymbol{v}$ with the vector-based morphology representation and the trailing p elements with the vector-based neural controller representation.
- *Tree-phase*, in which the genotype is a tree T with nodes in $\mathbb{R}^2$ with either 0 or 4 children: we obtain the robot by mapping the tree of the first elements of T nodes with the tree-based morphology representation and T with the tree-based phase controller representation.

– *Tree-neural*, in which the genotype is a pair $T, \boldsymbol{v}$ composed of a tree T with nodes in $\mathbb{R}$ with either 0 or 4 children and a vector $\boldsymbol{v} \in \mathbb{R}^p$: we obtain the robot by mapping T with the tree-based morphology representation and $\boldsymbol{v}$ with the vector-based neural controller representation.

3.2 Evolution of the development function

To evolve a development function, we employ a single, standard EA that we adapt, in the initialization and genetic operators, to the four different representations presented above.

In our EA, we iteratively evolve a population of n_{pop} solutions for n_{gen} generations. At each generation, we build the offspring by repeating n_{pop} times the following steps: (1) we randomly select the crossover (with probability p_{cross}) or the mutation (with probability $1 - p_{\text{cross}}$) genetic operator; (2) we select one or two parents (depending on the chosen operator) with a tournament selection of size n_{tour}; (3) we apply the operator to the parents obtaining a new individual. Then, we merge the parents and the offspring, we keep only the n_{pop} best individuals, and proceed to the next generation.

For initializing the population, we sample $U(-1, 1)$ for each element of the vector-based representations, and we use the ramped half-and-half initialization (with depth in $[d_{\text{min}}, d_{\text{max}}]$) for the tree-based representation. For the latter, we sample $U(-1, 1)$ for the values of the nodes.

Concerning the genetic operators, we do as follows. In the vector-based representations (grid-phase and grid-neural), we use the extended geometric crossover, where the child $\boldsymbol{v} \in \mathbb{R}^n$ is determined from the parents $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathbb{R}^n$ as $\boldsymbol{v} = \boldsymbol{v_1} + \boldsymbol{\alpha}(\boldsymbol{v_2} - \boldsymbol{v_1}) + \boldsymbol{\beta}$, where $\boldsymbol{\alpha} \in \mathbb{R}^n$ is sampled as $\alpha_i \sim U(-0.5, 1.5)$, and $\boldsymbol{\beta} \in \mathbb{R}^n$ is sampled as $\beta_i \sim N(0, \sigma_{\text{cross}})$. As mutation, we use the Gaussian mutation, where $\boldsymbol{v} = \boldsymbol{v_1} + \boldsymbol{\beta}$, with betas sampled from $N(0, \sigma_{\text{mut}})$.

In the tree-based representations, we use the standard subtree crossover, which consists in replacing a random subtree of one parent with a randomly chosen subtree of the other parent: both subtrees are picked to ensure the child tree has a maximum depth of d_{max}. As mutation, we use the standard subtree mutation, in which one random subtree is replaced with a newly generated tree, ensuring a maximum depth of d_{max} for the child. Only with the tree-phase representation, with 50 % probability, we apply a noise sampled from $N(0, \sigma_{\text{mut}})$ to each ϕ element of the tree instead of applying the standard subtree mutation.

In the combined representation (tree-neural), we do crossover by applying standard subtree crossover and extended geometric crossover to the two parts of the genotype. Similarly, we do mutation by applying Gaussian mutation and standard subtree mutation.

4 Experimental evaluation

We performed several experiments to answer to the following research questions: What is the most appropriate development schedule for artificial agents? Does it depend on the representation of the development function?

For answering to said questions, we evolved development functions to develop VSRs suited for the task of *locomotion*, in which the goal for the robot is to travel as far as possible on a terrain in a given amount of time. We employed two different development schedules, together with no development, to be considered as a baseline. A detailed description of the experimental procedure and results follows.

Concerning the representation, we used the following parameters: $n_\text{side} = 10$, $f = 1\,\text{Hz}$, $n_\text{signal} = 2$, $m = 4$, and $l_2 = l_3 = l_1 = 4+4\cdot 2 = 12$ (i.e., we used MLPs with two inner layers with the same size of the input layer). Regarding the EA, we used the following parameters: $n_\text{pop} = 96$ and $n_\text{gen} = 209$ (corresponding to 20 000 total fitness evaluations), $p_\text{cross} = 0.75$, $n_\text{tour} = 10$, $\sigma_\text{cross} = 0.1$, $\sigma_\text{mut} = 0.35$, $d_\text{min} = 3$, and $d_\text{min} = 6$. We verified that, for the chosen value of n_pop and n_gen, evolution was in general capable of converging to a solution, i.e., longer evolutions would have resulted in negligible fitness improvements.

To evaluate the effectiveness of an individual, i.e., a development function, given a schedule S, we proceeded as follows. At the beginning of the simulation, we (1) used the development function to obtain an initial VSR $d(n_0)$ from n_0 and (2) we placed it right above the terrain at the starting position. Then, at each $t_i \in S$ during the simulation, we (1) removed the VSR $d(n_{i-1})$ from the simulation, taking note of the x-coordinate x_left of its leftmost voxel, (2) used the development function to develop the VSR to $d(n_i)$ and (3) placed the developed VSR in the simulation right above the terrain in a position such that its leftmost part was at x_left. We stopped the simulation after 210 s (simulated time), took note of the run distance Δx, as the difference between the initial and final x-coordinate of the center of mass of the VSR, and used Δx as fitness.

We removed and added (i.e., *re-spawn*) the VSR just before and right after each development because the new voxel might have been added in positions that conflict with the current posture of the robot (e.g., under a foot, "inside" the terrain). As a consequence, each development step led to a re-spawning of the VSR, where its gait was interrupted.

We characterized the performance of the representations in conjunction with two development schedules, both encompassing 14 stages: *early* development $S_\text{early} = (10, 20, 30, \dots, 120, 130)$, resembling biological development, and *uniform* development $S_\text{uniform} = (15, 30, 45 \dots, 180, 195)$, accounting for continuous growth (numbers are in s). In addition, to have a baseline for comparisons, we also employed a non-developmental schedule $S_\text{no-devo} = \emptyset$: in this case VSRs were initialized according to the initial development, and no voxels were ever added to them. To ensure fairness in terms of re-spawning (as such events could slow down the VSR), we interrupted the gait to lift the VSR in the air according to S_uniform, even though no development was occurring.

Concerning the initial size n_0 of the VSRs, we aimed at being as fair as possible, since larger robots could, in principle, benefit from having more power. Hence, we chose $n_{0,\text{early}} = 6$, $n_{0,\text{uniform}} = 8$, and $n_{0,\text{no-devo}} = 14$ in order to have approximately the same weighted average VSR size during the simulation ($\bar{n}_\text{early} \approx 14.7$, $\bar{n}_\text{uniform} = 14.5$, and $\bar{n}_\text{no-devo} = 14$). In other words, the chosen values of n_0 resulted in all VSRs having approximately the same integral of size over the simulations.

For each of the $4 \cdot 3$ combinations of representation and schedule, we performed 10 independent, i.e., based on different random seeds, evolutionary optimizations, obtaining a total of 120 runs. When comparing results of pairs of combinations, we performed the Mann-Whitney U test, after having verified the proper requirements, with the null hypothesis of equality of the means—we report the p-values. Note that, since we performed multiple pairwise comparisons simultaneously, we applied the required Bonferroni corrections for evaluating the significance of results.

We used 2D-VSR-Sim [25] for the simulation setting all parameters to default values; in particular, we set the simulation time interval $\Delta t = \frac{1}{60}$ s. We made the code for the experiments publicly available at `https://github.com/giorgia-nadizar/VSREvoDevo`.

4.1 Results and discussion

The experimental results are reported in Figures 3 to 5. The most high level finding of our experiments is displayed in Figure 3, which depicts the distributions of the fitness Δx of the best individuals at the end of evolution for each representation and schedule. From such plots, we are able to compare the outcomes deriving from S_early and S_uniform: given the distributions and the p-values, we can conclude that, in general, early development is not worse than uniform development, and that for phase controllers it is significantly better. We can hence infer that development in artificial life is somehow similar to development in real life, and, even though creatures that continuously grow end up being larger, such trait does not really benefit their overall performance. We hypothesize that early development is more effective as the optimization of the controller of the agent is favored by the fact that the brain is able to interact with a fixed body for a longer amount of time (the last development stage, which is longer than the other ones). Therefore, we speculate that evolution finds a way to optimize the controller for the last body, since being optimal during this longer stage could result in more distance gain, hence in higher fitness. On the other hand, continuous growth seems to hinder brain development, as evolution cannot find an optimal controller to almost equally fit all the bodies the controller interacts with.

Comparison against no development. Reasoning further on Figure 3, it is interesting to also compare the results achieved by non-developing VSRs. From the previous assumptions, we would expect such outcomes to be significantly better than both early and uniformly developed VSRs, as in this case the brain is optimized for just one body. However, the results shown in the plots are less clear, as there is no evident winner among the non-developed, early-developed, and uniformly-developed VSRs. We explain these mixed findings by reasoning on two factors: the re-spawning and the lack of development. Concerning the re-spawning, it likely slowed VSRs down, even though they were not really developing, preventing them from exhibiting a fluid and effective gait. In addition, we speculate that the lack of actual development could have been detrimental for the overall performance achieved, consistently with [21].

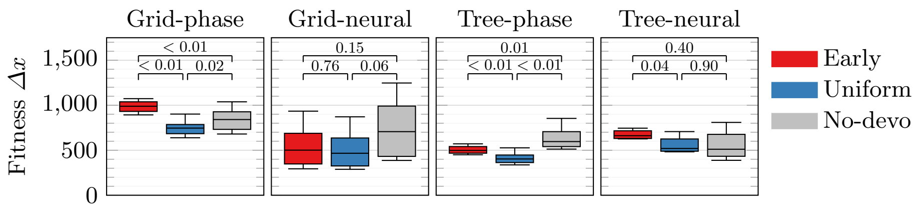


Fig. 3: Box plots of the fitness Δx of the best development functions at the end of evolution for different representations (plot columns) and development schedules (color). p-values are shown above pairs of boxes. We consider $\alpha = 0.05/3 \approx 0.017$ for statistical significance, due to the Bonferroni correction.

Analysis of VSR velocity. To gain further insight on the obtained results, we also measured the velocity of VSRs along development: we define $v_{x,i}$ as the average velocity achieved by a VSR in its i-th stage of development. Figure 4 depicts the distribution of the velocity of VSRs during the last development stage for each representation and schedule. These outcomes are aligned with the previous findings: early development yields to not slower VSRs compared to uniform development, whereas non developed VSRs tend to exhibit fuzzier relationships with the others. However, more interesting conclusions can be drawn if we interpret Figure 4 taking into consideration that VSRs in the last (i.e., 14-th) stage have different sizes according to the chosen schedule, namely $n_{14,\text{early}} = 19$, $n_{14,\text{uniform}} = 21$, and $n_{14,\text{no-devo}} = 14$. In fact, we can observe that, surprisingly, larger VSRs do not correspond to higher velocity. To explain this, we resort to the same motivation provided before. Namely, early development generates VSRs that are mostly optimized for the last stage of their lives, not only because the last robots are bigger in size, but also because the body-brain interaction is longer.

To conclude the velocity analysis, we provide in Figure 5 a display of the $v_{x,i}$ throughout the simulation. From these plots, the previously laid hypothesis as to why early development is more successful seems to be confirmed. In addition, we can note that both early and uniform development follow a general growing trend for the velocity, suggesting that also size plays a significant role in the achievement of good performance at locomotion.

Comparison among representations. Last, it is interesting to reason on the different outcomes produced by the representations we experimented with, both in terms of fitness Δx (Figure 3) and velocity $v_{x,14}$ in the last development stage (Figure 4). We summarize the outcomes of the statistical significance tests between pair of representations in Table 1, where we use colored dots (the color encoding the development schedule) to annotate distributions

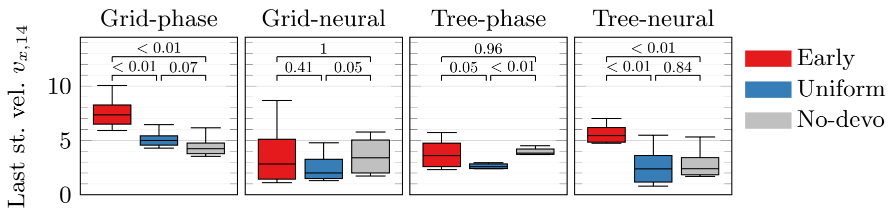


Fig. 4: Box plots of the last stage velocity $v_{x,14}$ of the best development functions at the end of evolution for different representations (plot columns) and development schedules (color). p-values are shown above pairs of boxes. We consider $\alpha = 0.05/3 \approx 0.017$ for statistical significance, due to the Bonferroni correction.

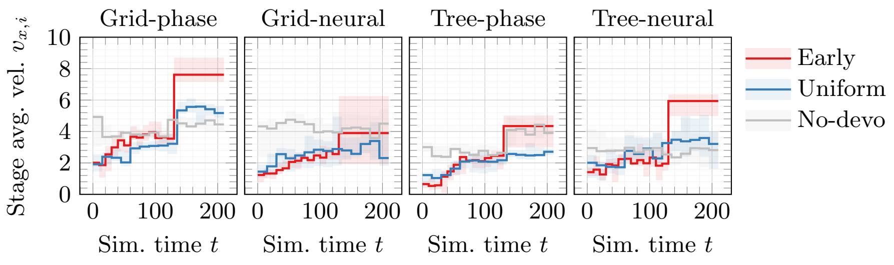


Fig. 5: Average velocity $v_{x,i}$ (median with lower and upper quartiles across the 10 repetitions) of the developing VSRs at different stages during the simulation, for different representations (plot columns) and development schedules (color).

which are significantly different (p-value $< \alpha = 0.05/6 \approx 0.008$). From the table we can immediately notice that the results obtained without development almost never significantly depend on the employed representation, whereas the outcomes coming from early or uniform development show more interesting variety among representations. Taking into account Figures 3 and 4 and Tables 1a and 1b, we can conclude that the grid-phase representation is in general not worse than the others, and is significantly better for a subset of schedules and representations. We speculate that this could depend on two factors: the direct representation and the superiority of a phase controller to a neural controller for developing VSRs. Namely, we hypothesize that it is easier for evolution to find suitable phases values for a growing body, than optimizing a single MLP to fit all voxels of a wide gamut of bodies.

To conclude the discussion on the experiments we show in Figure 6 an example of a developing VSR obtained at the end of one evolution with the grid-phase representation and the uniform schedule: in the figure, each frame shows the VSR during a developing stage. The corresponding video is available at `https://youtu.be/DD4D2OEH1sA`.

	GP	GN	TP	TN
GP	–	🔴	🔴🔵⚪	🔴🔵
GN	🔴	–		
TP	🔴🔵⚪		–	🔴🔵
TN	🔴🔵		🔴🔵	–

(a) Fitness Δx.

	GP	GN	TP	TN
GP	–	🔵	🔴🔵	🔴🔵
GN	🔵	–		
TP	🔴🔵		–	🔴
TN	🔴🔵		🔴	–

(b) Last stage vel. $v_{x,14}$.

🔴 Early
🔵 Uniform
⚪ No-devo

Table 1: Statistical significance results for different metrics and representations. Each cell is annotated with a dot if the p-value on the two representations with the same schedule is $< \alpha = 0.05/6 \approx 0.008$ (due to the Bonferroni correction).

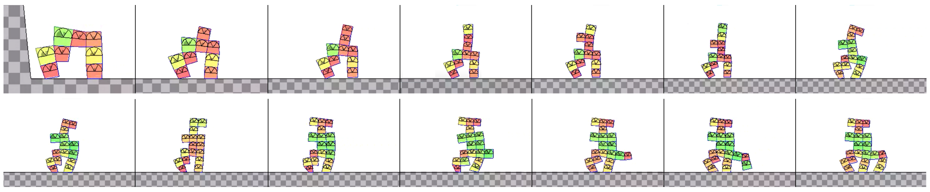


Fig. 6: View of a developing VSR (uniform schedule with the grid-phase representation). Each image is taken $\approx 0.5\,\mathrm{s}$ after a voxel has been added to the VSR body, to leave time to the robot to fall and exhibit its posture on the ground. Voxels color encodes the ratio between its current area and its rest area (red for contraction, yellow for rest, green for expansion).

5 Concluding remarks

In this work, we investigated the effects of different schedules for morphological development of Voxel-Based Soft Robots (VSRs), a kind of modular soft robots. To this extent, we evolved development functions, i.e., functions which can build and extend VSRs bodies and controllers, to generate and develop VSRs capable of successfully performing the task of locomotion. Aiming at achieving general results, our study encompassed four representations for development functions, based on different combinations of body-brain encodings, and we also included non-developing VSRs, as a baseline. Our experimental findings show that, similarly to living organisms, VSRs benefit from early development, whereas continuous growth tends to hinder the overall performance of the agent. In particular, we have noticed that, despite having more power, bigger VSRs deriving from continuous growth are not more effective than the early developed smaller ones, concluding that an appropriate development schedule plays a key role in determining the effectiveness of a VSR.

As an extension of this work, it might be of interest to experiment with additional representations, e.g., based on neural cellular automata, and to take into account environmental feedback [24] in development. Moreover, the concept of early mortality [16] could be introduced in our framework.

Acknowledgements

The experimental evaluation of this work has been partially supported by a Google Faculty Research Award granted to E.M.. K.M. was supported by the Hybrid Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organization for Scientific Research (`https://www.hybrid-intelligence-centre.nl`), grant number 024.004.022.

References

1. Liknes, E.T., Swanson, D.L.: Phenotypic flexibility of body composition associated with seasonal acclimatization in passerine birds. Journal of Thermal Biology **36**(6) (2011) 363–370
2. Fusco, G., Minelli, A.: Phenotypic plasticity in development and evolution: facts and concepts (2010)
3. Kelly, S.A., Panhuis, T.M., Stoehr, A.M.: Phenotypic plasticity: molecular mechanisms and adaptive significance. Comprehensive Physiology **2**(2) (2011) 1417–1439
4. Sapolsky, R.M.: Behave: The biology of humans at our best and worst. Penguin (2017)
5. Thomis, M.A., Towne, B.: Genetic determinants of prepubertal and pubertal growth and development. Food and nutrition bulletin **27**(4_suppl5) (2006) S257–S278
6. Butte, N.F., Garza, C., de Onis, M.: Evaluation of the feasibility of international growth standards for school-aged children and adolescents. The Journal of nutrition **137**(1) (2007) 153–157
7. Lui, J.C., Baron, J.: Mechanisms limiting body growth in mammals. Endocrine reviews **32**(3) (2011) 422–440
8. Batsis, J.A., Buscemi, S.: Sarcopenia, sarcopenic obesity and insulin resistance. In: Medical Complications of Type 2 Diabetes. IntechOpen (2011)
9. Jones, J.H.: Primates and the evolution of long, slow life histories. Current Biology **21**(18) (2011) R708–R717
10. Thompson, R.A., Nelson, C.A.: Developmental science and the media: Early brain development. American Psychologist **56**(1) (2001) 5
11. Nadizar, G., Medvet, E., Pellegrino, F.A., Zullich, M., Nichele, S.: On the effects of pruning on evolved neural controllers for soft robots. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. (2021) 1744–1752
12. Nadizar, G., Medvet, E., Huse Ramstad, H., Nichele, S., Pellegrino, F.A., Zullich, M.: Merging pruning and neuroevolution: towards robust and efficient controllers for modular soft robots. The Knowledge Engineering Review **37** (2022)
13. Prabhu, S.G.R., Seals, R.C., Kyberd, P.J., Wetherall, J.C.: A survey on evolutionary-aided design in robotics. Robotica **36**(12) (2018) 1804–1821
14. Lindenmayer, A.: Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. Journal of theoretical biology **18**(3) (1968) 280–299
15. Stanley, K.O.: Compositional pattern producing networks: A novel abstraction of development. Genetic programming and evolvable machines **8**(2) (2007) 131–162

16. Miras, K., Cuijpers, J., Gülhan, B., Eiben, A.: The impact of early-death on phenotypically plastic robots that evolve in changing environments. In: ALIFE 2021: The 2021 Conference on Artificial Life, MIT Press (2021)
17. Miras, K., Ferrante, E., Eiben, A.: Environmental regulation using Plasticoding for the evolution of robots. Frontiers in Robotics and AI **7** (2020)
18. Daudelin, J., Jing, G., Tosun, T., Yim, M., Kress-Gazit, H., Campbell, M.: An integrated system for perception-driven autonomy with modular robots. Science Robotics **3**(23) (2018) eaat4983
19. Doursat, R., Sayama, H., Michel, O.: Morphogenetic engineering: toward programmable complex systems. Springer (2012)
20. Kriegman, S., Cheney, N., Corucci, F., Bongard, J.C.: Interoceptive robustness through environment-mediated morphological development. arXiv preprint arXiv:1804.02257 (2018)
21. Kriegman, S., Cheney, N., Bongard, J.: How morphological development can guide evolution. Scientific reports **8**(1) (2018) 1–10
22. Bongard, J.: Morphological change in machines accelerates the evolution of robust behavior. Proceedings of the National Academy of Sciences **108**(4) (2011) 1234–1239
23. Hiller, J., Lipson, H.: Automatic design and manufacture of soft robots. IEEE Transactions on Robotics **28**(2) (2012) 457–466
24. Walker, K., Hauser, H., Risi, S.: Growing simulated robots with environmental feedback: An eco-evo-devo approach. GECCO '21, New York, NY, USA, Association for Computing Machinery (2021) 113–114
25. Medvet, E., Bartoli, A., De Lorenzo, A., Seriani, S.: 2D-VSR-Sim: A simulation tool for the optimization of 2-D voxel-based soft robots. SoftwareX **12** (2020)
26. Medvet, E., Bartoli, A., De Lorenzo, A., Seriani, S.: Design, Validation, and Case Studies of 2D-VSR-Sim, an Optimization-friendly Simulator of 2-D Voxel-based Soft Robots. arXiv (2020) arXiv–2001
27. Corucci, F., Cheney, N., Giorgio-Serchi, F., Bongard, J., Laschi, C.: Evolving soft locomotion in aquatic and terrestrial environments: effects of material properties and environmental transitions. Soft robotics **5**(4) (2018) 475–495
28. Talamini, J., Medvet, E., Bartoli, A., De Lorenzo, A.: Evolutionary synthesis of sensing controllers for voxel-based soft robots. In: Artificial Life Conference Proceedings, MIT Press (2019) 574–581
29. Medvet, E., Bartoli, A., De Lorenzo, A., Fidel, G.: Evolution of distributed neural controllers for voxel-based soft robots. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference. (2020) 112–120
30. Medvet, E., Bartoli, A., Pigozzi, F., Rochelli, M.: Biodiversity in evolved voxel-based soft robots. In: Proceedings of the Genetic and Evolutionary Computation Conference. (2021) 129–137
31. Ferigo, A., Iacca, G., Medvet, E.: Beyond Body Shape and Brain: Evolving the Sensory Apparatus of Voxel-Based Soft Robots. In: Applications of Evolutionary Computation. EvoApplications. Volume 12694., Springer, Cham (2021) 210–226
32. Rothlauf, F., Goldberg, D.E.: Redundant representations in evolutionary computation. Evolutionary Computation **11**(4) (2003) 381–415
33. Vanneschi, L., Castelli, M., Manzoni, L.: The k landscapes: a tunably difficult benchmark for genetic programming. In: Proceedings of the 13th annual conference on Genetic and evolutionary computation. (2011) 1467–1474
34. Pagliuca, P., Nolfi, S.: The dynamic of body and brain co-evolution. Adaptive Behavior (2020) 1059712321994685