Article

Adaptive Behavior

Adaptive Behavior
2018, Vol. 26(2) 47–64

DOI: 10.1177/1059712318762807
journals.sagepub.com/home/adb
SAGE

Evolving morphology automatically reformulates the problem of designing modular control

Anton Bernatskiy and Josh Bongard

Abstract

Modularity is a system property of many natural and artificial adaptive systems. Evolutionary algorithms designed to produce modular solutions have increased convergence rates and improved generalization ability; however, their performance can be impacted if the task is inherently nonmodular. Previously, we have shown that some design variables can influence whether the task on the remaining variables is inherently modular. We investigate the possibility of exploiting that dependence to simplify optimization and arrive at a general design pattern that we use to show that evolutionary search can seek such modularity-inducing design variable values, thus easing subsequent search for highly fit, modular organization within the remaining design variables. We investigate this approach with embodied agents in which evolutionary discovery of morphology enables subsequent discovery of highly fit, modular controllers and show that it benefits from biasing search toward modular controllers and setting the mutation rate for control policies higher than that for morphology. This work also reinforces our previous finding that the relationship between modularity and evolvability that is well-studied in nonembodied systems can, under certain conditions, be generalized to include embodied systems as well and provides a practical approach to satisfying the conditions in question.

Keywords

Reformulation, modularity, genetic algorithm, evolvable morphology, evolutionary robotics, morphological computation

Handling Editor: Tom Froese, National Autonomous University of Mexico (UNAM), Mexico

1. Introduction

To behave adaptively is to respond to changes in the environment in such a way as to achieve a certain goal. A system capable of such a behavior can be seen as an automatic problem solver for a certain range of tasks. A consequence of that is the possibility for natural adaptation mechanisms such as evolution and learning be modeled with problem solvers such as evolutionary or machine learning algorithms (Matarić, 1998; Orzack, Orzack, & Sober, 2001). The opposite is also true: When designing systems capable of adaptive behavior, general approaches to problem-solving are valuable tools that can be used either directly as a component of the solution (Bongard, Zykov, & Lipson, 2006) or to design a solution for a particular environment automatically (Floreano & Mondada, 1994).

One idea often instantiated in such approaches is to divide the whole problem into subproblems and solve each one of these with some degree of independence from others. This idea, which we will call the *division approach*[1] has proven to be effective for a wide variety of tasks. In politics and warfare, the principle *divide et impera* is used at least since antiquity (Strabo, AD23). The approach plays a fundamental role in engineering (Baldwin & Clark, 2006; Suh, 1990), including algorithmic design (Kleinberg & Tardos, 2005) and artificial adaptive systems (Ashby, 1960). Whenever the division approach is used, the resulting solution possesses a system property called modularity. It is broadly defined as a capacity of a system consisting of many components to be decomposed into groups of components (modules) such that within each group the dependencies

The University of Vermont, Burlington, VT, USA

Corresponding author:
Anton Bernatskiy, The University of Vermont, 210 Colchester Avenue, Burlington, VT 05405, USA.
Email: abernats@uvm.edu

between the components are strong, while the dependencies between the components that are in different groups are comparatively weak. Although the definitions of "components" and "dependencies" vary depending on the nature of the system, all systems that are produced with the division approach are modular at least in one way. In particular, components of the resulting system that are produced by solving a single subproblem all arise from the same solution process and in this sense are more dependent on each other than the components pertaining to different subproblems.

A peculiar property of modularity is that if the process of solving the task is incremental and the modularity is present in a partial solution, then even the simplest search methods can provide the same benefits as the division approach. Consider the problem of designing an internal combustion engine with a maximum total energy output. If the optimization process begins with an engine in which all subsystems (cooling, lubrication, etc.) are tightly coupled, then any chance improvement of one subsystem will likely cause many other subsystems to change their behavior, increasing the probability that the overall change will be detrimental to the performance. On the other hand, if the subsystems are functioning with relative independence from each other, then a possible improvement within one subsystem will not affect other subsystems much, resulting in an overall increase in performance (Suh, 1990). In incremental setting, this is how the division approach operates: After dividing the problem into subproblems, each of these is solved, to some extent, independently. This pattern can be enforced by the modular structure of the partial solution, or it can be built into the solution process by the designer, resulting in modular partial solutions. Thus, in engineering, whenever the incremental algorithms are used, the causal relationship between the solution modularity and the division approach is bidirectional.

In nature, modularity is observed in all biological systems from the molecular to the ecosystem level (Carroll, 2001; Girvan & Newman, 2002; Wagner, Pavlicev, & Cheverud, 2007). Its emergence in biological systems is hypothesized to be related to the reduction in cost of complexity that is associated with it (Welch & Waxman, 2003). Several mechanisms by which such emergence may occur have been proposed (Clune, Mouret, & Lipson, 2013; Draghi & Wagner, 2008; Espinosa-Soto & Wagner, 2010; Kashtan & Alon, 2005; Lipson, Pollack, Suh, & Wainwright, 2002; Solé & Fernández, 2003; Wagner et al., 2007). Many of these were investigated by evolving model problem solvers, such as genetic regulatory networks or neural networks, with genetic algorithms, and in almost all these experiments an increased rate of adaptation (i.e. rate of error reduction or fitness increase) was reported to coincide with the emergence of modular solution candidates (Clune et al., 2013; Espinosa-Soto & Wagner, 2010; Kashtan & Alon, 2005; Lipson et al., 2002). Similar improvement was observed even when modularity was introduced into the systems artificially by biasing search toward more modular solutions (Bernatskiy & Bongard, 2015; Durr, Floreano, & Mattiussi, 2010).

This coincidence appears analogous to the relationship between modularity and the division approach in engineered systems. In many models, it was indeed observed that evolution produced modular solutions consisting of quasi-independent modules solving subtasks and/or capable of evolving separately (Cappelle, Bernatskiy, Livingston, Livingston, & Bongard, 2016; Clune et al., 2013; Ellefsen, Mouret, & Clune, 2015; Espinosa-Soto & Wagner, 2010; Kashtan & Alon, 2005). This is a non-trivial observation because it has also been discovered that more modular networks, on average, tend to have smaller number of connections than their less modular counterparts (Bernatskiy & Bongard, 2015; Clune et al., 2013; Lipson et al., 2002). Thus, the increase in rate of adaptation could be attributed to the decrease in the effective size of the search space, as opposed to the independent optimization of modules.

There is evidence, however, that the viability of modular solutions, together with the benefits they bring, is heavily dependent on the task. In our previous work, we provide an example of a task and a robot morphology for which any controller that solves the task must be nonmodular, and we show that the rate of evolutionary adaptation for this task–morphology pair is much slower than the rate of evolutionary adaptation for the same task, but with a morphology that enables modular control (Bernatskiy & Bongard, 2017). Indirectly, the dependence of the feasibility of modular control on the morphology (and therefore on the structure of the control task) is hinted on by the results showing that modular controllers are more likely to evolve if the morphology evolves alongside the control, with the additional objective of behavioral conservatism (Bongard et al., 2015).

2. Reformulation

In most of the work cited above, the task itself is considered to be fixed. In engineering, however, this is rarely the case. For almost every real-world task, there is a space of possible ways to approach and formalize it. Changing the approach can trivialize tasks that initially appear intractable. Human problem solvers are well-known to be able to exploit that dependence. We will call the corresponding cognitive technique—reformulating the problem to make it easier to solve, instead of attacking it directly—*reformulation* (e.g. Choueiry, Iwasaki, & McIlraith, 2005). In human cognition, reformulation is arguably the instrument of choice when dealing with the hardest tasks. It is widely

hypothesized that it underlies many aspects of cognitive insight (Duncker & Lees, 1945; Sternberg & Davidson, 1995; Wicker, Weinstein, Yelich, & Brooks, 1978), including the so-called Eureka effect (Knoblich, Ohlsson, & Raney, 2001; Sternberg & Davidson, 1995).

We formally define reformulation as follows (Figure 1(a)). Consider a finite-dimensional optimization problem, for which solutions are encoded as vectors of N values that minimize some real-valued function. Suppose there is some measure of difficulty for finding such a solution. Examples of such measures include a binary value that indicates whether a certain solution technique worked, or the number of operations required to achieve an acceptable result. Suppose further that changing some subset of M ($M<N$) variables (which we will hereafter call *driving variables*) can significantly change the difficulty of optimizing the $N-M$ remaining *non-driving variables*. The approach is then to isolate the driving variables and optimize their values (i.e. the *formulation* of the problem of optimizing the non-driving variables) to ease optimization of the non-driving variables.

To instantiate the reformulation approach, a designer must isolate the driving variables, select the two optimizers—for driving and non-driving variables—and provide an appropriate quantitative definition of "optimization difficulty" of the non-driving variables' optimization given some values of driving variables. The driving variables must influence the difficulty of optimization of the non-driving variables *with the optimizer that has been selected for these*. Selecting such driving variables is generally a matter of domain knowledge, but some guidelines can be provided (see section 2.2). The definition of difficulty must be *correct* in a sense that it must not assign low difficulty to the driving variables' values that prohibit the discovery of good-enough solutions in optimization of non-driving variables. If this condition is not satisfied, a possibility of premature convergence arises. In addition, the definition must be significantly less computationally expensive than finding a good-enough solution for most values of driving variables, because otherwise it is less computationally expensive to solve the optimization problem without reformulating it.

If both driving and non-driving variables are optimized with incremental algorithms, one natural way of estimating the difficulty is to set values of the driving variables, run the algorithm for the non-driving variables for some number of iterations, and estimate the difficulty based on how much did the fitness improve. We will refer to this difficulty definition as *improvement-based*. The downside of this approach is that its correctness depends on the properties of the fitness landscape: If for some values of driving variables the fitness improves rapidly, yet its ultimate value is not good enough, then premature convergence is possible.

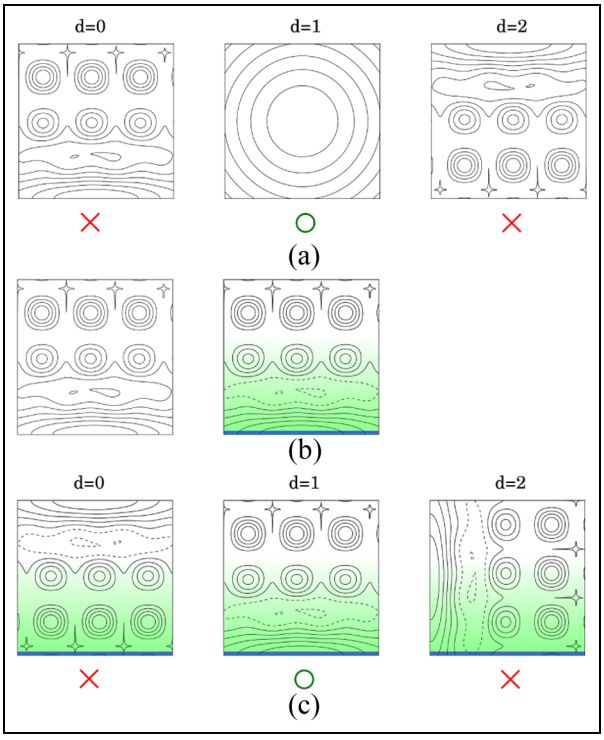

Figure 1. Graphical representations of the core approaches of the article. (a) *Reformulation*: Shape of the goal function landscape depends on some variables (exemplified by d). If for some values of these variables optimization on the landscape can be done more easily, we call them *driving variables*. To reformulate the optimization task is to optimize those variables to find the values that simplify the underlying optimization process. In our example, such a value is $d = 1$, corresponding to convex optimization. (b) *Bias*: The search is biased (as indicated by the green gradient) toward a subset of search space (indicated by the blue stripe in the bottom of the square). The technique introduces some assumptions about the fitness, in our example that the fitness landscape is more convex and contains good-enough solutions near the blue stripe. If these assumptions are not satisfied, the bias may be detrimental. (c) *Guided reformulation*: Shape of the goal function landscape depends on some variables (again exemplified by d). However, the difficulty of optimization does not depend on those variables unless the bias is applied, in which case the variable affects whether the assumptions of the bias are satisfied and thus the optimization difficulty. If the bias is applied, these variables become driving and can be optimized to reformulate the original optimization problem while using the bias. In our example, the optimal value is again $d = 1$: for that value, the fitness landscape is convex and has good-enough solutions around the area toward which the search is biased.

The number of iterations that are used to estimate the optimization difficulty for every value of the driving variables governs the reliability of this difficulty definition: If this number is unlimited, the definition is correct; if it is small, it is not correct on many fitness landscapes. We will refer to such definitions of difficulty as *approximately correct*. The conflict between the

need to spend the iterations on refining the estimates made with this definition and on optimizing the driving variables results in two timescales: a slow one for the driving variables and a fast timescale for the non-driving variables. This is an instance of separation of timescales of adaptation (Pfeifer & Bongard, 2006): a widespread property of natural and artificial systems originally discovered in evolutionary biology (Huxley, 1942; Martin, Bateson, & Bateson, 1993; Tinbergen, 1963) and later in neuroscience (Fairhall, Lewen, Bialek, & Van Steveninck, 2001; Ulanovsky, Las, Farkas, & Nelken, 2004; Wark, Fairhall, & Rieke, 2009) and machine learning (Hawkins & Blakeslee, 2007; Kiebel, Von Kriegstein, Daunizeau, & Friston, 2009; Yamashita & Tani, 2008).

2.1. Evolution and reformulation

Evolutionary computation is an incremental optimization technique that relies on trial and error. It is possible to use evolution to optimize both driving and non-driving variables (e.g. Cheney, Bongard, SunSpiral, & Lipson, 2017), or to only evolve the driving variables, while the non-driving ones are obtained via other techniques such as gradient descent (e.g. Hinton & Nowlan, 1987; Miikkulainen et al., 2017). Evolutionary optimization can be improved with reformulation and related ideas in many ways, and several major ideas of the field are, in the opinion of authors, related to the reformulation approach.

Canalization (Hinton & Nowlan, 1987; Waddington, 1942) can be described as a phenomenon of discovering a set of values of driving variables that contain much of the information about optimal non-driving variable values discovered with the corresponding optimizer. It occurs when the design variables are divided into two subsets, one in which the variable values are learned during the lifetime and the other in which they are learned by evolution. The variables under the evolutionary control affect the difficulty of lifetime learning and thus can be thought of as driving variables, with each set of values being a "learner." Learner's fitness is defined as a nondecreasing function of the proportion of the learner's lifetime that it spends with all the variables having exactly the right values; such fitness is a correctly defined difficulty of optimizing the non-driving variables. Lifetime learning of the non-driving variables enables the evolution to evaluate a mutation-continuous subset of the variable values rather than a single point, effectively smoothing the fitness landscape. As long as the values are correct, the more the information on the non-driving variable values is contained within the driving variable values, the easier the lifetime learning is; hence, there is an evolutionary pressure to store all the information on the non-driving variable values within the driving variables. Thus, the characteristics learned over lifetime (non-driving variable values) are assimilated into the genome (driving variable values) over the evolutionary time, ultimately producing a formulation that contains all the information on the solution.

A development of the idea of canalization is evolution for evolvability (Kounios et al., 2016; Watson & Szathmáry, 2016), where variables describing genotype-to-phenotype map can bias the evolution of underlying variables toward local optima and improve the rate of convergence when the population is near them. Such genotype-to-phenotype map can be thought of as a vector of driving variables. Evolving it in a slower timescale compared to the rest of variables and interleaving the periods of neutral and adaptive evolution forces the evolution to generalize over multiple ways of reformulating the underlying problem.

Morphological protection (Cheney et al., 2017) can be seen as a method for circumventing the incorrectness of the improvement-based difficulty definition. It investigates the evolution of morphology alongside the controller in robotics control tasks. Morphology governs the complexity of control and thus is a driving variable. Protecting values of this variable (i.e. formulations of the control problem) from modification for the period while the control is optimized enables these values to compete based on the final fitness of controllers that can be evolved for them and not the fitness improvement over any fixed number of generations.

In co-evolution (Axelrod, 1987; Nolfi & Floreano, 1998; Rosin & Belew, 1997), two groups of variables are optimized to make each other's optimization more difficult. Thus, co-evolution can be thought of as an approach similar to reformulation, but with positive influence of one optimization process on another's rate of convergence replaced with a negative feedback. While evolutionary reformulation is good at producing simple solutions, co-evolution tends to produce the solutions that complexify over evolutionary time, which can be valuable in many applications.

Aside from fairly general approaches, some evolutionary algorithms for more narrow classes of tasks can also be seen as instances of the reformulation approach. One example is Genetic Programming with Embedded Spatial Aggregation (GPESA), a method that evolves a pattern of aggregation of geospatial variables alongside a genetic programming model for value prediction (Kriegman, Szubert, Bongard, & Skalka, 2016). Another is the evolution of deep learning networks (Miikkulainen et al., 2017), in which the topology of the network is evolved to maximize its learning rate.

A similar but different procedure of reformulation is optimization of hyperparameters in machine learning (e.g. Bishop, 2006). The difference is in the designer's intention: In machine learning hyperparameter tuning, optimization is typically done to improve the final model, while reformulation aims at simplifying the *process* of constructing the model, to find good-enough solutions to the toughest tasks.

2.2. Guided reformulation and its application to evolutionary robotics

One difficulty in using the reformulation approach is the need to select the driving variables. In human cognition, the method is typically "thinking outside the box": finding some controllable variables that influence the difficulty of the task, but of which the designer is initially unaware. However, whether a given variable does or does not influence the difficulty of the task depends on the method that is used to optimize the non-driving variables. This opens up a possibility for either making certain variables into driving ones or increasing the influence that some known driving variables exert on the solution difficulty. Here we will explore an approach that uses that possibility.

One powerful approach to improving the performance of the optimization is to bias the search (Figure 1(b)). The bias prioritizes the solution candidates from a subset of the search space in which, based on the domain knowledge, some reasonably good solutions can be expected, or which has a reasonably good probability of intersecting a hill climbing path toward such solutions. For example, if an embodied agent such as human or animal searches the area for an object that can only be spotted from a short distance, a common procedure is to follow a linear trajectory that covers the area in such a way that the distance from every point of the area to the trajectory is reasonably small. The trajectory is also typically organized in such a way that the areas where, according to prior knowledge, the object can be found with higher probability are searched earlier. In machine learning, the same idea can be implemented with indirect encodings (Stanley & Miikkulainen, 2003) or by co-optimizing some heuristic parameter together with goal function. Some examples of such parameters relevant to neural network optimization are connection cost (Clune et al., 2013) and L2 norm of the weights (Bishop, 2006; both these should be minimized for a useful bias).

This approach effectively reduces the size of the search space and minimizes, based on the prior knowledge, the average time to arrive to the solution. It can also be used to select a solution with useful properties if multiple good-enough solutions exist. However, it does so at the cost of introducing some assumptions about the solution and the search space. If the assumptions do not hold, the bias can be detrimental. For example, in our previous work (Bernatskiy & Bongard, 2017), we have shown that for certain control tasks strengthening the bias toward sparsity can make the evolutionary optimization much more difficult.

Suppose that for some bias, some subset $\mathcal{A}$ of design variables can influence whether the assumptions introduced by the bias hold or not, with respect to the optimization of the remaining variables. If the assumptions introduced by the bias hold, the usage of the bias is likely to have a drastic impact on the difficulty of the problem of optimizing the variables not in $\mathcal{A}$. Thus, as long as the bias is used in optimization of the variables not in $\mathcal{A}$, the variables in $\mathcal{A}$ are *driving variables*. They then can be used to reformulate the problem in such a way that the assumptions introduced by the bias do hold and the usage of the bias improves the performance of optimization.

We will call the resulting approach *guided reformulation* (Figure 1(c)). It can be summarized as follows:

To reformulate an optimization problem, find a pair of an optimization bias and a set of variables such that the assumptions introduced into optimization by the bias are dependent on the variables in the set. The reformulation procedure that uses the variables in the set as driving and uses the bias in optimization of the non-driving variables is then likely to benefit from the bias (typically by producing good-enough solutions selected for additional properties by the bias more rapidly than in the absence of bias or reformulation procedure).

Note that this approach does not aim to eliminate the need for domain knowledge, but to provide some guidance on how to use it instead. Fully automatic instantiation of the reformulation approach is outside the scope of this article.

One particularly powerful type of bias that can be used in guided reformulation is the bias toward modularity (see section 1). Resistance to catastrophic forgetting and (in network optimization) correlation with sparsity can make evolutionary algorithms biased toward modularity converge much more rapidly than their counterparts with no such bias (Bernatskiy & Bongard, 2015; Clune et al., 2013). Here we describe a way to use guided reformulation to get the benefits of the bias toward modularity in a robotic control task, building on the body of research outlined in section 1.

We consider a task of controlling a robotic agent embedded in physical space. From our previous work (Bernatskiy & Bongard, 2017), we know that the capacity to admit modular control can depend on an agent's morphology. The results from Bernatskiy and Bongard (2017) also suggest that such capacity is sufficient for the bias toward modularity to make the convergence much more rapid than in the absence of the bias. We will refer to this statement as **Hypothesis 0**. If this hypothesis is correct, then whenever the morphological variables influence the capacity of the agent to admit modular control, they also qualify as driving variables under the bias of control optimization toward modularity.

Due to the generality of Hypothesis 0, it is impossible to confirm experimentally. Instead of attempting to do that, we adopt falsificationist attitude toward it. To this end, we formulate some of its consequences and attempt to show that they are false.

We instantiate guided reformulation by optimizing morphology alongside the controller, the parameters of which are the non-driving variables in this case. We adopt a genetic algorithm as the optimization technique

for both sets of variables and separate the optimization of driving and non-driving variables using different mutation rates for morphology and control. If the mutation rate for control is greater than for morphology, many controllers are considered for every morphology, ensuring that the current values of error of evolutionary individuals (composed of a morphology and a controller) provide approximate definitions of difficulty of optimization of control. The bias toward modularity in control optimization is implemented in two ways: using the connection cost technique (Clune et al., 2013) and with initial populations of sparse networks (Bernatskiy & Bongard, 2015). It can also be switched off.

The approach is applied to the "Arrowbot" task and the environment from Bernatskiy and Bongard (2017).

In this setup, two premises additional to Hypothesis 0 are important:

Premise 1: It is known that for the "Arrowbot" task and environment, the capacity of modular control depends on certain morphological variables. This is established in Bernatskiy and Bongard (2017).
Premise 2: Evolutionary algorithm that we use is capable of following error gradients and converge on error minima (i.e. it is a functioning search algorithm). This is ensured by elitism of the algorithm: At every change in generation, the best performing individual is preserved.

We predict the following:

Hypothesis 1: Within each run, rapid convergence of the control evolution will follow the discovery of a morphology that admits modular control. It follows from Hypothesis 0 and Premise 1.
Hypothesis 2: Evolution of morphology will converge on body plans that admit modular control. Premises ensure that there is a gradient of control optimization difficulty for the morphological evolution to follow and that the minima of such difficulty will coincide with body plans that admit modular control. It follows from Hypothesis 0, Premise 1, and Premise 2.
Hypothesis 3: Evolving the control with a randomly selected fixed morphology will result in a less rapid convergence than evolving the morphology alongside the control. Equivalently, the convergence rate will be lower if the morphological evolution cannot follow its gradient. It follows from Hypothesis 0, Premise 1, and Premise 2.
Hypothesis 4: Without the bias toward modularity, convergence to body plans that admit modular control will likely not happen or will happen in a larger number of generations than in the case when the bias is enabled. From Hypothesis 0 and Premise 1, we conclude that the impact of morphology on the rate of convergence of control evolution is dependent on the presence of the bias. With the bias switched off, morphology's impact on the rate of convergence of control evolution will be decreased, causing the gradient guiding the morphological evolution to disappear or weaken.
Hypothesis 5: By the same reasoning, reducing or disabling the bias toward modularity will cause a decrease in the adaptation rate.
Hypothesis 6: As long as the bias toward modularity is switched on, the qualitative behavior of the system should not depend on the details of implementation of the bias. Equivalently, the status of morphology as a vector driving variable should depend on the presence of the modularity bias, but not on how it is achieved. This follows from Hypothesis 0 and Premise 1.
Hypothesis 7: There should be a ratio between the mutation rates of morphology and control that maximizes the convergence rate and is not 0 or 1, but is biased such that more control mutations than morphological mutations occur. This follows from the tension between the need for the difficulty definition to be as correct as possible and the need to advance the optimization of morphology, all within the same iterated optimization process.

The expected behavior of the system integrated from all the hypotheses above is shown in Figure 2.

We show Hypotheses 1–5 and 7 to be correct as long as Hypothesis 0 is clarified as follows: Discovery of a morphology that admits modular control is a necessary but not sufficient condition for subsequent rapid

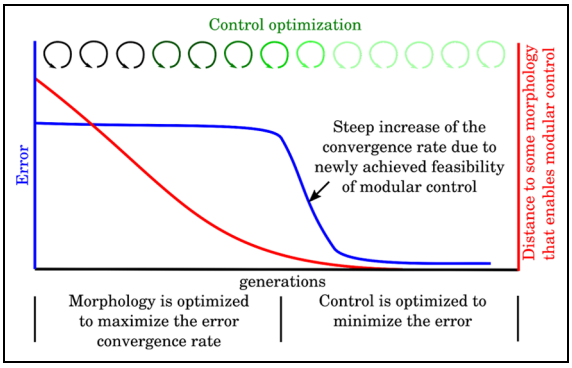

Figure 2. Expected behavior of evolutionary guided reformulation approach that uses morphology as vector driving variable and modularity as optimization bias for control (the non-driving variables). To exploit the speedup of the evolution that we hypothesize to arise for morphologies that make modular control feasible, we evolve the morphology alongside the control, but at a slower timescale; in addition, we bias the evolution of control toward modularity. Morphologies that admit modular control enable more rapid reduction in error and thus get an evolutionary advantage. The fast optimization timescale is shown with circular arrows at the top of the graph, with color intensity of the arrow signifying the difficulty of control optimization at that point of evolutionary history. If the task permits controllers consisting of a multitude of modules for some morphology, the error reduction rate will radically increase as this morphology is approached. Note that what the morphology influences is the rate of convergence, not the error itself.

convergence of controller evolution. We fail to obtain any results that contradict Hypothesis 0 clarified in that way and Hypothesis 6. We investigate how the results behave as the task is scaled up.

3. Methods

In this section, we describe in detail the task, environment, and the family of robotic morphologies that we used to test our hypotheses and also the genetic algorithm we used.

3.1. Robot and Task

We here study the interaction between morphological evolution and modularity-biased controller evolution using Arrowbots, a robotic substrate we previously used to demonstrate the influence of morphology on the feasibility of modular control and its evolution (Bernatskiy & Bongard, 2017).

An Arrowbot is a robot which consists of N segments with pointers stacked to form a column (Figure 3). The adjacent segments are connected with motors that share a common geometric axis of rotation, that is, the vertical axis; the bottommost segment is connected to a fixed base. Any motor input controls the angular velocity of the segment that it connects to, relative to the segment below it, or, in the case of the motor connected to the first segment, the fixed base. We will denote those relative orientations as r_i,[2] the angular velocities as $\dot{r}_i$, and the corresponding motor inputs as m_i.

Absolute orientations of the segments will be measured with respect to the fixed base and denoted as A_i. The relationship between the two is as follows

$$\begin{aligned} A_i &= \sum_{k=1}^{i} r_k \\ r_i &= \begin{cases} A_1 & \text{for} \quad i = 1 \\ A_i - A_{i-1} & \text{for} \quad i = 2, \ldots, N \end{cases} \end{aligned} \tag{1}$$

For convenience, later we will also use the vector notation

$$\begin{aligned} \boldsymbol{A} &= K\boldsymbol{r} \\ \boldsymbol{r} &= K^{-1}\boldsymbol{A} \\ K &\equiv \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 1 & 1 & \ldots & 0 \\ & \cdot & \cdot & \\ 1 & 1 & \ldots & 1 \end{bmatrix} \\ K^{-1} &= \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 & 0 \\ -1 & 1 & 0 & \ldots & 0 & 0 \\ 0 & -1 & 1 & \ldots & 0 & 0 \\ & & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \ldots & -1 & 1 \end{bmatrix} \end{aligned} \tag{2}$$

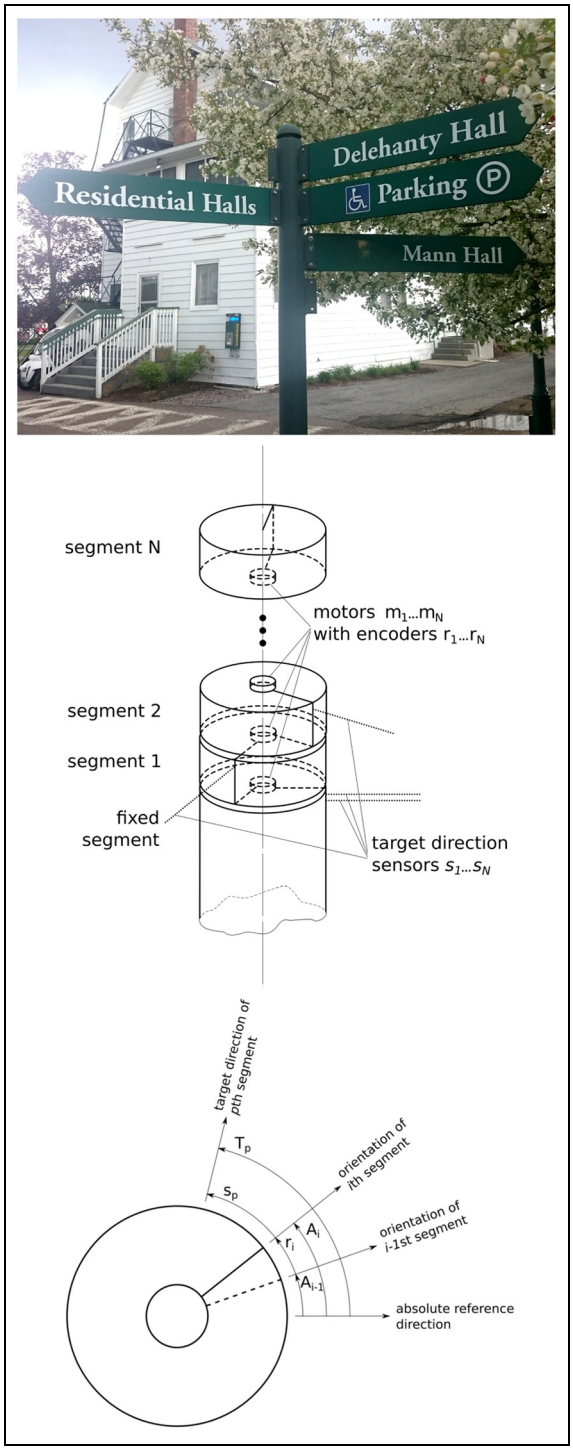

Figure 3. Multi-directional road sign (top), its dynamic version, the Arrowbot (middle), and the associated kinematic notation (bottom).

Arrowbots are equipped with sensors of two kinds: N proprioceptive sensors that directly measure the orientations of the segments they are attached to $\boldsymbol{r}$ and N target orientation sensors $\mathbf{s}$. Any target orientation sensor s_i outputs the difference between the orientation of the object it is attached to and the target orientation

T_i. Possible attachment points for any sensor include any segment and the fixed base. For example, sensor s_3 always measures the difference between the target direction of the third segment T_3 and whatever it is attached to. If it is attached to the second segment, it will output $T_3 - A_2$; if it is attached to the fixed base, it will output just T_3.

More generally

$$\mathbf{s} \equiv \boldsymbol{T} - J\boldsymbol{A} \tag{3}$$

where J is an $N \times N$ matrix such that $J_{ij} = 1$ if the jth target orientation sensor s_j is attached to the ith segment and 0 otherwise. Within a fixed environment, matrix J fully determines the effect that any motor action of an Arrowbot has on its sensors; therefore, it determines its morphology. Hereafter, we will use the terms "morphology", "target orientation sensor attachment pattern," and "matrix J" interchangeably.

A controller of an Arrowbot is the part that takes the sensor readings as inputs and feeds its outputs to the motors. It can be abstracted as a mapping $\mathbf{m} = \mathbf{m}(\mathbf{s}, \boldsymbol{r})$. Throughout this article, we will discuss the general case of nonlinear control and its properties; for the experimental part of the article, however, we will use linear control

$$\mathbf{m}(\mathbf{s}, \boldsymbol{r}) = W\mathbf{s} + Y\boldsymbol{r} \tag{4}$$

Furthermore, we limit the elements of matrices W and Y to be in the set $\{-1, 0, 1\}$ to reduce the size of the search space.

To define the connectivity of such controllers, we represent them as graphs. Each sensor or motor is represented as a node, and each nonzero coefficient in matrices W and Y is represented as a bidirectional connection between a sensor node and a motor node. With this definition, the analysis of the dependence (defined as in Bernatskiy & Bongard, 2017) is reduced to the analysis of connectivity of the graph it is represented with: Whenever there is a connected path between any two nodes, they are dependent. Thus, the connected components of the controller graph represent groups of variables in which every variable is dependent on every other one.

We will consider a controller to be modular if its graph has more than one connected component. This definition will be used for consistency with Bernatskiy and Bongard (2017), which is required for using some analytical results from that work. Compared to the more traditional Q metric (Newman, 2006), this definition provides a more coarse grained picture in which modular networks are guaranteed to have at least two groups of completely independent variables (see Figure 4). This, in turn, guarantees that the connections within each group can be changed without influencing any other group through the controller, ensuring that the pattern of adaptation characteristic of the division approach (see section 1) is induced within any trial-and-error optimization method with localized changes in connections.

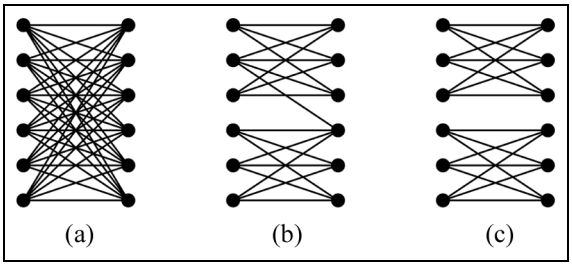

Figure 4. Types of network modularity in linear controllers. (a) Fully connected controller is nonmodular. (b) A controller with high Q, but with a single connected component. (c) A controller with two connected components. To use analytical results from Bernatskiy and Bongard (2017), we must require complete independence between modules within the controller for the controller to be called modular. Hence, only the networks of type (C) are considered modular within this article.

Knowing the morphology and the controller, it is possible to determine the dynamics of an Arrowbot

$$\dot{\boldsymbol{r}} = \mathbf{m}(\boldsymbol{r}, \mathbf{s}(\boldsymbol{r}, \boldsymbol{T})) \tag{5}$$

where $\mathbf{s}(\boldsymbol{r}, \boldsymbol{T})$ is given by equation (3).

The task of the robot is to orient its pointers in arbitrary target orientations $\boldsymbol{T}$ starting the movement at arbitrary initial conditions r_0. Ideally, the performance of the robot should be measured with all possible settings of initial conditions and target orientations; in addition, since we are only interested in the final pointing error and not the transient time, the evaluation times should be infinite. The performance in this case can be quantified as

$$E_{ideal} = \sup_{\{\boldsymbol{T}\},\{\boldsymbol{r}_0\}} \lim_{t\to\infty} (\boldsymbol{T} - \boldsymbol{A}(t)|_{\boldsymbol{r}_0,\boldsymbol{T}})^2 \tag{6}$$

where x^2 is a scalar product, $\mathrm{x}^2 \equiv \sum_i x_i^2$, and $\{\boldsymbol{T}\}$ and $\{\boldsymbol{r}_0\}$ are sets of all possible target orientation vectors and initial conditions' vectors, correspondingly.

In practice, we evaluated the performance of the robot by averaging its pointing error after a finite simulation time t over a finite set of initial conditions and target orientations

$$E = \frac{1}{n_{IC} n_{TO}} \sum_{\alpha=1}^{n_{IC}} \sum_{\beta=1}^{n_{TO}} (\boldsymbol{T}^\beta - \boldsymbol{A}(t)\big|_{\boldsymbol{r}_0^\alpha,\boldsymbol{T}^\beta})^2 \tag{7}$$

where n_{IC} and n_{TO} are the numbers of variants of initial conditions $\boldsymbol{r}_0^\alpha$ and target orientations $\boldsymbol{T}^\beta$ that the robot

is being tested at. In particular, throughout this article we used the following set of N target orientation vectors

$$\begin{aligned} \boldsymbol{T}^1 &= [1, 0, \ldots, 0],\ \boldsymbol{T}^2 = [0, 1, \ldots, 0],\ \ldots, \\ \boldsymbol{T}^N &= [0, 0, \ldots, 1] \end{aligned} \tag{8}$$

Each of these target orientation vectors was tested with just one set of initial conditions $\boldsymbol{r}_0 = [0, 0, \ldots, 0]$. Thus, $n_{IC} = n_{TO} = N$. $\boldsymbol{A}(t)$ is obtained by integrating equation (5) forward in time over the span of t with the fourth-order Runge–Kutta method. Fitness is computed using equation (7) using the final state of the system.

System (5) exhibits linear dynamics, resulting in a tendency for an exponential growth or decay of r over time. This results in heavy tailed distributions of the derived variables such as the approximate error E for Arrowbots with randomly generated morphologies and/or controllers. To simplify the statistical analysis of our data, we account for this tendency by running all statistical tests on the decimal logarithm of error, $\log_{10} E$.

3.2. Genetic encoding and mutation

In this article, we encoded each Arrowbot as a concatenation of two vectors of integers.

The first vector encodes the morphology of the Arrowbot in N integers from $\{0, 1, \ldots, N\}$. An integer at the ith position is the number of the segment to which the ith target orientation sensor s_i is attached, with the value of 0 representing attachment to the fixed base. It is thus a compressed encoding of sparse matrix J (equation (3)).

The second part encodes the linear controller (equation (4)) as $2N^2$ integers drawn from $\{-1, 0, 1\}$. The first N^2 elements of the genome are the elements of matrix W and the second half are the elements of Y, reshaped to form a linear array.

Random genomes for initial populations were generated by concatenating a vector of N integers randomly selected from $\{0, 1, \ldots, N\}$ (the morphology) with an encoding of a randomly generated controller. We compare two methods of generating the controllers randomly in this article. The default method for producing the controller is to generate a vector of $2N^2$ integers randomly selected from $\{-1, 0, 1\}$. This produces a linear controller in which about two-thirds of the coefficients are nonzero; we will refer to such controllers and to initial populations composed of them as *dense*. The alternative is to use the method introduced in Bernatskiy and Bongard (2015): Take a vector of $2N^2$ zeros and mutate it once using the control mutation operator described below. Due to the structure of the operator, such controllers will have exactly one nonzero coefficient drawn from $\{-1, 1\}$ and placed randomly. We will call such controllers and initial populations composed of them *sparse*.

We use a mutation operator that always changes the genome upon application. With a fixed probability P_{mm}, it changes the morphology; failing that, the controller is changed.

We considered two ways of performing a morphological mutation. By default, a randomly selected sensor was moved by one segment up or down, unless the sensor ended up below the fixed base or above the Nth segment as a result; in these cases, the operation was repeated starting with selecting the sensor randomly. The alternative was to discard the morphology altogether and replace it with a newly generated one. We will refer to this latter method as a random jump in the morphological space.

For control mutations, we used the same operator as in our previous work (Bernatskiy & Bongard, 2015, 2017). It treats the controller as a graph as described in sections 3.1. Each field of the weight vector represents a possible connection. Mutation can result in one of three outcomes: addition of a connection (probability 0.1), deletion (probability 0.1), and a density-preserving change in the network (probability 0.8). If a connection is to be added, a field with the value of 0 is found, a value from $\{-1, 1\}$ is randomly selected, and the field is set to the value. Deletion randomly selects a field with nonzero value and sets it to 0. Density-preserving mutation flips the sign of the value at a randomly selected nonzero field. If the operation is impossible (e.g. deletion is attempted on a network with no connections), the outcome selection is repeated until a feasible operation is selected.

3.3. Evolutionary algorithm

Our evolutionary algorithm is based on the simplified version (Bernatskiy & Bongard, 2015, 2017) of the biobjective performance and connection cost Pareto optimization technique introduced by Clune et al. (2013).

Here, the two objectives of the algorithm are minimization of the pointing error (equation (7)) and minimization of the number of nonzero-weight connections in the controller. The algorithm starts by generating an initial population consisting of a fixed number of genomes as described in the previous section. At each generation, a stochastic Pareto front is constructed as follows. Each genome is compared with every other genome in the population. Within each comparison, with probability P_{CC} the first genome is marked as dominated if the second genome has both lower error and less connections or, if genomes are equivalent with respect to these objectives, if the first genome was generated or mutated earlier than the second. With probability $1 - P_{CC}$, the first genome is marked as dominated if just its error is greater than the second genome's error. Genomes that are not marked as

dominated at the end of the procedure constitute the stochastic Pareto front.

For the values of constant P_{CC} not in $\{0, 1\}$, there is a non-negligible probability that every genome in the population will be marked as dominated, resulting in an empty stochastic Pareto front. Whenever that happens, a genome with the smallest error is added to the Pareto front.

All the genomes that are not on the stochastic Pareto front are removed from the population and replaced by offspring of genomes from the stochastic Pareto front. The population size is thus kept constant.

The parameter P_{CC} is added to control the relative emphasis evolution places on the two objectives (Clune et al., 2013). Here, we use it to investigate the role of connection cost in search for morphologies admitting modular control.

3.4. Prior knowledge of the morphospace

It has been shown (Bernatskiy & Bongard, 2017) that for the Arrowbots, there are two morphologies with radically different properties with respect to the fesibility of modular control (Figure 5).

If every target orientation sensor is placed on the segment for which it tracks the target, sensor placement matrix J is equal to $N \times N$ identity matrix I. In this case, every target orientation sensor measures the contribution of the segment it is attached to the total pointing error, $r_i + A_i$. We will refer to this morphology as $J = I$ (Figure 5(a)). Previously, we have shown that for this morphology there is a family of linear controllers with N disconnected modules with one connection (from s_i to m_i) each that reduces the ideal error (equation (6)) to 0.

It is also easy to see that this morphology admits controllers that have the maximum number of disconnected modules. Any graph in which the connections are only possible between the N motor nodes and sensor nodes with $N - 1$ or less connections will necessarily have a disconnected motor node, preventing the pointing error from converging to 0 unless the initial orientation of the corresponding segment is aligned with the target orientation. Thus, the task cannot be solved without at least N connections that can form up to N disconnected modules.

Another morphology corresponds to the case when all the target orientation sensors are attached to the fixed base. In this case, all elements of matrix J are zeros, $J = 0$, and the target orientation sensor measures the absolute angular positions of the targets (Figure 5(b)). For this morphology, any controller that reduces the ideal error to 0 must be fully connected, as shown in Bernatskiy and Bongard (2017).

In Bernatskiy and Bongard (2017), we found that evolution decreases the final pointing error much more rapidly for the $J = I$ morphology, compared to the morphology $J = 0$, and that the difference becomes more pronounced as the task is scaled up to more segments.

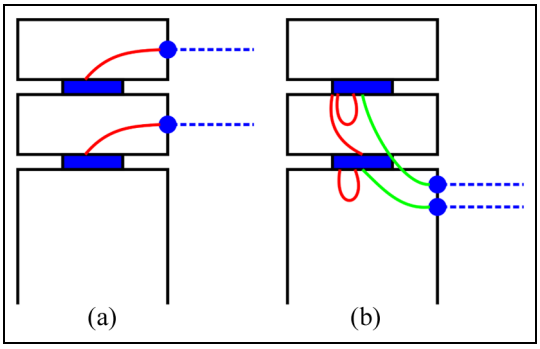

Figure 5. Some Arrowbot morphologies with certain known properties for $N = 2$. Blue circles with dotted lines represent target orientation sensors; blue rectangles represent motors with proprioceptive sensors within. The red and green lines show an example of an optimal (with respect to ideal error (6)) controller for each morphology: red for negative and green for positive feedback. (a) The $J = I$ morphology that can be controlled by a controller made of N disconnected modules. (b) For the $J = 0$ morphology, provably no controller that is optimal can have more than one connected component.

Here, we additionally provide a way to design a controller for any morphology that reduces the ideal pointing error (equation (6)) to 0 and provides a constant baseline performance that does not depend on the morphology for the practical error (equation (7)). See Appendix 2 for details.

4. Results

We began by comparing the performance of the evolution for Arrowbots comprising three segments with morphological mutation enabled ($P_{mm} = 0.2$) and without morphological mutations ($P_{mm} = 0$) with initial populations of dense controllers. For each of these settings, we performed 100 evolutionary runs and computed statistics on the decimal logarithm of final pointing error (Figure 6, left panel). We have found that for the case when morphological mutations were possible, evolution converged more rapidly and after 600 generations achieved a lower final error, consistent with Hypothesis 3.

Next, we analyzed morphologies that were evolved. We have found that in all the runs with morphological mutation, the $J = I$ morphology was discovered. We verified this using the new parameter μ—the minimal Hamming distance of the morphologies of the genomes on the Pareto front to the $J = I$ morphology (Figure 6, middle panel). As expected, in the absence of morphological mutation, the parameter did not change much throughout the run and stayed around a value of 3 which is typical for initial populations of randomly

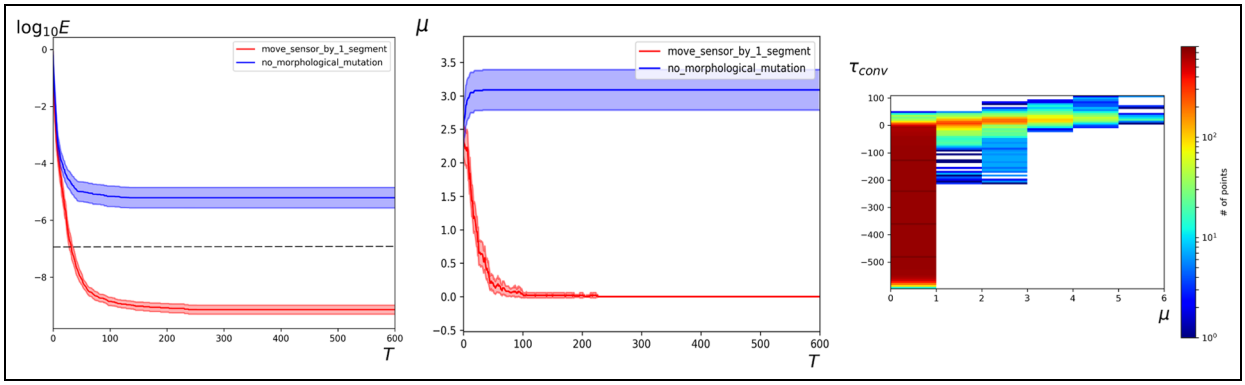

Figure 6. Evolving morphology alongside control facilitates the convergence of evolution of the latter and leads to a morphology admitting modular control. (Left) Error time series for evolution without morphological mutations ($P_{mm} = 0$, blue line and error strip) and with morphological mutations that move a randomly selected sensor by one segment in random direction ($P_{mm} = 0.2$, red line). The time series were obtained by evolving three-segment Arrowbots with a population size of 50. Solid lines represent averages and 95% confidence intervals of the decimal logarithm of error based on a sample of 100 evolutionary runs. Dashed line represents the baseline level of performance for $N = 3$ that is achievable for any morphology (see Appendix 2). (Middle) Time series of the minimal Hamming distance μ to the morphology $J = I$ across the Pareto front for the same setups. The initial change is due to the evolution using the diversity of the morphologies present within the initial population. (Right) Defining the moment of convergence as the generation when the error goes below the baseline performance level and the time to convergence τ_{conv} as the difference between the current time and this moment, we consider the state of the population at each generation as a point on the $\mu - \tau_{conv}$ plane. The figure is the density plot for such points. It can be observed that in the majority of runs, convergence occurs after the morphology $J = I$ is found ($\mu = 0$).

generated genomes. However, with morphological mutation, the parameter μ reached 0 in every run, consistent with Hypothesis 2.

We have also investigated the causal relationship between convergence and the discovery of $J = I$ morphology. We arbitrarily defined the moment of convergence as generation t_b when the error goes below the baseline performance of a hand-designed controller (see Appendix 2). We define τ_{conv} as the difference between the current evolutionary time and this moment

$$\tau_{conv} = t_b - t \tag{9}$$

Note that if evolution has reached the baseline performance in the past, τ_{conv} is negative. For every generation of every run in P_{mm} group, we considered the position of the population on $\mu - \tau_{conv}$ plane. The density of the points is shown in Figure 6 (right panel). It can be seen that convergence ($\tau_{conv} \leqslant 0$) in the majority of cases occurred after discovering the $J = I$ morphology, and in almost all cases after coming within a Hamming distance of 1 from it. It can also be seen that even after the $J = I$ morphology is found, it took more than 50 generations for some runs to converge (compared with the maximum convergence time of about 200 (data not shown)). These results are consistent with Hypothesis 1, but suggest the following clarification of Hypothesis 0: Approaching a morphology that admits modular control is usually a necessary but not sufficient condition for rapid convergence of modularity-biased control evolution.

Table 1. Parameters of evolutionary runs for experiments involving Arrowbots of different sizes.

Number of segments N	Population size	Total evolutionary time
3	50	600
5	100	1600
10	200	4000

Next, we investigated whether the outcome of evolution depends on whether the connection cost objective is used. To do that, we investigated the dependence of the behavior of the system on the probability of connection cost to be taken into account when deciding on dominance, P_{CC} (Figure 7(a)). Dependence was investigated for three values of robot size ($N = 3, 5, 10$) with the population size and maximum number of generations selected separately for each value to account for the disparity in task difficulty (Table 1). The populations were initialized with dense networks as described in section 3. For each value of N, we selected five equidistant temporal slices at which we measured two properties of the population—smallest achieved error E and parameter μ. We varied P_{CC} across $\{0, 0.05, 0.1, \ldots, 1\}$ and observed that for all values of N there is a sharp decrease in both error and μ as P_{CC} approaches 1. This suggests that the presence of bias toward modularity is crucial for evolving the morphologies that admit modular control and for achieving rapid convergence of control evolution, consistent with Hypotheses 4 and 5.

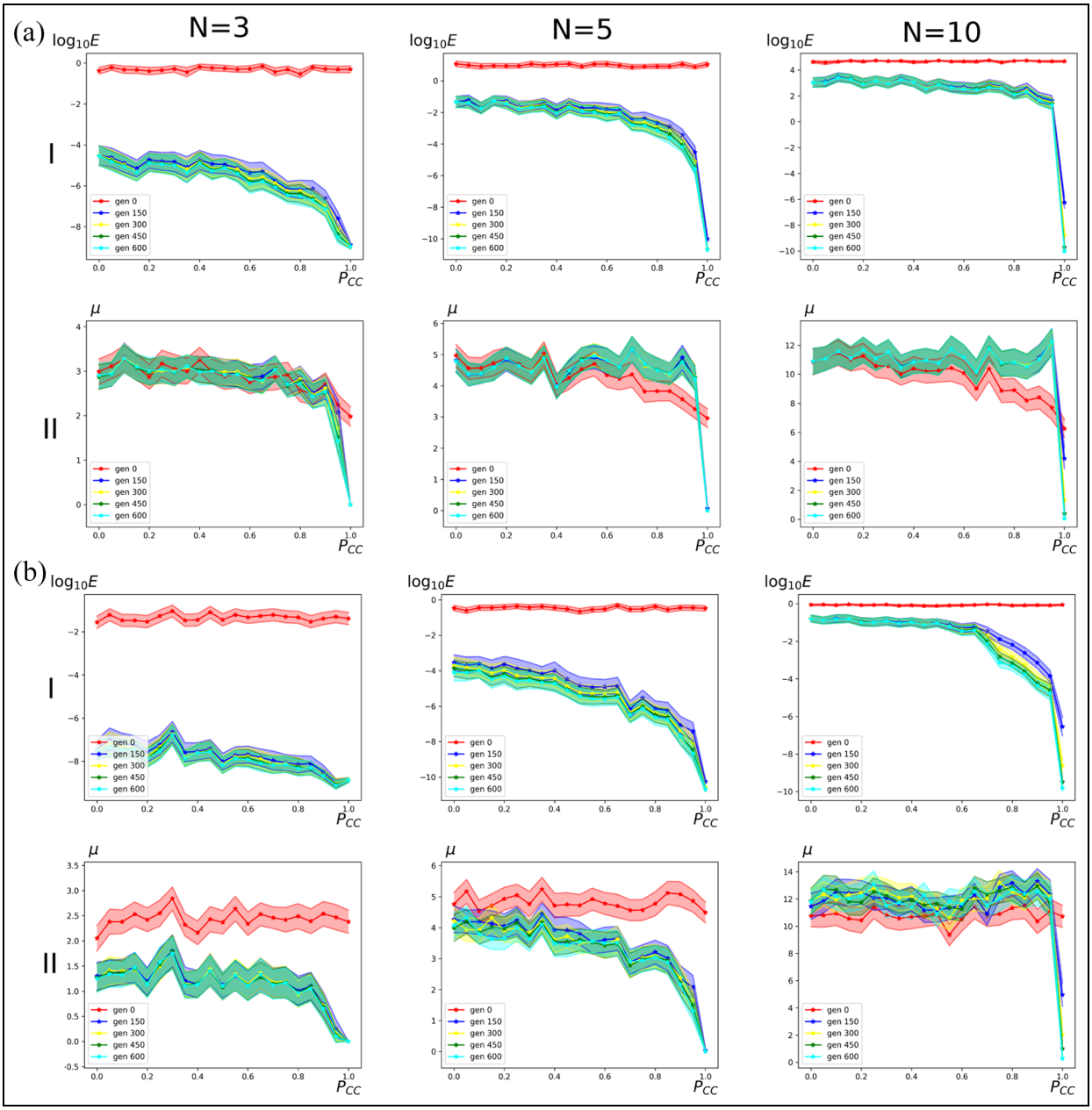

Figure 7. (a) Parameters of populations as a function of probability of connection cost to be taken into account when comparing individuals, P_{CC}, with initial populations of dense networks. Columns correspond to numbers of segments in Arrowbots N (see Table 1 for details about evolutionary algorithm's parameters). Top (I) rows shows the average decimal logarithm of error $\log_{10} E$ and its 95% confidence interval computed using data from 100 evolutionary runs; bottom (II) rows shows the average minimal distance μ to the morphology $J = I$ across the stochastic Pareto front and its 95% confidence interval. Different lines represent values measured at different points of evolutionary time (see legend). It can be seen that the convergence to the morphology $J = I$ (corresponding to $\mu = 0$) coincides with the lowest observed errors and is reached reliably only for $P_{CC} = 1$. (b) Same, but with an initial population of sparse networks. This modification decreases the error more rapidly and approaches the $J = I$ morphology for all values of P_{CC}, not just $P_{CC} \approx 1$. However, the impact of initial population decreases as the task is scaled up.

In addition, we have found that the decrease in the error and μ becomes more pronounced as the task is scaled up. This suggests that for more complex tasks, bias toward modularity is more crucial than for the simpler ones.

We also investigated whether the approach relies on the particular bi-objective performance and connection cost technique, or if it is agnostic with respect to the particular biasing technique. Previously, we demonstrated that initializing the population with sparse networks can cause modularity to evolve even in the absence of the connection cost objective, albeit modularity tends to decrease at the later stages of evolution under such conditions (Bernatskiy & Bongard, 2015).

This suggests that initializing the population with sparse networks can be an efficient way to bias evolution toward modular networks, although the efficiency of the approach may decrease as the number of generations required to reach an optimal solution increases (e.g. for more complex tasks).

We repeated the experiments detailing the dependence of evolutionary dynamic on P_{CC} with initial populations of sparse networks and found that evolution exhibits smaller final errors and more rapid convergence for all values of P_{CC}, although the effect decreased as the task was scaled up (Figure 7(b)). In particular, for $N = 3$, the final error for $P_{CC} = 0$ (connection cost not used) was comparable to the final error of the runs initialized with dense networks with $P_{CC} = 1$ (fully bi-objective approach). The runs with sparse populations also exhibited on average smaller minimal distance from the $J = I$ morphology to the Pareto front, μ. For greater values of N, the differences were qualitatively the same, but of magnitude that decreased with N. This is consistent with Hypothesis 6.

Finally, we investigated the relationship between the probability of mutation to be morphological, P_{mm}, and the evolutionary behavior of the system (Figure 8(a)). We found that convergence of the morphology to $J = I$ occurs for most values of P_{mm} within $(0, 1)$, except for values that are very close to 0 or 1. However, lower values of P_{mm} lead to a more rapid error reduction and make the discovery of $J = I$ occur more rapidly than for higher values of P_{mm}. This is consistent with Hypothesis 7. The data we currently have do not allow to reliably measure the optimal value of P_{mm}, but the value appears to not depend strongly on the number of segments N.

We also checked how the adaptation rate depends on the morphological mutation operator. To that end, we repeated the P_{mm} investigation with the mutation operator replaced by a random jump in the morphological space (Figure 8(b)). It can be seen that the $J = I$ morphology is still found and that evolution still reaches the baseline level, although it takes longer, especially as the task is scaled up. This suggests that the complexity of the control optimization task is sufficient to justify random search of morphology, especially for lower values of N. As the task is scaled up, however, the design of the morphological mutation operator starts to play a more significant role. Another observation is that if the morphological mutation is a random jump, then the number of generations required to perform the search in the morphological space increases and so does the optimal value of probability P_{mm}.

5. Discussion

We have shown that at least for one task and family of robot morphologies, it is possible to discover, via evolutionary search, a morphology admitting highly modularized and successful control by evolving the morphology alongside the control in a particular manner (Hypothesis 2). We found that biasing evolution toward modular controllers is crucial for finding such a morphology (Hypotheses 4 and 5), that the change in implementation of the bias does not prevent the morphology from being found (Hypothesis 6), and that under such bias and once such a morphology is found, rapid convergence of control toward a modular solution is likely to occur (Hypotheses 1 and 3). In addition, we have found that slowing the morphological change relative to change in control is beneficial for the occurrence of the phenomenon (Hypothesis 7), unless the morphological mutation operator is very inefficient (random jump).

Those results are consistent with our previous findings that morphology can profoundly influence the difficulty of evolving modular control and describe conditions that make modular control more likely to evolve. By supporting these findings, our results also indirectly support the idea of morphological computation (Pfeifer, Iida, & Gómez, 2006), as they represent a case of radical simplification of control due to the choice of morphology.

These results can be used in engineering, in particular, in evolutionary robotics, where they may allow the designers to place a greater number of heterogeneous design variables under the control of evolution, while avoiding, through modularity, the requirement for a combinatorially large number of evaluation environments (Cappelle et al., 2016) and the problem of catastrophic forgetting (Ellefsen et al., 2015) and thus retaining the capacity of evolution to find good-enough solutions in reasonable time. For example, these results may be of interest to engineers and researchers who wish to evolve whole physical agents, complete with morphology, circuitry, and actuator design.

The results also have implications for biology, as living systems satisfy the basic requirements of our approach to the evolution of modular control: The bias toward modular nervous systems is present in living systems in the form of metabolic connection cost (Clune et al., 2013), and morphology is evolved alongside the control. Based on that, we speculate that morphological evolution might be a factor in evolution of modularity of nervous systems.

To formulate our approach to evolving modular control, we formalized the reformulation approach to problem-solving (Choueiry et al., 2005; Sternberg & Davidson, 1995; see section 2). Within the reformulation framework, morphology is one example of a more general entity which we term driving variables, defined as design parameters that influence the search difficulty of finding the best overall values of the remaining (non-driving) variables. Finding such variables requires domain knowledge, but once they are found, they can be optimized to reduce the search difficulty of

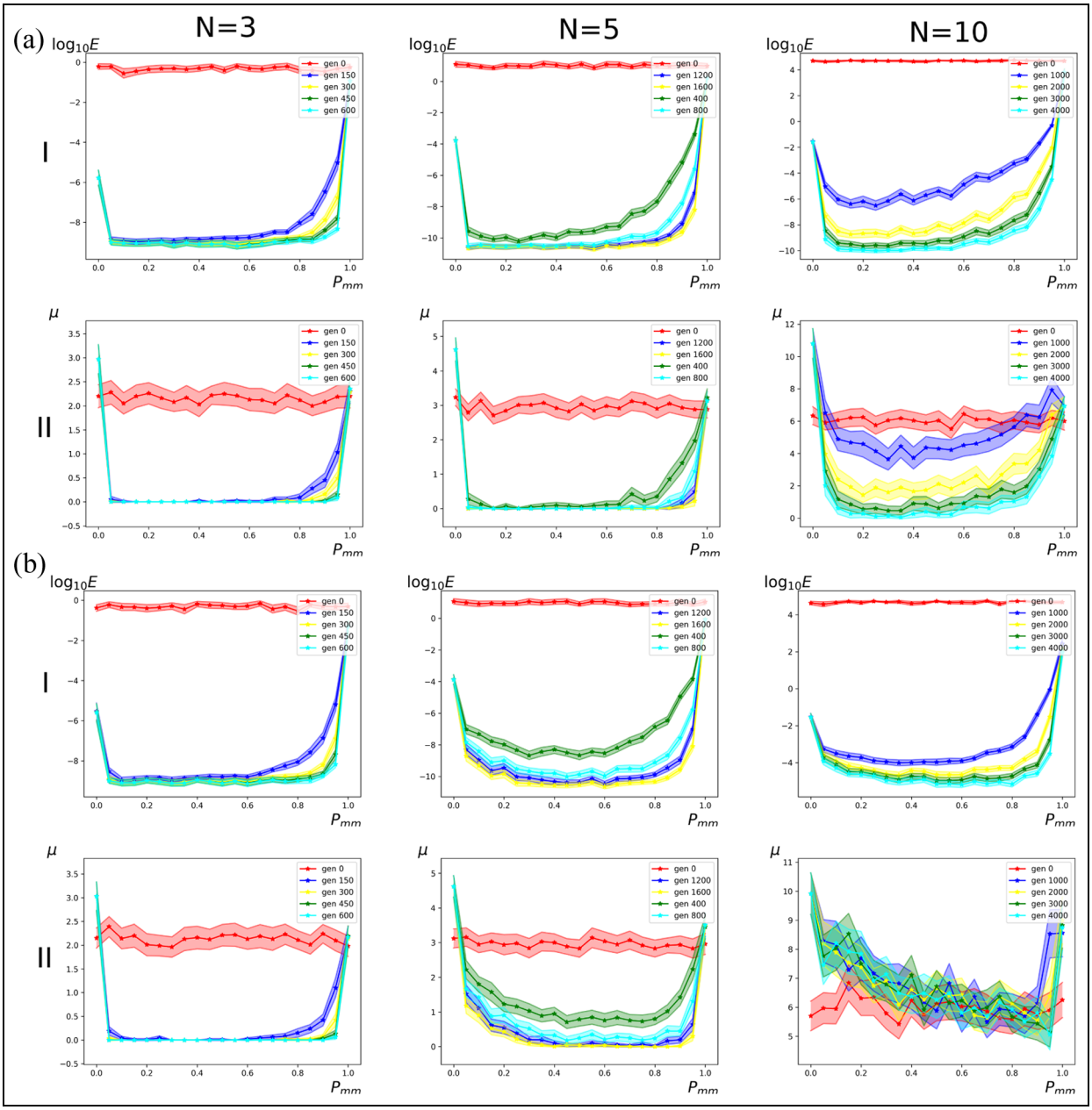

Figure 8. (a) Parameters of populations as a function of probability of mutation to be morphological P_{mm}. Columns correspond to numbers of segments in Arrowbots N (see Table 1 for details about evolutionary algorithm's parameters). Top (I) row shows the decimal logarithm of error $\log_{10} E$ and its 95% confidence interval computed using data from 100 evolutionary runs; bottom (II) row shows the minimal distance μ to the morphology $J = 1$ across the Pareto front and its 95% confidence interval. Different lines represent values measured at different points of evolutionary time (see legend). It can be seen that the convergence to $J = 1$ is reached for a wide range of P_{mm} values; however, this process occurs most rapidly for lower values of P_{mm}, resulting in lower errors being achieved earlier on. The effect gets more pronounced as the task is scaled up. (b) Same, but with the morphological mutation replaced by a random jump in the morphospace. It can be seen that convergence does occur even if the space of morphologies is searched randomly, although the performance of this approach suffers more as the task is scaled up, compared to the mutation operator that moves a sensor by one segment.

optimizing the remaining variables and thus "reformulate" the corresponding optimization problem. We extend the reformulation approach with a qualitative rule of thumb that suggests that the driving variables can be found by examining the influence of the design variables on the behavior of optimization of non-driving variables under biases. We call the resulting approach guided reformulation. The method for evolving modular control then follows from the hypothesis that morphology influences the effectiveness of evolution if a bias toward modularity is present. We speculate that the guided reformulation approach can

be applied to reformulating a wider range of optimization problems than the problem of controlling embodied agents, possibly using optimization biases other than the bias toward modularity.

We briefly review one mechanism of optimization difficulty reduction, the division approach, in section 1. We link it with solution modularity and explain why morphology can influence this mechanism, thus establishing morphology as a candidate for a vector driving variable. Our set of testable hypotheses about the properties of the resulting optimization approach is based on this analysis.

We see three challenges for applying the reformulation approach and guided reformulation to a wider range of tasks. First, a possibility of premature convergence arises if the method is used with a definition of optimization difficulty that can mistakenly assign low difficulty to driving variable values that prohibit convergence of optimization of non-driving variables to good-enough solutions. Our instantiation of reformulation based on morphology and modularity-biased control optimization uses such a definition, and we plan to correct that in our future work using methods similar to morphological protection (Cheney et al., 2017). Second, despite the guidance provided by the guided reformulation rule-of-thumb, the identification of driving variables currently must be performed in an ad hoc manner by a human designer. Systematizing and/or automating the identification of driving variables will be the focus of our future studies. Third, for more complex tasks, a good mutation operator for driving variables is required. At present, there is no systematic way to assess or design such operators.

Acknowledgments

The computational resources provided by the UVM's Vermont Advanced Computing Core (VACC) are gratefully acknowledged.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by NSF awards INSPIRE-1344227.

ORCID iD

Anton Bernatskiy https://orcid.org/0000-0002-7624-1615

Supplemental material

See the following GitHub repository for all the materials and instructions required to reproduce our results: https://github.com/abernatskiy/reformulation2018

Notes

1. The term was chosen to distinguish from divide-and-conquer methodology in algorithmic design. Division approach is a broader term that encompasses all useful ways to split the problem into subproblems, while divide-and-conquer is its instance that focuses on non-overlapping subproblems. Approaches that involve overlapping subproblems, such as dynamic programming, are also instances of the division approach.
2. Note that in our treatment, all orientations do not wrap around, that is, angles ϕ radians and $\phi + 2k\pi$ radians, $k \in \mathbb{Z}$ are treated as *different*. In this treatment, there are no special angle values and the need to keep track of angle units is eliminated.

References

Ashby, W. R. (1960). *Design for a brain: The origin of adaptive behavior* (2nd ed.). London, England: Chapman & Hall.

Axelrod, R. (1987). The evolution of strategies in the iterated prisoners dilemma. In C. Bicchieri, R. Jeffrey, & B. Skyrms (Eds.), *The dynamics of norms* (pp. 1–16). Cambridge, UK: Cambridge University Press.

Baldwin, C. Y., & Clark, K. B. (2006). Modularity in the design of complex engineering systems. In D. Braha, A. A. Minai, & Y. Bar-Yam (Eds.), *Complex engineered systems* (pp. 175–205). Berlin, Germany: Springer.

Bernatskiy, A., & Bongard, J. C. (2015). Exploiting the relationship between structural modularity and sparsity for faster network evolution. In S. Silva (Ed.), *Proceedings of the companion publication of the 2015 annual conference on genetic and evolutionary computation* (pp. 1173–1176). New York, NY: ACM.

Bernatskiy, A., & Bongard, J. C. (2017). Choice of robot morphology can prohibit modular control and disrupt evolution. In C. Knibbe, G. Beslon, D. Parsons, D. Misevic, J. Rouzaud-Cornabas, N. Bredèche, . . . H. Soula (Eds.), *Proceedings of the 14th European conference on artificial life* (pp. 60–67). Cambridge, UK: MIT Press.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Berlin, Germany: Springer.

Bongard, J. C., Bernatskiy, A., Livingston, K., Livingston, N., Long, J., & Smith, M. (2015). Evolving robot morphology facilitates the evolution of neural modularity and evolvability. In S. Silva (Ed.), *Proceedings of the 2015 annual conference on genetic and evolutionary computation* (pp. 129–136). New York, NY: ACM.

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

Cappelle, C. K., Bernatskiy, A., Livingston, K., Livingston, N., & Bongard, J. (2016). Morphological modularity can enable the evolution of robot behavior to scale linearly with the number of environmental features. *Frontiers in Robotics and AI*, *3*, 59.

Carroll, S. B. (2001). Chance and necessity: The evolution of morphological complexity and diversity. *Nature*, *409*, 1102–1109.

Cheney, N., Bongard, J., SunSpiral, V., & Lipson, H. (2017). *Scalable co-optimization of morphology and control in embodied machines*. Retrieved from https://pdfs.semantic

scholar.org/2ef5/6896200072eb64f3e9076c4bf74600760bb2.pdf

Choueiry, B. Y., Iwasaki, Y., & McIlraith, S. (2005). Towards a practical theory of reformulation for reasoning about physical systems. *Artificial Intelligence*, *162*, 145–204.

Clune, J., Mouret, J.-B., & Lipson, H. (2013). The evolutionary origins of modularity. *Proceedings of the Royal Society B:Biological Sciences*, *280*. doi:10.1098/rspb.2012.2863

Draghi, J., & Wagner, G. P. (2008). Evolution of evolvability in a developmental model. *Evolution*, *62*, 301–315.

Duncker, K., & Lees, L. S. (1945). On problem-solving. *Psychological Monographs*, *58*(5), i.

Durr, P., Floreano, D., & Mattiussi, C. (2010). Genetic representation and evolvability of modular neural controllers. *IEEE Computational Intelligence Magazine*, *5*, 10–19.

Ellefsen, K. O., Mouret, J.-B., & Clune, J. (2015). Neural modularity helps organisms evolve to learn new skills without forgetting old skills. *PLoS Computational Biology*, *11*(4), e1004128.

Espinosa-Soto, C., & Wagner, A. (2010). Specialization can drive the evolution of modularity. *PLoS Computational Biology*, *6*(3), e1000719.

Fairhall, A. L., Lewen, G. D., Bialek, W., & de Ruyter Van Steveninck, R. R. (2001). Multiple timescales of adaptation in a neural code. In M. I. Jordan (Ed.), *Advances in neural information processing systems* (pp. 124–130). Cambridge, UK: The MIT Press.

Floreano, D., & Mondada, F. (1994). Automatic creation of an autonomous agent: Genetic evolution of a neural network driven robot. In D. Cliff, P. Husbands, J.-A. Meyer, & S. W. Wilson (Eds.), *Proceedings of the 3rd international conference on simulation of adaptive behavior: From animals to animats 3* (pp. 421–430). Cambridge, UK: MIT Press.

Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, *99*, 7821–7826.

Hawkins, J., & Blakeslee, S. (2007). *On intelligence: How a new understanding of the brain will lead to the creation of truly intelligent machines*. New York, NY: Palgrave Macmillan.

Hinton, G. E., & Nowlan, S. J. (1987). How learning can guide evolution. *Complex Systems*, *1*, 495–502.

Huxley, J. (1942). *Evolution: The modern synthesis*. Crows Nest, Australia: George Allen & Unwin.

Kashtan, N., & Alon, U. (2005). Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences of the United States of America*, *102*, 13773–13778.

Kiebel, S. J., Von Kriegstein, K., Daunizeau, J., & Friston, K. J. (2009). Recognizing sequences of sequences. *PLoS Computational Biology*, *5*(8), e1000464.

Kleinberg, J., & Tardos, E. (2005). *Algorithm design*. London, England: Pearson.

Knoblich, G., Ohlsson, S., & Raney, G. E. (2001). An eye movement study of insight problem solving. *Memory & Cognition*, *29*, 1000–1009.

Kounios, L., Clune, J., Kouvaris, K., Wagner, G. P., Pavlicev, M., Weinreich, D. M., & Watson, R. A. (2016). Resolving the paradox of evolvability with learning theory: How evolution learns to improve evolvability on rugged fitness landscapes. arXiv preprint arXiv:1612.05955.

Kriegman, S., Szubert, M., Bongard, J. C., & Skalka, C. (2016). Evolving spatially aggregated features from satellite imagery for regional modeling. In J. Handl, E. Hart, P. R. Lewis, M. López-Ibáñez, G. Ochoa & B. Paechter (Eds.), *International conference on parallel problem solving from nature* (pp. 707–716). Berlin, Germany: Springer.

Lipson, H., Pollack, J. B., Suh, N. P., & Wainwright, P. (2002). On the origin of modular variation. *Evolution*, *56*, 1549–1556.

Martin, P., Bateson, P. P. G., & Bateson, P. (1993). *Measuring behaviour: An introductory guide*. Cambridge, UK: Cambridge University Press.

Matarić, M. J. (1998). Behavior-based robotics as a tool for synthesis of artificial behavior and analysis of natural behavior. *Trends in Cognitive Sciences*, *2*, 82–86.

Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., & . . . Hodjat, B. (2017). Evolving deep neural networks. arXiv preprint arXiv:1703.00548.

Newman, M. E. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, *103*, 8577–8582.

Nolfi, S., & Floreano, D. (1998). Coevolving predator and prey robots: Do arms races arise in artificial evolution?*Artificial Life*, *4*, 311–335.

Orzack, S., Orzack, S. H., & Sober, E. (2001). *Adaptationism and optimality*. Cambridge, UK: Cambridge University Press.

Pfeifer, R., & Bongard, J. (2006). *How the body shapes the way we think: A new view of intelligence*. Cambridge, UK: MIT press.

Pfeifer, R., Iida, F., & Gómez, G. (2006). Morphological computation for adaptive behavior and cognition. *International Congress Series*, *1291*, 22–29.

Rosin, C. D., & Belew, R. K. (1997). New methods for competitive coevolution. *Evolutionary Computation*, *5*, 1–29.

Solé, R. V., & Fernández, P. (2003). *Modularity "for free" in genome architecture?* Retrieved from https://pdfs.semanticscholar.org/84c2/43d31437c884e9bff708fc8db4ff7203e322.pdf

Stanley, K. O., & Miikkulainen, R. (2003). A taxonomy for artificial embryogeny. *Artificial Life*, *9*, 93–130.

Sternberg, R. J., & Davidson, J. E. (1995). *The nature of insight*. Cambridge, UK: MIT Press.

Strabo (AD23)*Geography 1903* (H. C., Hamilton, & W. M. A., Falconer, Trans.). Medford, MA: Perseus Digital Library.

Suh, N. P. (1990). *The principles of design* (Number 6). Oxford, UK: Oxford University Press.

Tinbergen, N. (1963). On aims and methods of ethology. *Ethology*, *20*, 410–433.

Ulanovsky, N., Las, L., Farkas, D., & Nelken, I. (2004). Multiple time scales of adaptation in auditory cortex neurons. *Journal of Neuroscience*, *24*, 10440–10453.

Waddington, C. H. (1942). Canalization of development and the inheritance of acquired characters. *Nature*, *150*, 563–565.

Wagner, G. P., Pavlicev, M., & Cheverud, J. M. (2007). The road to modularity. *Nature Reviews Genetics*, *8*, 921–931.

Wark, B., Fairhall, A., & Rieke, F. (2009). Timescales of inference in visual adaptation. *Neuron*, *61*, 750–761.

Watson, R. A., & Szathmáry, E. (2016). How can evolution learn?*Trends in Ecology & Evolution*, *31*, 147–157.

Welch, J. J., & Waxman, D. (2003). Modularity and the cost of complexity. *Evolution*, *57*, 1723–1734.

Wicker, F. W., Weinstein, C. E., Yelich, C. A., & Brooks, J. D. (1978). Problem-reformulation training and visualization training with insight problems. *Journal of Educational Psychology*, *70*, 372–377.

Yamashita, Y., & Tani, J. (2008). Emergence of functional hierarchy in a multiple timescale neural network model: A humanoid robot experiment. *PLoS Computational Biology*, *4*(11), e1000220.

Appendix 1

Notation

A_i	absolute orientation of ith segment
$E(t)$	practical pointing error (see equation (7)) minimized across the population at generation t
J	$N \times N$ target orientation sensor attachment matrix (see equation (3))
K	constant $N \times N$ matrix filled with ones on and below the main diagonal and zeros everywhere else (see equation (2))
m_i	motor output for ith segment (see equation (5))
N	number of segments
r_i	relative orientation of ith segment (see equation (1))
s_i	reading of the target orientation sensor measuring the orientation of the ith segment relative to the body the sensor is attached to (see equation (3))
$\boldsymbol{T}, \boldsymbol{A}, \boldsymbol{r}, \mathbf{s}, \mathbf{m}$	corresponding N-dimensional vectors
T_i	target orientation of ith segment
W, Y	matrices of coefficients of a linear controller (see equation (4))
$\mu(t)$	Hamming distance to the morphology $J = I$ minimized across the Pareto front observed at generation t of the evolution

Appendix 2

Baseline performance for an arbitrary morphology

Theorem. It is possible to construct a linear controller satisfying equation (4) that reduces the ideal error (equation (6)) to 0 for every valid sensor attachment matrix J.

Proof. Substituting equations (3) and (2) into equation (4), we get

$$\begin{aligned} \dot{r} &= W\mathbf{s} + Yr = WT - WJKr + Yr \\ &= WT + (Y - WJK)r \end{aligned} \tag{10}$$

At equilibrium, $r = K^{-1}T$ and

$$\begin{aligned} &WT + (Y - WJK)K^{-1}T = 0 \\ &Y = W(J - I)K \end{aligned} \tag{11}$$

Assuming this condition holds, equation (10) simplifies to

$$\begin{aligned} \dot{r} &= WT + (W(J - I)K - WJK)r \\ &= WT - WKr \end{aligned} \tag{12}$$

Consider a controller with $W = I$: a matrix with no values outside of $\{-1, 0, 1\}$. The Jacobian of the system then becomes $-K$, making the system stable.

Next, we verify that for each such controller, a matrix Y exists such that all its elements are in $\{-1, 0, 1\}$. From equation (11)

$$Y = (J - I)K \tag{13}$$

Each row of matrix $J - I$ contains at most one entry equal to -1 and one equal to 1, with the rest of entries being zeros. Multiplying that by the ith column of K will produce a sum of all entries to the right of the ith field, including the ith field itself. There will be at most two nonzero components in this sum, one equal to -1 and one equal to 1. Such a sum necessarily lies in $\{-1, 0, 1\}$. ■

The Jacobian for all controllers constructed in this way is the same, thus the temporal trajectories of the systems are the same. We verified that this is true by evaluating the controllers constructed as described above for 100,000 randomly picked morphologies for the three values of $N = 3, 5, 10$. Within each group, all logarithms of the pointing error equation (7) coincided up to 10th decimal point: $\log_{10} E(N = 3) = -6.9408767850$, $\log_{10} E(N = 5) = -6.0563588899$, and $\log_{10} E(N = 10) = -5.3115293183$.

We will use these values to define a baseline for controller performance. Such definition has the following property: given N, we can guarantee that for any morphology there exists a controller that achieves the pointing error no greater than the baseline.

About the Authors

Anton Bernatskiy is currently a PhD student at University of Vermont. He holds a Specialist degree in Physics from Lomonosov Moscow State University. His main interests lie in the field of embodied AI and its applications to space exploration; however, he has also done some work in ocean physics and Monte Carlo algorithms for simulating chemical systems.

Josh Bongard is the Veinott professor of Computer Science at the University of Vermont. He also serves as the director of the Morphology, Evolution & Cognition Laboratory, and the Vermont Advanced Computing Core, UVM's high-performance computing facility. He has a background in computer science, cognitive science, and evolutionary and adaptive systems. His interests lie in the fields of evolutionary computation, evolutionary robotics, and machine science. More specifically, his work often involves investigations in embodied approaches to evolving intelligent behavior in machines. As a secondary research interest, he is investigating crowdsourcing approaches to evolutionary robotics and the resulting synergies between evolutionary and social dynamics. He is the author of the book *How the Body Shapes the Way We Think* and runs an evolutionary robotics online course at reddit.com.