Analysis of Lamarckian Evolution in Morphologically Evolving Robots

Milan Jelisavcic[1], Rafael Kiesel[1], Kyrre Glette[2], Evert Haasdijk[1] and A. E. Eiben[1]

[1]Vrije Universiteit Amsterdam, Dept. of Computer Science, Amsterdam, The Netherlands
[2]University of Oslo, Department of Informatics, Oslo, Norway
m.j.jelisavcic@vu.nl

Abstract

Evolving robot morphologies implies the need for lifetime learning so that newborn robots can learn to manipulate their bodies. An individual's morphology will obviously combine traits of all its parents; it must adapt its own controller to suit its morphology, and cannot rely on the controller of any one parent to perform well without adaptation. This paper investigates the practicability and benefits of Lamarckian evolution in this setting. Implementing lifetime learning by means of on-line evolution, we first establish the suitability of an indirect encoding scheme that combines Compositional Pattern Producing Networks (CPPNs) and Central Pattern Generators (CPGs) as a relevant learner and controller for open-loop gait controllers. We then analyze a Lamarckian set-up and the effect of the parental genetic material on the early convergence to good locomotion performance.

Introduction

A robot's behaviour is the result of the interaction between its morphology, controller, and environment (Pfeifer and Iida, 2004). Evolutionary robotics offers a methodology to consider the development and adaptation of robot morphology and control holistically (Eiben and Smith, 2015). Simultaneous evolution of morphology and control was introduced with Sims' simulated virtual creatures (Sims, 1994) and has been investigated without regards to physically producible results many times since then (Lehman and Stanley, 2011; Cheney et al., 2013, among others). Lipson and Pollack (2000) first demonstrated that this approach is also applicable in systems where the final (i.e., after evolution has run its course) results are realised and evaluated as actual physical robots, with substantial research revisiting this approach (e.g., Hornby et al., 2003; Auerbach et al., 2014).

The simultaneous development of robot morphologies and control systems is a difficult task, and we have only seen relatively simple results so far, as noted by Cheney et al. (2016). Some of the difficulty is due to the increased dimensionality of the search, but a more pernicious aspect may be the increased ruggedness of the search space: a small mutation in the morphology can easily offset the performance of the controller-body combination found earlier. Cheney et al. (2016) illustrate this by casting the morphology as a physical interface between controller and environment; the variation operators that generate a new individual can then be seen as "scrambling" this interface. An obvious remedy would be to allow the controller to adjust to the new morphology, on a different timescale from the morphological changes—i.e., to enable lifetime learning for new robot bodies.

There are two principal options for evolution to exploit lifetime learning: Baldwinian and Lamarckian evolution. The former does not directly store the results of lifetime learning phase, only the resulting fitness values. Lamarckian evolution, by contrast, does explicitly store the locally learned improvements in the individual genomes, so that lifetime learning can directly accelerate the evolutionary process and vice versa (Ackley and Littman, 1994). While this mechanism has largely not been seen as a correct description of biological evolution, some recent research has reported a Lamarckian type of evolution in nature (Dias and Ressler, 2014).

In this paper, we research the benefits of Lamarckian evolution for control when the morphology of robots evolves without central oversight. This means that we are principally interested in a setting where the robots evolve and learn online as proposed by Eiben and Smith (2015), without any central oversight in a physical habitat where a population of robots operates. For these initial investigations, we focus on the development of locomotion, although we think that lifetime learning is particularly important to achieve more complex behaviours.

Robotic locomotion requires the creation of rhythmic patterns which satisfy multiple constraints: generating stable and energy efficient forward motion, and coping with changes in the environment or the organism (Sproewitz et al., 2008). In evolutionary robotics, simple but efficient control methods can be found in tables of control sequences (Bongard et al., 2006) and spline-based cyclical patterns (Kober and Peters, 2009). A more nature-inspired approach exploits central pattern generators (CPG), which model neural circuitry that outputs cyclic patterns as found in vertebrates (Sproewitz et al., 2008). In this case, robot actua-

tors are controlled by the signal generated by coupled synchronised CPGs, allowing synchronised movement. Clune et al. (2011) used controllers based on more general artificial neural networks to develop controllers for efficient locomotion. They used the HyperNEAT indirect encoding which is based on evolving a Compositional Pattern Producing Network (CPPN) that encodes a function to determine connection weights in the *substrate* artificial neural net that actually controls the robot (Stanley et al., 2009).

Most of the mentioned research considers the *off-line* development of locomotive controllers, i.e., controller optimisation as a separate phase before deployment with a view to developing controllers that remain fixed once deployed. Weel et al. (2017) considered *on-line* gait learning, where the controller is adapted to the robot's task environment during deployment. Weel et al. showed that spline-based controllers with the RL PoWER algorithm provide efficient autonomous *on-line* gait learning capabilities. Jelisavcic et al. (2016) showed that RL PoWER is very similar to an on-line $(\mu + 1)$ evolutionary strategy ($(\mu + 1)$ ES).

Note, that the methods for gait development mentioned above are all evolutionary. This may cause some confusion, as we consider them in the role of lifetime learning in an overarching evolutionary process where the morphologies evolve. Thus, we consider systems comprising of *two* adaptive processes. At the highest level, the robot morphologies evolve: a new individual implies a unique body that is the result of applying variation operators to its parents' genomes. We have argued that this necessitates a second adaptive process of lifetime learning that operates at a different timescale to optimise the individual's controller to suit its body and environment. We consider on-line evolution as a relevant technique for this second phase—it can be seen as an instance of reinforcement learning (Haasdijk et al., 2012). So, reiterating: there are two interleaved evolutionary processes: one that adapts morphologies and another that adapts controllers, and the latter implements lifetime learning for the former.

To implement Lamarckian evolution of morphology and control, the robot's genome must encode the robot's controller as well as its morphology. Lifetime learning schemes that directly encode parameters for particular actuators make less sense than indirect encodings: it is difficult or even impossible–e.g., when expression of the morphology is non-deterministic or depends on the environment (e.g., Liu and Winfield, 2011)–to identify the mapping of controller parameters to actuators in a new morphology where some actuators may no longer occur and new ones have appeared. An indirect encoding scheme such as HyperNEAT would not be hampered in this way: a different layout of actuators would merely imply a change in input values when expressing the genome. Implementations that do encode the robot controllers directly exclude recombination operators and have deterministic morphogenesis and therefore are less susceptible to this issue (Lipson and Pollack, 2000). Several approaches to co-evolution of morphology and control with indirect and coupled body-brain encodings exist, e.g., based on graphs or L-systems (Sims, 1994; Hornby et al., 2003), where the control components are generated along with the morphology.

Until recently, there has been little research into the effect of Lamarckian set-ups combining morphological evolution and lifetime learning of control. Jelisavcic et al. (2017) report preliminary findings that indicate that such a Lamarckian set-up with CPG-based controllers and an indirect encoding using CPPNs can improve performance. This paper builds on these encouraging findings in two directions.

First, we investigate whether an *indirect* encoding scheme based on CPPNs and CPGs can provide efficient on-line gait learning. As a benchmark, we use the *directly* encoded $(\mu + 1)$ ES approach that Jelisavcic et al. (2016) showed to be a successful technique for the on-line evolution of gaits. We test both learning approaches on two controller architectures: one based on splines, and the other based on CPGs.

Next, with the indirect encoding scheme enabling Lamarckian evolution, we compare the performance of lifetime learning in a Lamarckian and a non-Lamarckian set-up and analyse the results in terms of performance and the persisting amount of parental genetic material.

Method

The first set of experiments[1] in this paper establishes an indirect encoding scheme for on-line development of good gaits. As a baseline, we consider a $(\mu + 1)$ ES with a spline-based control scheme. Jelisavcic et al. (2016) showed that this is a suitable scheme for developing open-loop controllers for locomotion. We compare its performance to a scheme that evolves CPPNs to configure controllers based on CPGs or on splines. We also compare an implementation of $(\mu + 1)$ ES to configure CPGs to ensure that any difference in performance can be accounted for. These methods are tested on a number of hand-designed shapes.

The second set of experiments investigates the effect of Lamarckian evolution with the CPPN-CPG scheme. We generate offspring from the hand-designed shapes considered in the first set of experiments (using random selection). We then run CPPN-CPG on each offspring with the population initialised randomly (the non-Lamarckian case) or seeded with CPPNs from both parents (the Lamarckian case). This experimental set-up allows us to thoroughly study the effects on the lifetime learning process without confounding effects of selection or stochasticity in the morphological evolution. Implementation details are provided in the following subsections.

[1] Code for the experiments and supplementary material is available on-line at `http://tinyurl.com/y9cbbytm`

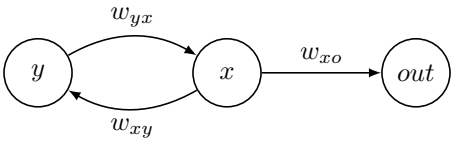

Figure 1: A differential oscillator with output node as used in the CPG controller.

Controller Schemes

We compare two different controller schemes in this paper: CPG-based and spline-based controllers. Both controller schemes are open-loop and produce control signals which are converted to angular positions for each joint.

CPG-based Controller The main components of the CPG controllers are differential oscillators. Each oscillator is defined by two neurons that are recursively connected as shown in Fig. 1. These generate oscillatory patterns by calculating their activation levels x and y according to the following differential equation:

$$\dot{x} = w_{yx}y + \mathit{bias}_x$$
$$\dot{y} = w_{xy}x + \mathit{bias}_y$$

with w_{xy} and w_{yx} denoting the weights of the connections between the neurons; bias_x and bias_y are parameters of the neurons. If w_{yx} and w_{xy} have different signs the activation of the neurons x and y is periodic and bounded. An oscillator's x node is connected to a linear output neuron that in turn connects to the robot's active hinge. Output neurons use the following activation function:

$$f(x) = (w_{xo} \cdot x - \mathit{bias}) \cdot \mathit{gain}.$$

with x the activation level from the oscillator, w_{xo} the weight of the connection between oscillator and output node and *bias* and *gain* parameters. Each active joint in the robot body is associated with an oscillator and connected to it through an output neuron that determines the joint's angle.

The oscillators of neighbouring hinges (i.e., hinges separated by a single component) are interconnected by means of weighted connections between their x neurons. This results in a chain-like neural network of differential oscillators that extends across the robot body, as illustrated in Fig. 2. The learning algorithm then optimises the connection weights, the node biases, and the gain levels of the output nodes.

Spline-based Controller These controllers are based on cyclical splines that describe the joint control output at any point in time. A cyclic spline is an interpolation function that is defined using a set of n control points. Each control point is defined by (t_i, α_i) where t_i represents time and α_i the corresponding value. $t_i \in [0, 1]$ is defined as

$$t_i = \frac{i}{n}, \forall i = 0, \ldots, (n-1)$$

and $\alpha_i \in [0, 1]$. An additional control point (t_n, α_n) is defined to enforce that the last value is equal to the first, i.e., $\alpha_0 = \alpha_n$, and so enforce cyclic splines. A spline-based controller defines one such spline for each active joint in the robot body, and the learning algorithm operates on the control points for each spline. As in Shen et al. (2012), we follow a scheme of incremental complexification of the spline by initializing the number of control points, n, to 4 and incrementing by 1 every 100 evaluations of the learning algorithm. New control points are inserted such a way that the spline shape does not change.

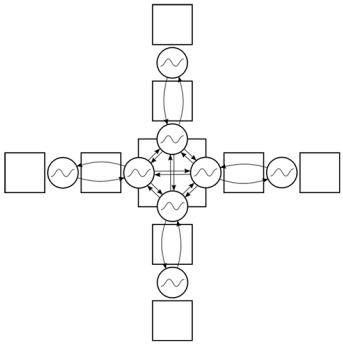

Figure 2: Schematic view of the CPG network generated for the body shown in the top-left panel of Fig. 3. The rectangular shapes indicate passive body parts, the circles show active hinges, each with their own differential oscillator, and the arrows indicate the connections between the oscillators.

Learning Schemes

We employ two different lifetime learning schemes in this paper; NEAT to evolve CPPNs, enabling the Lamarckian experiments, and $(\mu + 1)$ ES, which has proved efficient in our earlier experiments with on-line learning (Jelisavcic et al., 2016), and thus serves as a benchmark approach. Both learning approaches test the individual controllers in a time-sharing scheme, as is common in on-line evolutionary robotics, and employ an internal population size of 10. The target of both learning algorithms is to maximise the forward locomotive performance of the robots.

Learning with CPPNs Stanley et al. (2009) proposed HyperNEAT, an indirectly encoded evolutionary algorithm for neural networks. The idea behind HyperNEAT is to assign the nodes in a substrate neural network a location in an n-dimensional hypercube. The assigned relative positions should in some way reflect a relationship between the nodes. allowing the algorithm to exploit the geometry of the

problem. The coordinates of two nodes in the hypercube are then input values for a CPPN, which outputs a value for the weight of their connection. The CPPN evolves using NEAT (Stanley and Miikkulainen, 2002) so that the substrate network's performance is optimised.

In this paper, the evolution of CPPNs follows (Stanley et al., 2009), with some modifications: it uses binary tournament selection for two parents within a species if there is more than one individual in that species. If there is only one individual in a species, the best individual of a random other species is selected as the second parent. Finally, the implementation uses elitism, transferring the best 10% of the individuals to the next population.

The CPG nodes are positioned in a three-dimensional hyperspace. Two dimensions are the relative position of the active hinges in the robot morphology as proposed by Haasdijk et al. (2010). Such *modular differentiation* allows specialisation of the active hinge's movements depending on its relative position in the robot. The hinge coordinates are obtained from a top-down view of the robot body. Thus, two coordinates of a node in the CPG controller correspond to the relative position of the active hinge it is associated with. The value of the third (z) coordinate varies according to the type of node and the kind of connection: for connections within an oscillator, X and y nodes have $z = 1$, respectively $z = -1$. For connections between neighbouring oscillators, $z = 0$. The CPPNs have six inputs denoting the coordinates of a connection's source and target and three outputs: the weight of the connection and the bias and gain for the target node. For inter-oscillator connections (when $z = 0$), the gain and bias outputs are ignored.

For spline-based controllers, the CPPNs must output the α_i value for a given t_i for each spline in the controller. Obviously, t_i must be one of the inputs for the CPPN. As before, two further inputs correspond with the position of the active hinge associated with the spline. Thus, CPPNs for splines have three inputs: the coordinates of the active hinge and t_i. They have one output: α_i.

Learning with $(\mu + 1)$ ES Using a straightforward direct encoding, the genotype is a vector of real values corresponding to all control points for the spline-based controllers, and to the weights of the connections, the biases and the gain parameters of the nodes for CPG-based controllers. For the experiments in this paper, we use binary tournament selection and a self-adjusting σ value *cf.* Jelisavcic et al. (2016). Further details can be found in the source code and accompanying documentation.

Lamarckian Set-up

With lifetime learning by means of an on-line evolutionary algorithm as in this research, each robot carries an internal population of controllers that evolve during the robot's lifetime. In such a system, a simple but effective implementation of Lamarckian evolution is to seed an individual's population from that of its parents. We test two variants of seeding an offspring's population. One option initialises the NEAT population with the best 5 CPPNs from each parent, this variant is labeled as $5 + 5$ in the following sections. The second option only takes 3 CPPNs from each parent and randomly initialises the remaining 4 as for standard NEAT (labeled as $3 + 3 + 4$). These randomly initialized networks only contain the input and output neurons and connections from every input to every output neuron with randomly initialized weights and neuron parameters.

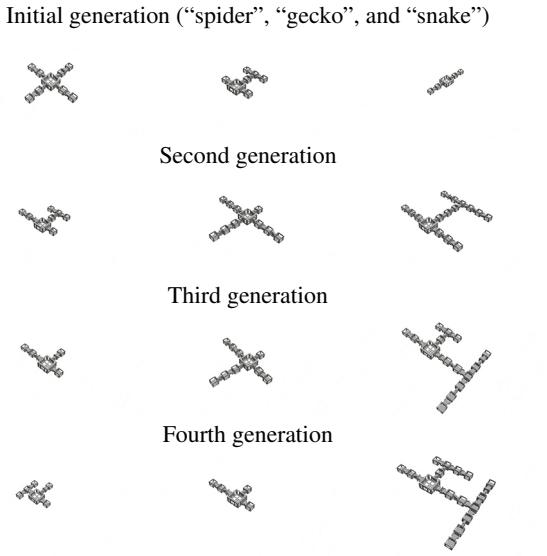

Figure 3: The morphologies used in this study. The top row shows the smallest versions of the three hand-designed initial morphologies. The following rows of robots result from recombination and mutation of the previous generations.

NEAT requires some bookkeeping of the complexification process that occurs by adding and removing nodes and connections, and this is implemented by means of innovation numbers that uniquely identify inserted material. When combining individuals from two separate NEAT runs, the innovation numbers must be updated so that no conflicts occur. Offsetting the innovation numbers from one of the two parents proved a convenient method to achieve this, allowing the CPPNs from both parents to be (re)combined in the offspring's population.

The Robots

The robots and their genetic representation are based on RoboGen (Auerbach et al., 2014). In this study, we use a subset of those 3D-printable components: *fixed bricks*, a *core component*, and *active hinges*. For the experiments in this paper, the robots are simulated in a custom simulator based on Gazebo. Each robot's genotype describes its layout and consists of a tree structure with the root node represent-

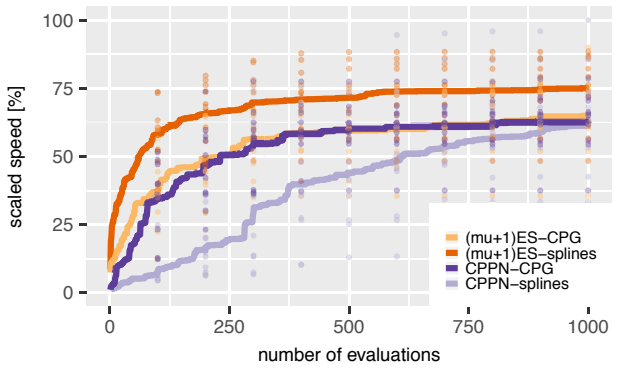

Figure 4: Learner and controller performances for the three size variations of the "spider" morphology. To adjust for the higher speeds obtained by the larger robots, the performances are normalised and combined into a single plot. The lines show the averaged best result and the dots indicate the performance of the separate runs.

ing a core module from which further components branch out. Similar to genetic programming, recombination is implemented by replacing a sub-tree in one parent with a randomly selected sub-tree from the other parent. There are some restrictions with which the result must comply, e.g., the body cannot intersect with itself. If these restrictions are violated, the recombination operation fails.

For these experiments, we take three manually designed morphologies as representing an ancestral population, shown in the top row of Fig. 3. We also vary the size of each shape by adding hinge-brick modules at the end of each extremity, giving us a total of nine ancestors: the ones shown and six larger variants with one, respectively two modules added to each extremity. The first set of offspring ("second generation") consists of three robots generated by recombining two parents from this ancestral population. A further iteration ("third generation") consists of three offspring of the second generation robots, and the fourth generation comprises of three offspring from the third generation.

Results

This section presents the results from the two main experiments: First, learner-controller combinations are tested on the nine ancestral morphologies, represented by the top row of Fig. 3. We then test the effect of a Lamarckian approach using an indirect encoding on the offspring robots, shown in the three bottom rows of Fig. 3. Further, we analyze the results from the Lamarckian experiments with respect to genetic material retention and performance increase.

Lifetime Gait Learning Comparison

Each morphology was tested in three sizes, and learning was performed in ten replicate runs. In order to gain some intuition on the four tested learner-controller schemes, performances are normalised as a percentage of the overall best result with all four tested schemes for a given robot morphology and size. The normalised performances are then averaged over all runs, yielding a combined performance for a given morphology type. The results from the "spider" morphology are shown in Fig. 4. Plots for the other morphologies are available in the supplementary material.

Table 1: p-values from Dunn's test comparing performance between the different learner and controller combinations. MPO denotes $(\mu + 1)$ ES, SPL denotes splines.

	MPO-SPL	MPO-CPG	CPPN-SPL
MPO-CPG	0.0890		
CPPN-SPL	0.0001	0.0065	
CPPN-CPG	0.0102	0.1654	0.0655

Table 2: p-values from Dunn's test comparing performance between the baseline and the two Lamarckian scenarios at two points in the learning process.

Evaluation	5+5	3+3+4	non-parent
250	0.017	0.013	1.00
1000	1.00	1.00	1.00

A Kruskal-Wallis test shows that there is a statistically significant difference between the learner-controller combinations in the last evaluation ($H(2) = 15.63, p = 0$). This was followed by Dunn's test to determine which scenarios differ, and the results are shown in Tab. 1. The CPPN-splines approach performs worst, while the $(\mu + 1)$ ES-splines approach is better than the CPPN-CPG approach (with $p < 0.05$). There is no statistical difference between the performance of CPPN and $(\mu + 1)$ ES when applied to CPG controllers.

Lamarckian vs. Non-Lamarckian Performance

The results from running new rounds of learning on the morphology offspring are shown in Fig. 5. The Lamarckian approach is compared to learning based on re-initialized controllers. We also include an approach where we initalize the learning with a control system not stemming from the parent morphologies, but instead the remaining morphology of the previous generation. This comparison is included for control purposes – to observe whether there is a performance change in learning based on controllers from parental morphologies rather than a randomly picked morphology. The plots show the best performance over time: the individual dots indicate the results of the separate replicate runs at intervals of 100 evaluations and the lines show the average for each learning scheme.

We used the Kruskal-Wallis test to determine whether the

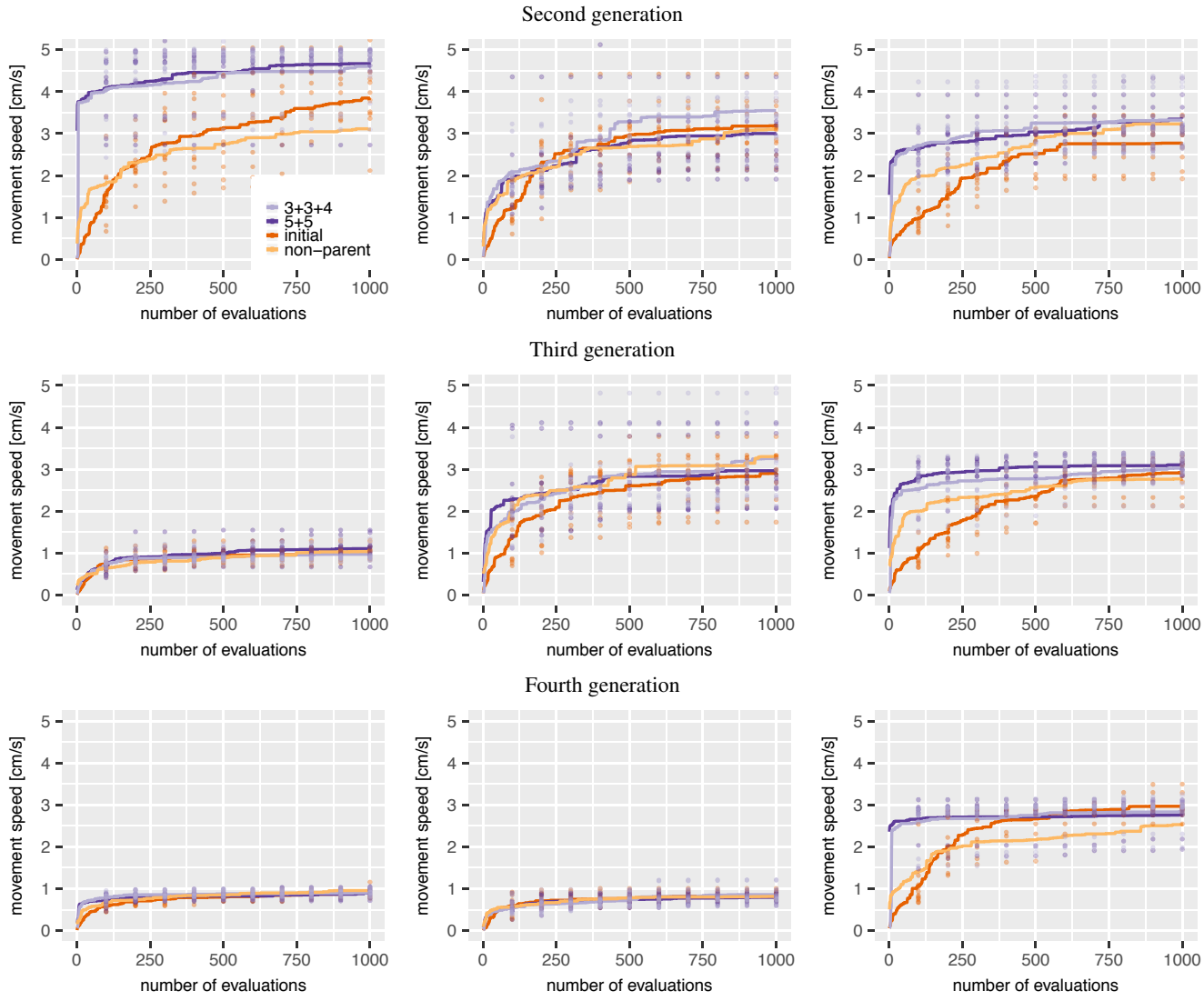

Figure 5: Results from running lifetime learning on offspring with new morphologies, using either re-initialized CPPNs (*initial*), CPPNs from the non-parent robot, 5 CPPNs from each parent (*5+5*), or a mix between re-initialized and 3 CPPNs from each parent (*3+3+4*). All learning methods are run 10 times, lines show the mean and dots show the individual run results. The plots are arranged coincidentally with the images in Fig. 3.

differences in performance are statistically significant between the scenarios at evaluations 250 ($H(2) = 13.57, p = 0$) and 1000 ($H(2) = 1.53, p = 0.67$). This was followed by Dunn's test to determine which scenarios differ, and the results are shown in Tab. 2. Note that the table does not show the comparison between the two Lamarckian scenarios, as no significant differences were found between these. The performance differences between both of the Lamarckian and the non-Lamarckian scenario are significant at evaluation 250. After 1000 evaluations a performance difference seems to persist, but more samples would be needed to statistically confirm this.

Genetic Material vs. Performance Increase

We further seek to analyse the factors causing the differences in lifetime learning between the non-Lamarckian and the Lamarckian set-ups. In order to do this, we investigate the correlation between the amount of remaining parental genetic material as lifetime learning progresses and the performance increase of the Lamarckian set-up.

We calculated the percentage of parental genetic material for the currently best performing CPPNs in the (5+5) approach, that is, the number of nodes and connections present from the parental CPPNs with which the population was seeded as a percentage of the total number of nodes and con-

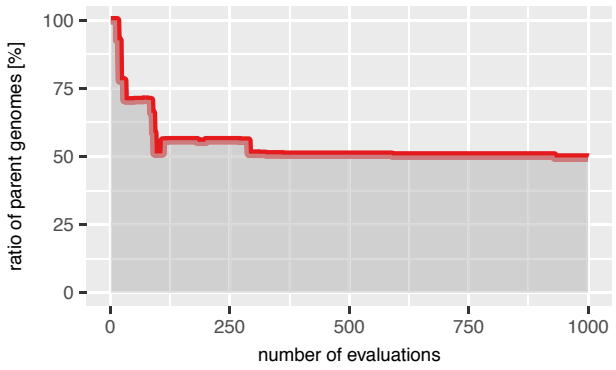

Figure 6: Percentage of parental genetic material in the best-performing CPPNs as lifetime learning progresses for the first depicted robot in the second generation.

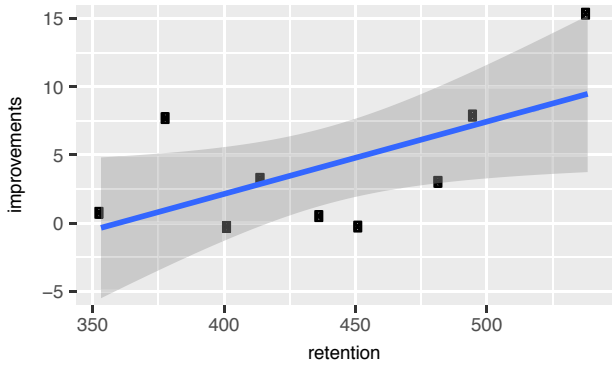

Figure 7: Correlation between amount of retained genetic material and the improvement afforded by the Lamarckian set-up. The blue line shows a linear model with the confidence interval in grey.

nections. One example of such a development of the remaining amount of genetic material is shown in Fig. 6.

We calculate the area under the curve for all of the 9 offspring, such that a higher number means that a higher amount of genetic material has been retained during the lifetime. Similarly, we calculate the area between the Lamarckian learning curve (*5+5*) and the non-Lamarckian curve (*initial*) from the trials in Fig. 5, such that a higher number means a larger improvement over the non-Lamarckian approach. We then plot these value pairs for each of the trials in Fig. 7, together with a fitted linear model (R^2=0.363, p=0.086).

Discussion

From Fig. 4 and Tab. 1 we can confirm that $(\mu + 1)$ ES working on a direct coding is an efficient learning approach, which is in line with earlier studies (Jelisavcic et al., 2016). However, while $(\mu + 1)$ ES seems to be the overall most efficient method for the tested scenarios, the CPPN-based encoding delivers competitive performance, especially when applied to the CPG controllers. We can thus conclude that CPPN-based lifetime learning is an approach meriting further investigation, especially since this enables a Lamarckian evolution paradigm.

Fig. 5 shows that either the Lamarckian approach achieves slightly better performance after 1 000 evaluations, or that there are only small differences between the performance of three approaches. This may indicate that, given enough time, any of the approaches could reach the practical limits for a specific morphology. However, the Lamarckian approaches tend to achieve good performances earlier in the evolutionary runs. Scenarios, where offspring start their learning phase with CPPNs inherited from their parents show high performance from the outset compared to those where the CPPNs are re-initialised and also compared to the control case where non-parent CPPNs were inherited. This would be a major advantage in a set-up where learning is performed on real robots. Faster convergence would allow for an earlier termination of the learning process, reducing the physical wear and tear on the robots and accelerating the overarching evolutionary process. We also observe that there are some morphologies where all learning methods struggle to achieve good performances, which seems related to the size of the morphologies.

From the example in Fig. 6, we observe that in our Lamarckian set-up there seems to be a rapid replacement of old genetic material with new in the initial 100 to 200 evaluations, before stabilising on around 50% ancestral genetic material. Apparently, this ancestral material plays a crucial role in the behaviour of the controllers, and Fig. 7 suggests there is indeed a correlation between the improvement over the re-initialized controllers and the *amount* of genetic material retained from parental controllers. There may be more complex reasons underlying this observation, which should be investigated by further experiments and analysis of the genetic material.

While the employed CPPN-CPG approach may seem counterintuitive for achieving lifetime learning –due to it traditionally being used for evolutionary timescales and being associated with the development phase of a robotic morphology– we argue that it makes sense here as it allows for Lamarckian evolution and also shows promising performance compared to a typical directly encoded learning algorithm. In these experiments, the morphology and the controller are encoded separately, which helps separate the evolution and learning components, one could also picture this approach being possible for indirect encodings that combine morphology and control genes, such as in Hornby et al. (2003). It would then be a matter of "freezing" the morphology genes while allowing for variations of the controller genes on a lifetime learning timescale.

Conclusion

In this paper we have studied a system where both the morphologies and controllers of modular robots can evolve or be learned. We have shown that evolving CPPNs as an indirect encoding for CPG controllers is a viable scheme for lifetime learning and that this approach effectively enables a Lamarckian paradigm for the non-trivial case of evolving morphologies. We have analyzed the approach of transferring learned knowledge to offspring, showing that the faster convergence towards good locomotion performance is related to the genetic material inherited from parents and that this improvement is correlated with the amount of retained genetic material.

The results from this paper should be seen as the first step into more research on lifetime learning-enabled morphology-control evolution scenarios. For future work, it would be beneficial to study evolutionary timescale effects (e.g., of different selection schemes) on lifetime learning, and to investigate both Lamarckian and Baldwinian approaches for different scenarios.

References

Ackley, D. H. and Littman, M. L. (1994). A Case for Lamarckian Evolution. *Artificial Life III*, pages 3–10.

Auerbach, J., Aydin, D., Maesani, A., Kornatowski, P., Cieslewski, T., Heitz, G., Fernando, P., Loshchilov, I., Daler, L., and Floreano, D. (2014). Robogen: robot generation through artificial evolution. In *Proceedings of the Artificial Life Conference (ALIFE XIV)*, pages 136–137. MIT Press.

Bongard, J., Zykov, V., and Lipson, H. (2006). Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121.

Cheney, N., Bongard, J., Sunspiral, V., and Lipson, H. (2016). On the Difficulty of Co-Optimizing Morphology and Control in Evolved Virtual Creatures. In *Proceedings of the Artificial Life Conference (ALIFE XV)*, pages 226–234. MIT Press.

Cheney, N., MacCurdy, R., Clune, J., and Lipson, H. (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*, GECCO '13, pages 167–174. ACM.

Clune, J., Stanley, K. O., Pennock, R. T., and Ofria, C. (2011). On the performance of indirect encoding across the continuum of regularity. *IEEE Transactions on Evolutionary Computation*, 15(3):346–367.

Dias, B. G. and Ressler, K. J. (2014). Parental olfactory experience influences behavior and neural structure in subsequent generations. *Nature neuroscience*, 17(1):89–96.

Eiben, A. and Smith, J. (2015). From evolutionary computation to the evolution of things. *Nature*, 521(7553):476–482.

Haasdijk, E., Rusu, A. A., and Eiben, A. E. (2010). HyperNEAT for locomotion control in modular robots. In *International Conference on Evolvable Systems*, pages 169–180. Springer.

Haasdijk, E., Smit, S. K., and Eiben, A. E. (2012). Exploratory analysis of an on-line evolutionary algorithm in simulated robots. *Evolutionary Intelligence*, 5(4):213–230.

Hornby, G. S., Lipson, H., and Pollack, J. B. (2003). Generative representations for the automated design of modular physical robots. *IEEE Transactions on Robotics and Automation*, 19(4):703–719.

Jelisavcic, M., Carlo, M. D., Haasdijk, E., and Eiben, A. E. (2016). Improving RL Power for On-Line Evolution of Gaits in Modular Robots. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE.

Jelisavcic, M., Kiesel, R., Glette, K., Haasdijk, E., and Eiben, A. E. (2017). Benefits of lamarckian evolution for morphologically evolving robots. In *Proceedings of the 19th annual conference on Genetic and evolutionary computation*. ACM. In press.

Kober, J. and Peters, J. (2009). Learning motor primitives for robotics. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 2112–2118. IEEE.

Lehman, J. and Stanley, K. O. (2011). Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 211–218. ACM.

Lipson, H. and Pollack, J. (2000). Automatic design and manufacture of robotic lifeforms. *Nature*, 406(August):974–978.

Liu, W. and Winfield, A. (2011). Autonomous morphogenesis in self-assembling robots using IR-based sensing and local communications. *Swarm Intelligence*, pages 107–118.

Pfeifer, R. and Iida, F. (2004). Embodied Artificial Intelligence: Trends and Challenges. In Iida, F., Pfeifer, R., Steels, L., and Kuniyoshi, Y., editors, *Embodied Artificial Intelligence*, pages 1–26. Springer Berlin Heidelberg.

Shen, H., Yosinski, J., Kormushev, P., Caldwell, D. G., and Lipson, H. (2012). Learning Fast Quadruped Robot Gaits with the RL PoWER Spline Parameterization. *Cybernetics and Information Technologies*, 12(3):66–75.

Sims, K. (1994). Evolving virtual creatures. In *Proceedings of the 21st annual conference on computer graphics and interactive techniques*, SIGGRAPH '94, pages 15–22. ACM.

Sproewitz, A., Moeckel, R., Maye, J., and Ijspeert, A. J. (2008). Learning to move in modular robots using central pattern generators and online optimization. *The International Journal of Robotics Research*, 27(3-4):423–443.

Stanley, K. O., D'Ambrosio, D. B., and Gauci, J. (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212.

Stanley, K. O. and Miikkulainen, R. (2002). Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation*, 10(2):99–127.

Weel, B., D'Angelo, M., Haasdijk, E., and Eiben, A. (2017). Online gait learning for modular robots with arbitrary shapes and sizes. *Artificial Life Journal*, 23(1):80–104.