Reducing Training Environments in Evolutionary Robotics Through Ecological Modularity

Collin Cappelle(✉), Anton Bernatskiy, and Josh Bongard

The University of Vermont, Burlington, VT 05405, USA
collin.cappelle@uvm.edu

Abstract. Due to the large number of evaluations required, evolutionary robotics experiments are generally conducted in simulated environments. One way to increase the generality of a robot's behavior is to evolve it in multiple environments. These environment spaces can be defined by the number of free parameters (f) and the number of variations each free parameter can take (n). Each environment space then has n^f individual environments. For a robot to be fit in the environment space it must perform well in each of the n^f environments. Thus the number of environments grows exponentially as n and f are increased. To mitigate the problem of having to evolve a robot in each environment in the space we introduce the concept of *ecological modularity*. Ecological modularity is here defined as the robot's modularity with respect to free parameters in its environment space. We show that if a robot is modular along m of the free parameters in its environment space, it only needs to be evolved in n^{f-m+1} environments to be fit in all of the n^f environments. This work thus presents a heretofore unknown relationship between the modularity of an agent and its ability to generalize evolved behaviors in new environments.

1 Introduction

One of the major challenges to evolutionary robotics in particular, and evolutionary computation in general, is the relatively slow rate of convergence toward acceptable solutions due to these algorithms' stochastic elements. This challenge is exacerbated when robust behavior is desired: In such cases robots must be evolved in multiple environments until the robots exhibit the desired behavior in all of them. However, because of catastrophic forgetting [8], it is not usually possible to evolve robots in one environment, discard that environment, continue evolving them in a different environment, and have them retain their ability to succeed in the first environment. Thus, robots must be trained in some set of static environments, or gradually exposed to a growing set of training environments over evolutionary time. [14] pointed out convergence time explodes in such multiple-environment contexts because of the combinatorics of parametrically-defined environments. Typically, a set of training environments is generated before evolution commences by defining a number of free parameters f, which

M. Mangan et al. (Eds.): Living Machines 2017, LNAI 10384, pp. 95–106, 2017.
DOI: 10.1007/978-3-319-63537-8_9

represent aspects of the environment that change from one to another. For each of these free parameters, there are n possible settings. For example, given an object in the environment, a free parameter could be the starting position of that object. If the object may have different sizes as well as starting positions in a given environment from taken for the total set of possible environments, then $f = 2$. If there are two possible sizes, and two different starting positions, then $n = 2$. [14] showed that if we wish evolved robots to succeed in all environments defined for a given f and n, then the robots will have to be evolved in n^f environments.

1.1 Robustness

Much work has been done to increase the robustness of evolved behavior in robots. For instance, Jakobi [10] investigated the introduction of noise to guard against evolutionary exploitation of any inaccuracies in the simulator used to the evolve the behaviors. Lehman [12] demonstrated experiments in which explicit selection pressure was exerted on robots to respond to their sensor input, thus ensuring that evolved robots would behave differently when placed in different environments where they could sense the changes. Bongard [2] demonstrated that robots with ancestors that changed their body plans during their lifetimes tended to be more robust than robots with fixed-morphology ancestors, because the former lineages tended to experience wider ranges of sensorimotor experiences than the latter lineages. However, these and similar works did not investigate the role that modularity might play in the evolution of robust behavior. One exception is the work of Ellefsen *et al.* [6], in which an evolutionary cost is placed on the synapses of disembodied neural networks trained to compute logical functions. They had previously shown that such connection cost tends to lead to the evolution of modular networks [5], and, in [6], this neural modularity enabled evolved networks to rapidly adapt to new environments without losing their ability to succeed in the original environments.

1.2 Modularity

Like robust behavior, the ubiquity of modularity in evolved systems has spawned an active literature. Work in this area can be divided into investigations into the evolution of modularity in disembodied systems and embodied systems, such as robots.

Wagner [17] forwarded a theoretical argument that modularity evolves when systems experience combinations of directional and stabilizing selection on different parts of their phenotypes. This was subsequently verified by experiments using non-embodied data structures [13], neural networks [5,11], and models of gene networks [7].

Investigations into the evolution of modularity in embodied systems begin with Gruau [9], who employed an indirect genotype to phenotype mapping that allowed for the construction of neural modules in a robot. Yamashita et al. [18] demonstrated robots capable of learning independent motor primitives and then

combining them in novel sequences. In [1,3], Bongard *et al.* showed how to evolve structurally modular neural controllers for autonomous robots.

However, none of these approaches investigate the relationship between both morphological and neurological modularity as a way to increase generalization. That is, how the structure of the robot's morphology and controller may interact with the environment to reduce the minimum number of environments robots must be evolved in to generalize across the entire environment space.

1.3 Morphological and Neurological Modularity

Modularity has shown to be important in evolution of networks and robots because it helps the agent avoid catastrophic forgetting when presented with a new environment [6]. Catastrophic forgetting is a problem when, in order to learn a new task, an agent must forget what it previously learned [8].

However, most of the modularity research in robotics has focused on modularity with respect to the controller of the robot. Most often the controller is a neural network so network metrics are used. Most notably the Q-metric has been used to define modularity in networks [15]. Q is a metric which measures the fraction of edges which fall between within a group subtracted by the expected fraction of edges within that group given a random network with the same degree distribution. However, Q disregards many aspects of the morphology and control of robots which may be important in determining if the robot is made up of actual useful modules.

More recent work has defined both neural and morphological modularity in terms of the sensor-motor feedback loop [4]. It was shown that the number of necessary training environments for robots that are morphologically and neurologically modular in this manner grows less rapidly than the number of necessary training environments in non-modular cases when the number of free parameters, f, was held constant and the number of variations, n, was increased.

In this work, we build upon this research by holding n constant and increasing f. Also, we continue to use those definitions of neurological and morphological modularity and expand them by considering the robot's interactions with its environment space a property which we here term 'ecological modularity'.

1.4 Ecological Modularity

We define the following terms and variables to be used throughout the paper:

- **F** - The set of free parameters in the system with cardinality f. Free parameters are the dimensions of the environment space which change.
- **n** - number of variations of each free parameter in F. Because we are only considering free parameters that vary, $n \geq 2$. For simplicity, all free parameters are assumed to have the same number of variations.

- **Discrete Environment Space** - The set, E, comprised of all the possible combinations of free parameter variations. These are environment spaces which can be discretized and organized into an f-dimensional hypercube with n^f hypervoxels each corresponding to one individual environment. Therefore there are a total of n^f environments in E. Each environment can therefore be defined as a f-tuple consisting of the variations of each free parameter.
- **Orthogonal Environments** - Orthogonal environments are those in which none of the variations of the free parameters are equal. Thus given two environments e_1 and e_2, $\pi_{e_1}^{(j)} \neq \pi_{e_2}^{(j)}$ for all j in F. For example,

$$\left(\pi_1^{(1)}, \pi_1^{(2)}, \ldots, \pi_1^{(f)}\right) \perp \left(\pi_2^{(1)}, \pi_2^{(2)}, \ldots, \pi_2^{(f)}\right)$$

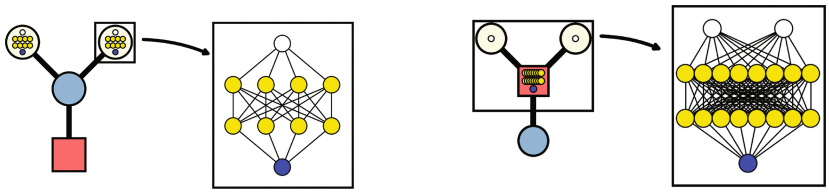
  because $\pi_1^{(j)} \neq \pi_2^{(j)}$ for each j.

- **Orthogonal Environments along a Subset of Free Parameters** - Let $D \subset F$. Then two environments, e_1, e_2, are orthogonal along D if for each $d \in D$, $\pi_{e_1}^{(d)} \neq \pi_{e_2}^{(d)}$.
- **Modularity along U Free Parameters** - Let U be a subset of F with cardinality u. Let $O_U \subset E$ represent a subset of orthogonal environments along U. A robot is said to be modular along U if, when the robot achieves sufficient fitness in all u environments in O_U, the robot will maintain its fitness in the remaining environments where the variations along the $F \backslash U$ free parameters remain fixed. We note $1 \leq u \leq f$ for every robot and environment space.
- **Ecological Modularity** - Let M be a subset of F with cardinality m such that M is the maximal subset of F a robot is modular with respect to. That is the robot is modular with respect to every free parameter in M but none of the free parameters in $F \setminus M$. Then ecological modularity is defined to be the degree to which the robot is modular with respect to its environment, m. Robots with $m = f$ are said to be fully ecologically modular, robots with $1 > m > f$ are said to be partially ecologically modular, and robots with $m = 1$ are said to be ecologically non-modular.

Using the definitions above, we claim the total number of environments necessary for a robot to be evolved in is $n^{(f-m+1)}$. Meaning when we have a robot which is fully ecologically modular ($m = f$) we only need to evolve the robot in n mutually orthogonal environments, the easiest example of which is the 'grand diagonal' of the hypercube representation of the environment space. When the robot is ecologically non-modular ($m = 1$) we need to evolve the robot in all n^f environments. The term $f - m + 1$ represents the number of free parameters the robot is not modular with respect to.

2 Methods

In this section we describe the structure of the environment spaces, robot design, evolutionary algorithm, and experimental design.



(a) $\mathcal{M}$ (b) $\mathcal{NM}$

Fig. 1. The modular (1a) and non-modular (1b) robots' morphology and control structures. The morphology consisted of fixed hinges (red squares), free hinges (large blue circles), and sensor nodes (large beige circles). The networks are presented blown up for each robot. They consisted of sensor (white circles), hidden (yellow circles), and motor (blue circles) neurons. Motor neuron output controls the hinge joint at the base of the node the motor neuron is in. Connections between layers were feed-forward and feed-back. There are also recurrent connections for each hidden and motor neuron not depicted. (Color figure online)

2.1 Robot Design

Robot Morphology. The robots were designed with a branching, hierarchical morphology. A tree structure was chosen because it is symmetric, can easily be made modular/non-modular by fixing different branch hinges, and is easily expandable. Each tree consisted of one root node and two leaf nodes. The root node was connected to a point in space by a hinge joint. The leaf nodes were connected to the root node by hinge joints. Sensors were distance sensors placed in the leaf nodes of the robot. When the robot was pointing at an object they returned the distance to that object. When the robot was not pointing at anything, the sensor values returned a default value of ten.

Each robot was composed of three cylinders, one root node and two leaf nodes, of length one. The base of each leaf node was attached to the tip of the root node. Robots were initially positioned such that the leaves were horizontally rotated $+45°$ and $-45°$ with respect to the root node. In this paper we explored two variations of robot morphology.

First is the modular morphology, $\mathcal{M}$. In the modular morphology, the root node of the robot is fixed while the leaf nodes of the robot are free to move. Each leaf could rotate horizontally $[-45°, +45°]$ with respect to its starting position.

Second is the non-modular morphology, $\mathcal{NM}$. In the non-modular morphologies, the root of the robot is free to move while its leaf nodes are fixed. The root could rotate horizontally $[-120°, +120°]$.

Robots were simulated using Open Dynamics engine.

Robot Controllers. Robots were controlled by artificial neural networks. All networks were layered networks with both feed-forward and feed-back synapses as well as recurrent connections on both the hidden neurons and motor neurons.

Different cognitive architectures were employed for robots with different morphologies. Each of these architectures are reported in Fig. 1. For the modular morphologies, each leaf node had a separate, self-contained network connecting the leaf sensor to the motor neuron in the leaf (Fig. 1a). Each leaf network consisted of the one sensor neuron, two hidden layers with four neurons each, and the one motor neuron.

For the non-modular morphologies, the network connected the two leaf sensor neurons to the one root motor neuron. This network consisted of the one sensor neuron, two layers with eight hidden neurons each, and the one motor neuron (Fig. 1b).

Sensor neurons could take values between $[0, 10]$. Hidden and motor neurons could take values between $[-1, 1]$. Sensors could take any real valued number. Neurons in the network were updated at each time step in the simulation. The value of each neuron was determined by

$$y_i^{(t)} = \tanh\left(y_i^{(t-1)} + \sum_{j \in J} w_{ji} y_j^{(t-1)}\right) \tag{1}$$

where y_i^t denotes the i neuron's new value at time step t. $y_i^{(t-1)}$ denotes that neurons value in the previous time step. w_{ji} denotes the weight of the synapse connecting neuron j to neuron i.

2.2 Environmental Setup

Environments consisted of two clusters of cylinders set up on the left and right of the robot such that on the first time step of simulation, the robot pointed at the center of each cluster as shown in Fig. 2. Cylinders were organized on a line segment perpendicular to the direction of the leaf nodes. Clusters were placed such that the robot was initially pointing at their center. The diameter of each cylinder was equal to the length of the line segment divided by the number of cylinders in the cluster. A small constant value of $\varepsilon = .1$ was then added to the

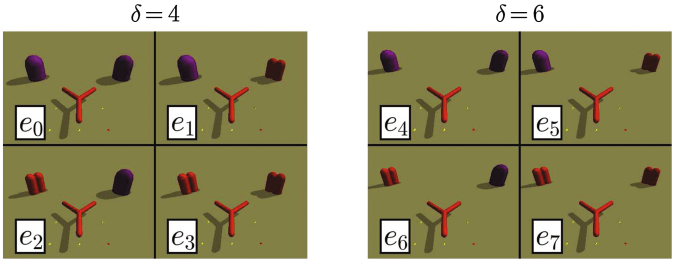


Fig. 2. The starting point of the robots in simulation for each environment. The environment space $E_2 = \{e_0, e_1, e_2, e_3\}$ is shown by the four left environments in the figure and $E_3 = \{e_0, e_1, \ldots, e_7\}$ is shown by all eight environments which make up the figure. The δ variable defines the initial distance of both clusters from the robot.

diameters so there were no gaps between cylinders in the cluster. Thus, each environment consisted of three free parameters:

- c_L: The number of cylinders in the left cluster. $c_L \in \{1, 2\}$.
- c_R: The number of cylinders in the right cluster. $c_R \in \{1, 2\}$.
- δ: The distance, in simulator units, the center point of each cluster is from the tip of its corresponding sides leaf node of the robot. $\delta \in \{4, 6\}$.

From the variables described above, we can categorize each environment as a 3-tuple (δ, c_L, c_R).

We can generate environment spaces by restricting which parameters are free and which are fixed. In this manner we generate two different environment spaces we are interested in:

- $E_2 = (\delta = 4, c_L = *, c_R = *)$
- $E_3 = (\delta = *, c_L = *, c_R = *)$

where $*$ indicates that parameter is free to vary. From this we see E_2 is a 2×2 environment space with four total environments and E_3 is a $2 \times 2 \times 2$ environment space with eight total environments.

We can then enumerate individual environments by the corresponding tuples parameter values. For example, we let $e_{(0,0,0)}$ represent an environment that consists of the first variation of each parameter, namely $e_{(0,0,0)} = e_0 = (\delta = 4, c_L = 1, c_R = 1)$. Thus $e_{(1,1,1)} = e_7 = (\delta = 6, c_L = 2, c_r = 2)$ and so on for each environment. All of the environments considered in this work are presented in Fig. 2.

2.3 Physical Implementation

The robot was also made in out of Legos as shown in Fig. 3. While the physical implementation can move and respond in the same manner as the simulation, it is still in development so no evolution was performed using the physical robot.

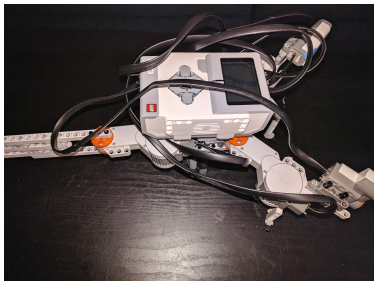


Fig. 3. Physical robot made out of Legos. Can represent the $\mathcal{M}$ or $\mathcal{NM}$ robot by fixing/freeing motors.

2.4 Evolutionary Setup

The goal of the robot was to point towards clusters containing an even number of cylinders and away from clusters containing an odd number of cylinders. This was implemented using a simple counting method detailed in Eq. (4).

The fitness scores of each sensor for each time step, $(s_L(t), s_R(t))$, were then summed and normalized with respect to the environment so the overall fitness was in $[0, 1]$ for each environment in the space (Eq. 3).

The fitness scores of each individual environment were then sorted from lowest to highest (worst to best) and a weighted average was performed meaning the overall fitness of the entire environment space also in the range $[0, 1]$ (Eq. 2). Weighting was performed by the geometric sequence $w_i = 1/(2i)$ for $i = \{1, 2, \ldots, \|O\| - 1\}$ where O is subset of the environment space considered. In order to make the weights sum to one, the last weight was set equal to the second to last weight. The other weighting schemes considered were a mean average and simply taking the worst individual environment fitness as the fitness for the whole environment set. Both converged more slowly than method we use.

$$\text{Overall Fitness} = \sum_{i \in ||O||} w_i \ \text{fit}(e_{(i)}) \tag{2}$$

$$\text{fit}(e_i) = \frac{1}{\text{normalize}(e_i)} \sum_{t=T/2}^{T} s_L(t) + s_R(t) \tag{3}$$

$$s_{\{L,R\}}(t) = \begin{cases} 1 & \text{if the sensor is pointing at} \\ & \text{an even cluster at time } t \\ 0 & \text{if the sensor is not pointing at} \\ & \text{an object at time } t \\ -1 & \text{if the sensor is pointing at} \\ & \text{an odd cluster at time } t \end{cases} \tag{4}$$

Evolution was performed using Age Fitness Pareto Optimization (AFPO) with a population size of 50 [16]. AFPO is a multi-objective optimization method using a genome's age and fitness as objectives. Mutations occurred by way changing synapse values in the neural network. If a synapse was chosen for mutation, a new weight was drawn from a random Gaussian value with mean equal to the previous weight and standard deviation equal to the absolute value of the previous weight. This mutation operator is employed because it allows weights near zero to mutate very slightly, while large-magnitude weights can be mutated in a single step over a much broader range. A mutation rate was chosen such that the expected number of synapses mutated each step was one.

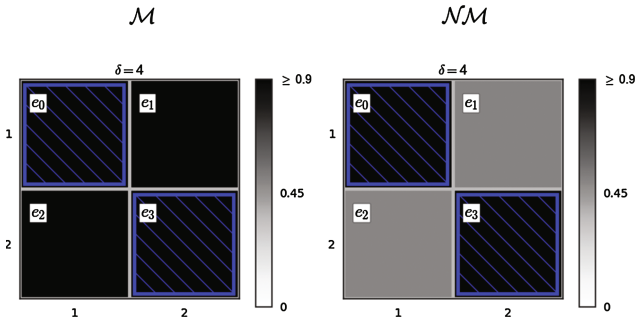
2.5 Experimental Setup

Robots were evolved in a subset, O, of the total environment space, E. O was designated as the training set. When the best robot in the population achieved

a certain fitness threshold for each environment in O, evolution was halted and the best robot was then tested in the remaining unseen environments, $E \setminus O$. We chose a fitness value of 0.9 as the threshold. We performed 30 trials for each experiment.

3 Results

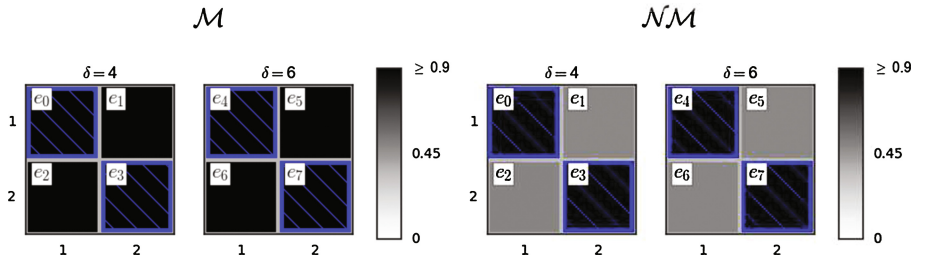
The first environment space explored was E_2, the 2×2 environment space where only c_L and c_R were varied. The training set of the robots was $O_{2,2} = \{e_0, e_3\}$. In E_2, $O_{2,2}$ represents the grand diagonal of the space. Figure 4a shows that $\mathcal{M}$ was able to achieve sufficient fitness in the entirety of E_2 when the robot achieved sufficient fitness in $O_{2,2}$. Figure 4b shows that $\mathcal{NM}$ was not able to achieve sufficient fitness in all environments of E_2. The robot was not above the fitness threshold in any unseen environment for any trial.



(a) Modular robot in E_2 (b) Non-modular robot in E_2

Fig. 4. Average fitness scores for $\mathcal{M}$ (4a) and $\mathcal{NM}$ (4b) robots in E_2 with training set $O_{2,2} = \{e_0, e_3\}$. $O_{2,2}$ is represented by the blue outlines around the environments. (Color figure online)

The second environment space we explored was E_3, the $2 \times 2 \times 2$ environment space where c_L, c_R and δ were free parameters. For this environment space, the training set was $O_{3,4} = \{e_0, e_3, e_4, e_7\}$. This training set represents diagonal sub-spaces of environments for each value of δ. The $\mathcal{M}$ robot was able to be sufficiently fit in the whole space while only being evolved in the environments in $O_{3,4}$ (Fig. 5a). The $\mathcal{NM}$ robot was not able to achieve sufficient fitness in the rest of E_3 after achieving sufficient fitness in $O_{3,4}$ (Fig. 5b). We also evolved the $\mathcal{M}$ robot in E_3 using a different training subset. For this experiment, $O_{3,2} = \{e_0, e_7\}$ which is the grand diagonal of E_3. Results presented in Fig. 6 show the $\mathcal{M}$ was not able to be sufficiently fit.



(a) $\mathcal{M}$ robot in E_3 (b) $\mathcal{NM}$ robot in E_3

Fig. 5. Average fitness scores for $\mathcal{M}$ (a) and $\mathcal{NM}$ (b) robots in E_3 with training set $O_{3,4} = \{e_0, e_3, e_4, e_7\}$. $O_{3,4}$ is represented by the blue outlines around the environments. (Color figure online)

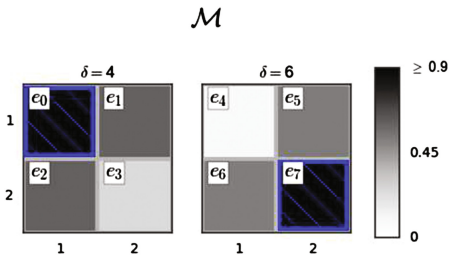
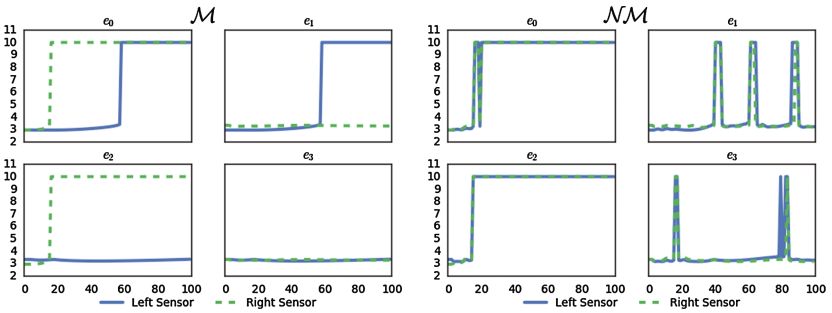


Fig. 6. The $\mathcal{M}$ robot evolved with training set $O_{3,2} = \{e_0, e_7\}$. We see that the robot does not achieve adequate fitness when only evolved in $O_{3,2}$.



(a) Modular Robot (b) Non-modular Robot

Fig. 7. Sensor values of the left and right distance sensors for randomly chosen $\mathcal{M}$ and $\mathcal{NM}$ robots evolved in training set $O_{2,2} = \{e_0, e_3\}$. We see that the modular robot can move one leaf node without affecting the sensor value of the other arm. In contrast the non-modular robot cannot.

4 Discussion and Conclusion

When a robot with ecological modularity is presented with a new environment, it is able to break down that environment along M, the free parameters it is modular with respect to. The robot then can recognize this new environment as a combination of percepts it has seen before and act accordingly. This means the robot only needs to be evolved in a subset of the whole environment space. Specifically, the minimal size of this subset is n^{f-m+1}.

As the ecological modularity in the robot increases, it is able to break down more free parameters in the environment space. This is shown by the fact the $\mathcal{M}$ robot can achieve fitness at or above 0.9 in environments it has not seen before while the $\mathcal{NM}$ robot cannot as seen in Figs. 4 and 5.

We claim that in both the E_2 and E_3 environment spaces the $\mathcal{M}$ robot has $m = 2$ while the $\mathcal{NM}$ robot has $m = 1$. This is because the robot can break down its environment because it is able to move its sensors independently (Fig. 7a). The $\mathcal{NM}$ robot cannot break down its environment into left and right percepts because it is morphologically and neurologically non-modular (Fig. 7b). When it senses a new percept on the right it fundamentally changes how it views its environment even if the percept on the left remains constant. Therefore, in the E_2 environment space $m = 2$ for $\mathcal{M}$ and $m = 1$ for $\mathcal{NM}$.

Neither the $\mathcal{M}$ nor $\mathcal{NM}$ robots are modular with respect to δ. This is shown by the fact that they cannot be simply trained along the diagonal of E_3 to be sufficiently fit in the whole space (Figs. 5 and 6). Therefore, in the E_3 environment space $m = 2$ for $\mathcal{M}$ and $m = 1$ for $\mathcal{NM}$. We hypothesize that if a robot was able to categorize the clusters independently of distance then, in E_3, the robot would have $m = 3$ and only $n^{3-3+1} = n$ environments would be necessary.

In this paper we introduced the concept of ecological modularity and showed that robots which are ecological modular can be sufficiently fit in an entire environment space even though they are only evolved in a subset of its environments. Robots that are not morphologically modular cannot move without changing their entire perception of their environment and thus cannot break down their environment into familiar percepts. Similarly, ecologically non-modular robots cannot view the varying environments in terms of unfamiliar combination of familiar percepts because they cannot sense their world in a manner which breaks down the environment into individual percepts.

In future work we would like to investigate whether ecological modularity can be discovered by evolution instead of from human construction of these robots.

Acknowledgments. We like to acknowledge financial support from NSF awards PECASE-0953837 and INSPIRE-1344227 as well as the Army Research Office contract W911NF-16-1-0304. We also acknowledge computation provided by the Vermont Advanced Computing Core.

References

1. Bongard, J., Bernatskiy, A., Livingston, K., Livingston, N., Long, J., Smith, M.: Evolving robot morphology facilitates the evolution of neural modularity and evolvability. In: Proceedings of the 2015 Genetic and Evolutionary Computation Conference, p. 129136. ACM, Madrid (2015)
2. Bongard, J.: Morphological change in machines accelerates the evolution of robust behavior. Proc. Nat. Acad. Sci. **108**(4), 1234–1239 (2011)
3. Bongard, J.C.: Spontaneous evolution of structural modularity in robot neural network controllers. In: Proceedings of the 2011 Genetic and Evolutionary Computation Conference, pp. 251–258. ACM, Dublin (2011)
4. Cappelle, C.K., Bernatskiy, A., Livingston, K., Livingston, N., Bongard, J.: Morphological modularity can enable the evolution of robot behavior to scale linearly with the number of environmental features. Front. Rob. AI **3**, 59 (2016)
5. Clune, J., Mouret, J.B., Lipson, H.: The evolutionary origins of modularity. Proc. R. Soc. B Biol. Sci. **280**(1755), 20122863 (2013)
6. Ellefsen, K.O., Mouret, J.B., Clune, J.: Neural modularity helps organisms evolve to learn new skills without forgetting old skills. PLoS Comput. Biol. **11**(4), e1004128 (2015)
7. Espinosa-Soto, C., Wagner, A.: Specialization can drive the evolution of modularity. PLoS Comput. Biol. **6**(3), e1000719 (2010)
8. French, R.M.: Catastrophic forgetting in connectionist networks. Trends Cogn. Sci. **3**(4), 128–135 (1999)
9. Gruau, F.: Automatic definition of modular neural networks. Adapt. Behav. **3**, 151–183 (1994)
10. Jakobi, N., Husbands, P., Harvey, I.: Noise and the reality gap: the use of simulation in evolutionary robotics. In: Morán, F., Moreno, A., Merelo, J.J., Chacón, P. (eds.) ECAL 1995. LNCS, vol. 929, pp. 704–720. Springer, Heidelberg (1995). doi:10.1007/3-540-59496-5_337
11. Kashtan, N., Alon, U.: Spontaneous evolution of modularity and network motifs. PNAS **102**(39), 13773 (2005)
12. Lehman, J., Risi, S., D'Ambrosio, D., Stanley, K.O.: Encouraging reactivity to create robust machines. Adapt. Behav. **21**, 484–500 (2013)
13. Lipson, H., Pollack, J.B., Suh, N.P., Wainwright, P.: On the origin of modular variation. Evolution **56**(8), 1549–1556 (2002)
14. Matarić, M., Cliff, D.: Challenges in evolving controllers for physical robots. Rob. Auton. Syst. **19**(1), 67–83 (1996)
15. Newman, M.E., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E **69**(2), 026113 (2004)
16. Schmidt, M., Lipson, H.: Age-fitness pareto optimization. In: Riolo, R., McConaghy, T., Vladislavleva, E. (eds.) Genetic Programming Theory and Practice VIII. Genetic and Evolutionary Computation, vol. 8, pp. 129–146. Springer, New York (2011). doi:10.1007/978-1-4419-7747-2_8
17. Wagner, G., Pavlicev, M., Cheverud, J.: The road to modularity. Nat. Rev. Genetics **8**(12), 921–931 (2007)
18. Yamashita, Y., Tani, J.: Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment. PLoS Comput. Biol. **4**(11), e1000220 (2008)