Spontaneous Evolution of Structural Modularity in Robot Neural Network Controllers

Artificial Life/Robotics/Evolvable Hardware

Josh C. Bongard
Department of Computer Science
University of Vermont
josh.bongard@uvm.edu

ABSTRACT

In order to evolve large robot controllers for increasingly complex tasks, fully connected neural networks are not feasible. However, manually designing sparse neural connectivity is not intuitive, and thus should be placed under evolutionary control. Here I show how spontaneous structural modularity can arise in the connectivity of evolved robot controllers if the controllers are boolean networks, and are selected to converge on point attractors that correspond to successful robot behaviors.

Categories and Subject Descriptors

I.2.9 [**Computing Methodologies**]: Artificial Intelligence—*Robotics*

General Terms

Experimentation, Algorithms, Reliability

Keywords

Evolutionary Robotics, Modularity, Evolutionary Algorithms

1. INTRODUCTION

In the field of evolutionary robotics [11, 19], evolutionary algorithms have been employed to optimize neural network controllers such that the robots succeed at some desired tasks. These tasks are typically simple sensor-motor coordination tasks, such as legged locomotion (e.g. [10]), wall following (e.g. [17]), visually-guided navigation [11], social coordination (e.g. [28]), or object manipulation [16]. Given the simplicity of the tasks, the robots are usually similarly simple: they are composed of at most a dozen sensors and motors attached to mechanical degrees of freedom (DOFs), and a corresponding number of internal neurons.

However in order to evolve robots capable of more complex tasks, it will be necessary to construct robots with hundreds or thousands of sensors and motors, and as many internal neurons. This requires a relaxation in the assumption that the neural controller is fully connected, as the number of connections scales quadratically with the number of nodes in a fully connected network. This raises the question then of what the structure of the connectivity should then be for a given robot and task. One desired form of connectivity is structural modularity, in which certain groups of neurons are densely connected, and there is little or no connectivity between groups.

Previous work has investigated allowing for neural networks to evolve modularity if the task calls for it. In [18] however, the number and type of modules, as well as the connectivity pattern was predetermined.

Gruau [10] employed an indirect genotype to phenotype mapping that allowed for the construction of neural modules. However, this method requires an evolutionary operator for explicitly creating modules. In other work [5] an evolutionary operator could duplicate existing modules. The work presented here does not require such high-level operators; modules emerge (or dissipate) as connectivity density gradually increases (or decreases) over evolutionary time. Using a more recent indirect encoding method [23], structural modularity—even though explicitly selected for—failed to evolve on all but the simplest of problems [6].

In domains outside of evolutionary robotics it has been shown that modularity can be achieved by allowing evolution to hierarchically compose independent genetic material [27]. [25, 26] have argued that modularity evolves as a result of directional selection acting on one part of the network, and stabilizing selection acting on another part. This latter dynamic has been realized in a number of evolutionary simulations [15, 12, 24]. In particular, Espinosa-Soto and Wagner [8] showed how simple models of genetic regulatory networks (GRNs) could evolve to become more modular if:

1. the GRNs were modeled as random boolean networks with n nodes;
2. a biased mutation was used to keep the average incoming connections to any node near to 2 or 3;
3. the GRNs were evolved to settle into point attractors from a set of initial conditions (eg. `1010...1010` and all strings with a Hamming distance of two or less);
4. for a uniform initial pattern across all n nodes (`1010 ...1010`), the first $k = n/2$ neurons should settle into one pattern (`1010...10`) and the second k neurons should settle into the same pattern (`1010...10`); and

5. for a different set of initial conditions (`1010...0101`), the first k neurons should settle into the same pattern as before (`1010...10`) but the second k neurons should settle into a new pattern (`0101...01`).

This process in effect selects for the two groups of k neurons to settle into different attractors. The first group experiences stabilizing selection such that the two sets of initial conditions become the basin of attraction for the same attractor (`1010...1010,1010...0101` $\rightarrow$ `1010...10`). The second group experiences directional selection in that it must evolve to converge on two different attractors given two sets of initial conditions (`1010...1010` $\rightarrow$ `1010...10`, `1010...0101` $\rightarrow$ `0101...01`).

A major assumption in [8] though is that the two groups of k nodes are chosen *a priori*. In a large neural network robot controller it is not obvious how to group the neurons, nor how to exert directional or stabilizing selection on each group to achieve modularity.

In this work it is shown how this assumption can be relaxed by modifying Espinosa-Soto and Wagner's method: Instead of a GRN, the random boolean network represents a robot neural network controller, and the controllers are evolved such that the robot performs a given task. The results show that, under the right conditions, the controllers evolve increasing structural modularity in which certain neuronal groups evolve dense connectivity, there is sparse connectivity between the groups, and the user does not need to specify the groups beforehand nor formulate explicit module-creation evolutionary operators.

The next section describes the general methodology; section 3 reports results from a number of experiments, in some of which structural modularity evolved; and section 4 provides some discussion of how this approach could be scaled up to more complex robots and tasks.

2. METHODS

This section describes the robot, its task, the controllers, and the evolutionary algorithm.

2.1 The Robot Body

The robot employed here is a two-dimensional seven-element, six degree-of-freedom arm and hand system. Various poses that the arm can take are shown in Figs. 1 and 2. The arm is composed of three segments; each of the two fingers are composed of two segments each. Each joint connecting neighboring units together can rotate through $[-45^o, +45^o]$.

The robot may be evaluated in any of four environments, where the goal is the same: to 'grasp' the circular object found in that environment by manoeuvering the finger tips to meet its circumference. When placed any of the four environments, the robot begins in one of fifteen possible initial conditions (ICs), giving 60 possible training environments.

The robot is evaluated in a kinematic simulator: at each time step of simulation the desired angle is computed at each DOF, and the joint's angle is set to this angle at the next time step. There is thus no concept of mass, motor strength or momentum in the simulation.

2.2 The Controllers

The robot is controlled by artificial neural networks which are encoded as synchronous, deterministic boolean networks, initially introduced by Kauffman [13, 14]. A boolean network can be viewed as a graph in which the nodes take

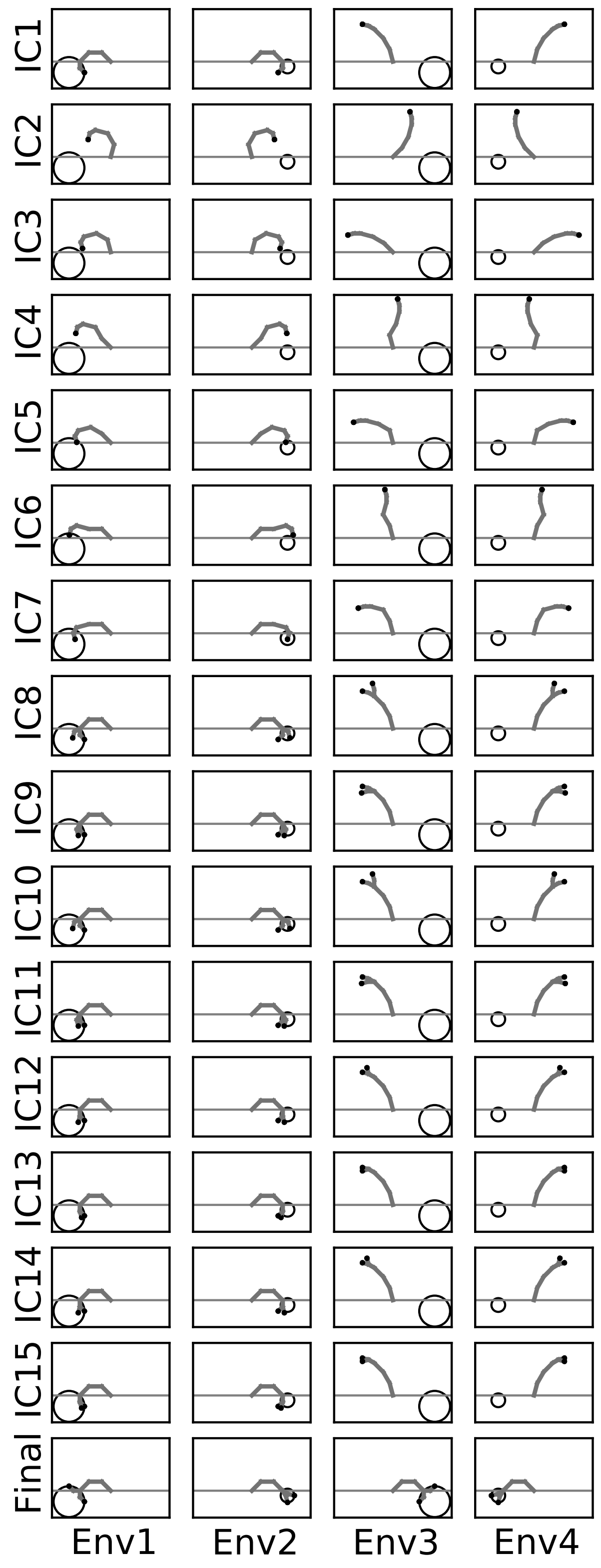

Figure 1: The robot can be exposed to up to four environments (Env*; columns). In each environment, the robot starts in one of fifteen possible initial conditions (IC*; rows). For each environment there is only one configuration that will allow the robot to grasp the target object (Final; bottom row).

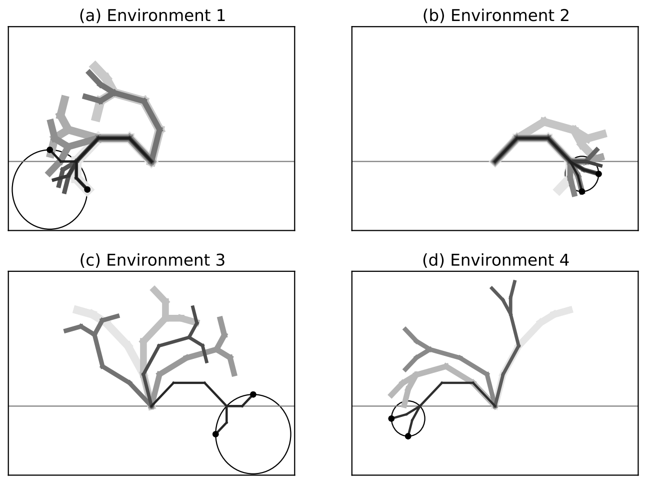

Figure 2: Results from a typical evolved controller. In environment 1 (a) the robot begins with initial condition zero (Fig. 1, top left panel), shown with a thick pale tracing. The robot then passes through eight configurations until it touches the target object. Each subsequent pose is shown with a darker, thinner tracing. Circles indicate the fingertips in contact with the object. In environments 2, 3 and 4 (b,c,d) the robot transitions through 6, 6 and 5 poses, respectively, before reaching the successful configuration. In some of these behaviors the arm reverses direction before settling.

binary values, and a connection between them is likewise binary: it is either excitatory or inhibitory. In this work we assume that a set of number of $k = 2$ nodes are assigned to each of the seven mechanical degrees of freedom (DOFs), yielding a boolean network with 14 nodes

$$\{x_1^{(1)}, x_1^{(2)}, x_2^{(1)}, x_2^{(2)}, \dots x_7^{(1)}, x_7^{(2)}\} \quad (1)$$

where $x_i^{(j)}$ is the binary value of the jth node assigned to the ith DOF.

During the first time step of the evaluation of a controller, the nodes are treated as sensor neurons. In the second and subsequent time steps, the nodes are treated as motor neurons. In this way the boolean network remains an autonomous system in the sense that there is no external influence on the network's behavior over time. This integration of boolean networks with robots differs from that proposed in [20], in which the robot's sensors are connected to a subset of the boolean nodes, and the motors are connected to another, non-overlapping subset of nodes.

During the first time step of evaluation, the nodes provide sensory information about the robot's current environment. The first of $k = 2$ nodes at each DOF indicates where the object is (left or right), and the second node indicates the size of the object (small or large). Thus each DOF has all required information about the environment in which the robot finds itself.

More specifically, if the robot is placed in environment 1 or 4 in which the object is on the robot's left, the first of the $k = 2$ nodes at each degree of freedom are set to -1 ($x_*^{(1)} = -1$). If the robot is placed in environment 2 or 3, in which the object is to the robot's right, $x_*^{(1)} = +1$. (In this work the nodes take on either +1 or -1, compared to 0 or 1 as is sometimes the case in random boolean networks.) Similarly, if the robot is placed in environment 1 or 3 in which the object is large, $x_*^{(2)} = -1$. If the robot is placed in environment 2 or 4 in which the object is small, $x_*^{(2)} = +1$.

It is assumed that the sensors may be noisy. So, once the nodes have been set based on the sensory stimuli from the environment, they may be perturbed by flipping one bit. This leads to 15 possible initial conditions: either the sensor readings are all correct (no bits are flipped), or one of the 14 node values is flipped.

Once the robot is supplied with information about its environment, the robot moves as follows. At each time step, the two binary nodes assigned to each DOF are translated into a desired angle that the joint should rotate to at the next time step. Given the two binary nodes, there are four possible desired angles:

$$x_k^{(1)} = -1, \quad x_k^{(2)} = -1 \quad \rightarrow -45^o \quad (2)$$

$$x_k^{(1)} = -1, \quad x_k^{(2)} = +1 \quad \rightarrow -15^o \quad (3)$$

$$x_k^{(1)} = +1, \quad x_k^{(2)} = -1 \quad \rightarrow +15^o \quad (4)$$

$$x_k^{(1)} = +1, \quad x_k^{(2)} = +1 \quad \rightarrow +45^o. \quad (5)$$

In this way, the binary boolean network can specify discrete robot poses. For seven DOFs and $k = 2$ nodes per DOF, there are $4^7 = 16384$ possible poses. Although the robot's motion is thus discrete, the robot's motion could be made more continuous by increasing k.

At the first time step, the robot is moved into the pose dictated by the initial, (possibly perturbed) node values. For each environment this means that there are 15 possible initial poses; all 60 such poses are shown in Fig. 1.

The controller is updated at each time step in the standard fashion for boolean networks:

$$x_{m,t+1}^{(n)} = \begin{cases} -1 & : \sum_{i=1}^{7}\sum_{j=1}^{2} w_{im}^{(jn)} x_{i,t}^{(j)} < 0 \\ +1 & : \text{otherwise}, \end{cases} \quad (6)$$

where $x_{m,t+1}^{(n)}$ is the new value of the nth node assigned to the mth DOF at time step $t+1$, and $w_{im}^{(jn)}$ is the connection strength (-1 or +1) from node $x_i^{(j)}$ to $x_m^{(n)}$. If no connection exists from one node to another, $w_{im}^{(jn)} = 0$.

Once the new values of all nodes have been computed, the robot moves to the new pose as dictated by the new node values. This in effect 'erases' the sensory data initially supplied to the robot. However, the robot must perform a successful sequence of poses such that it comes to rest with its fingertips touching the circumference of the object. This kind of behavior—in which the robot is given an initial snapshot of the world which it loses as soon as it starts moving—has been explored previously in [22].

Fig. 2 shows the behavior of a typical evolved controller. In environment 1 (Fig. 2a) the robot begins flexed completely to the left (thick pale lines) and then transitions through several poses (indicated by increasingly thin, dark lines) until it comes to rest in contact with the object.

As the controller is an autonomous boolean network, the network may converge to a stable state. The pattern of activation across the nodes may reach a fixed point, after which the robot remains stationary as the values of the nodes no longer change. Or, it may converge on a cyclic attractor in which case the arm exhibits rhythmic motion. In order to minimize computational effort, each controller was updated until it reached a steady state, or a maximum of 12 updates were completed.

2.3 The Evolutionary Algorithm

Each evolutionary run reported here was initialized with a population of 10 random boolean networks of $n = 14$. Following [8], $2n = 28$ excitatory and inhibitory connections were added to each network. Each network was used to control the robot, in one or more of the 60 possible environmental conditions shown in Fig. 1. The fitness of a controller is set to

$$f = (\sum_{i=1}^{15}\sum_{j=4}^{2}(1 - \frac{D_{\mathrm{L}} + D_{\mathrm{R}}}{2D_{\mathrm{max}}}))/14 \quad (7)$$

where D_{L} and D_{R} is the distance from the left and right fingertip to the circumference of the (penetrable) object when the controller is exposed to the ith initial condition in the jth environment, respectively, when the boolean network settles into an attractor or 12 updates have elapsed. D_{max} is the maximum distance a fingertip can be from the circumference of the object. This yields $f = 0$ if the fingertips are furthest from the objects in all of the conditions in which the controller is tested, and $f = 1$ if the fingertips contact the object's circumference in all of the conditions in which the controller is tested.

Shaping [7, 21, 3, 4] is employed here, in which the controllers are initially evolved in only one environmental condition: for each controller the robot is placed in environment 1, and the network nodes are initialized with the appropriate sensor readings without perturbation (top-left panel in Fig. 1). The controllers are evolved until one is found that successfully brings the fingertips into contact with the object (bottom-left panel in Fig. 1). Once a controller evolves that succeeds in the first environmental condition, the current controllers are re-evaluated against two conditions. Evolution continues until a controller is found that succeeds in both conditions, after which a third condition is added. This process continues until a controller is found that succeeds in all 60 conditions, or two hours of CPU time elapse.

Each generation is executed as follows. After all 10 controllers have been evaluated, elitism is employed: the most fit controller is copied into the next generation. Controllers are then selected using fitness-proportionate selection, copied, mutated, and placed in the population of the next generation until the remaining 9 slots are filled. Following [8], each controller is mutated as follows. Each node undergoes change with a probability of 5%. For each node u targeted for change, the probability that a random incoming connection will be removed is computed as

$$p(u) = \frac{4r_u}{4r_u + (n - r_u)} \quad (8)$$

where r_u represents the number of regulators (following the terminology in [8]), or incoming connections, to node u. With $q(u) = 1 - p(u)$, a random incoming connection is added instead. This biased mutation operator ensures that nodes maintain on average 2 or 3 incoming connections. This bias was justified in [8] by the observation that this sparse connectivity is observed in many species.

3. RESULTS

Six sets of 500 evolutionary runs each were performed. In the first three sets, a shaping schedule was used that added stable attractors to the evolved controllers, and then gradually widened the basins of attraction for each attractor. This was accomplished by first evolving the controllers to succeed in environment 1, using initial condition 1. Once a successful controller was discovered the controllers were re-evaluated in environment 1 using the first initial condition for that environment (IC1, Env1 in Fig. 1), and in environment 2 using the first initial condition for that environment (IC1, Env2 in Fig. 1). This selected for controllers that, given two different initial conditions, converged on two different attractors that correspond to the two successful poses that result in grasping the object (Final, Env1 and Final, Env2 in Fig. 1). Once a successful controller was discovered, the third environment was added (IC1, Env3 in Fig. 1), and so on.

Once a controller was found that succeeded in all four environments, the controllers were re-evaluated in the four environment, plus the second initial condition for environment 1 (IC2, Env1 in Fig. 1). This selected for controllers with an expanded basin of attraction for the first attractor: two sets of initial conditions should converge to the successful pose for the environment 1. This process continues until controllers succeed in all four environments from two initial conditions. Then the third initial condition is added, and so on, until controllers are discovered that succeed in all 60 conditions.

In the second three sets of runs, a shaping schedule was used that evolved controllers to converge on one attractor, then gradually expanded the basin of attraction for that attractor before selecting for the ability to converge on a second attractor. This was accomplished by first evolving controllers to succeed starting from the first initial condition in environment 1, as before. Once a successful controller is found the controllers are re-evaluated twice in environment 1: once using the first initial condition and again using the second initial condition (IC2, Env1 in Fig. 1). Once a successful controller is re-discovered for these two conditions, controllers are re-evaluated using the first three initial conditions for environment 1, and so on.

Once a controller succeeds in all fifteen initial conditions, the controllers are re-evaluated in 16 conditions: the first 15 initial conditions for environment 1 and the first initial condition for environment 2. Evolution again continues along this shaping schedule until all 60 conditions can be accomplished by a controller.

In the first and fourth sets of runs, additional constraints were placed on the fitness of the controllers. For each condition in which a controller is evaluated, it must converge on a point attractor in order to obtain fitness for that condition:

$$f = (\sum_{i=1}^{15}\sum_{j=1}^{4}\alpha_{ij}(1 - \frac{D_{\mathrm{L}} + D_{\mathrm{R}}}{2D_{\mathrm{max}}}))/14 \quad (9)$$

where $\alpha_{ij} = 1$ if the controller converges on a point attractor starting from the ith initial condition in environment j, and $\alpha_{ij} = 0$ if it fails to.

In sets 2 and 5, controllers were again constrained to settle into an attractor in order to obtain fitness, but they could settle into a point or any cyclic attractor. As each controller was allocated a maximum of 12 time steps, any attractor

Table 1: Summary of evolutionary runs.

Set	Shaping schedule	Fitness constraints	Runs
1	Add attractors, then widen their basins	Only allow point attractors	500
2	Add attractors, then widen their basins	Allow point or cyclic attractors	500
3	Add attractors, then widen their basins	Allow any behavior	500
4	Widen attractor basin, then add new attractors	Only allow point attractors	500
5	Add attractors, then widen their basins	Allow point or cyclic attractors	500
6	Add attractors, then widen their basins	Allow any behavior	500

with a period from 1 to 12 was allowable:

$$f \quad = \quad (\sum_{i=1}^{15}\sum_{j=1}^{4}\beta_{ij}(1-\frac{D_{\mathrm{L}}+D_{\mathrm{R}}}{2D_{\max}}))/14 \qquad (10)$$

where $\beta_{ij}=1$ if the controller converges to a point or cyclic attractor starting from the ith initial condition in environment j, and $\beta_{ij}=0$ if it fails to.

In sets 3 and 6, controllers were not constrained to be stable. If they settled into a point or cyclic attractor, fitness was computed from the pose resulting from the first binary pattern to repeat during behavior. If they did not, fitness was computed from the pose achieved at the twelfth time step (Eqn. 7). A summary of the experimental runs is given in Table 1.

Fig. 2 depicts four typical evolved behaviors from one evolutionary run from set 1. Fig. 2a shows the behavior that results from the controller that evolved to succeed in the first environment from the first initial condition. This controller is shown in Fig. 3a. After further evolution a controller emerged that succeeded in environments 1 and 2 from the first initial conditions for those environments. It is shown in Fig. 3b, and its resulting behavior in environment 2 is shown in Fig. 2b.

Fig. 3c shows the first controller from that run to succeed in the first three environments; Fig. 2c shows its behavior in environment 3. Similarly, Fig. 3d shows the first controller from that run to succeed all four environments; Fig. 2c shows its behavior in environment 4.

As can be seen in Fig. 3, the density of connectivity between nodes associated with the arm (the six white circles) and between those nodes associated with the hand (the eight gray circles) gradually increases from the first to the fourth environment. Also, the connectivity between these two groups gradually decreases. This suggests that structural modularity is evolving: control within the arm and control within the hand is gradually becoming more integrated, but influence of the arm on the hand, and vice versa, is lessening.

This progression can be seen more clearly in Fig. 4. The successful controller from the first condition was extracted from each of the 500 runs from set 1, and the fraction of connections (excitatory or inhibitory compared to no connection) was computed between each pair of nodes across the 500 networks. The result is shown in Fig. 4a, where

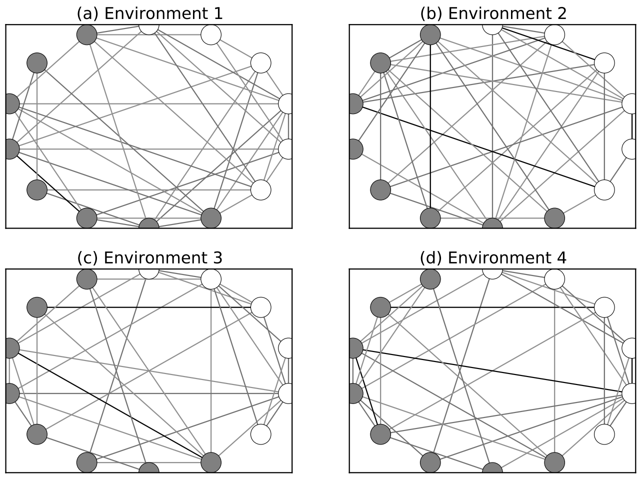

Figure 3: Four evolved controllers. White circles represent arm neurons; gray circles represent hand neurons. Gray lines indicate an excitatory or inhibitory connection between neurons; black lines indicate a connection exists from neuron i to j as well as a connection from neuron j to i. (a): A successful controller after being evolved in environment 1. (b) Its descendent controller which is successful in environments 1 and 2. (c) Its descendent, which is successful in environments 1, 2 and 3. (d) The final controller which succeeds in all four environments; its behavior is depicted in Fig. 2.

lighter lines indicate fewer connections between that node pair, and darker lines indicate more connections.

This computation was repeated on the 500 controllers that succeeded on the first two conditions, and is shown in Fig. 4b. It was again repeated on the 500 controllers that succeeded on the first three conditions (Fig. 4c) and on the 500 controllers that succeeded on the first four conditions (Fig. 4d).

Fig. 5 reports the evolutionary dynamics in the first three sets of runs in which controllers were first evolved to converge to different attractors, and then to widen the basin of attraction for those attractors.

Fig. 5a reports the number of runs that were able to produce successful controllers for the 60 scaffolding stages. The four thin vertical lines within each group indicate the incremental addition of four environments; the thick vertical lines indicate the gradual addition of new initial conditions to each environment. It is clear that when controllers are forced to converge on point attractors (thick black line), the system is highly evolvable: almost all of the 500 runs produce controllers for all 60 conditions before two CPU hours elapse. When either point or cyclic attractors are allowed (medium line), evolvability significantly decreases. When converge to an attractor is not enforced evolvability decreases yet further (thin line).

Fig. 5b reports that, among the successful controllers found at each of the 60 scaffolding stages, how many generations elapsed before that controller was found. The rapid increase over the first four conditions indicates that it is relatively difficult to add point attractors to an already-evolved controller (12K generations for set 1), but it is much more

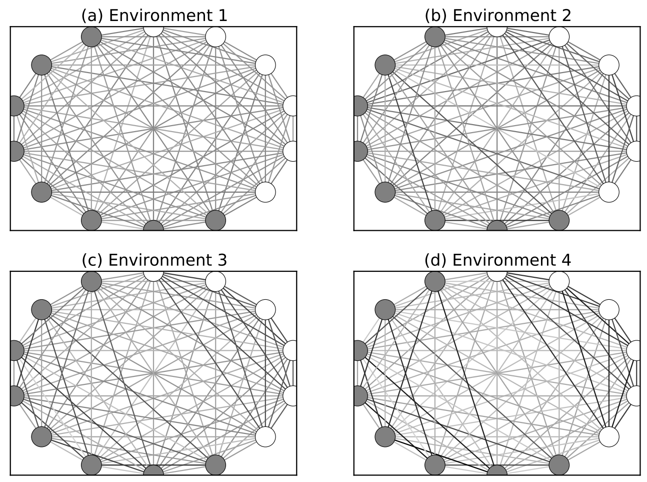

Figure 4: Mean connectivity within evolved controllers. (a) Connectivity averaged across the 500 controllers evolved to succeed in environment 1, initial condition 0 (Fig. 1, top-left panel). (b) Connectivity averaged across the 500 descendent controllers evolved to succeed in environments 1 and 2 using initial conditions 0. (c) Connectivity averaged across the 500 descendent controllers successful in environments 1, 2 and 3. (d) Connectivity within the 500 controllers successful in all four environments.

difficult if any attractors are allowed (>20K), and extremely difficult if the network does not need to converge on an attractor at all (>40K).

For set 1 however, it is then relatively easy to widen the four basins of attraction: only a few more 100 generations elapse, on average, before controllers are evolved that succeed in all 60 conditions (flat black line). For set 2, it is more difficult to widen the basins of attraction (rising middle line). For set 3, some runs make rapid progress (flat thin line), but then become mired on local optima and make no further progress over the remainder of the allowed time.

Fig. 5c reports the structural modularity found within the successful controllers. It is common in the literature (e.g. [9, 8]) to define structural modularity as the ratio between connectivity density within clusters compared to the connectivity density between clusters. Thus, here, the structural modularity of a network is computed as

$$m = \frac{c_{a,a} + c_{h,h}}{c_{a,h} + c_{h,a}} \tag{11}$$

$$c_{a,a} = (\sum_{i=1}^{3}\sum_{j=1}^{2}\sum_{m=1}^{3}\sum_{n=1}^{2} |w_{im}^{(jn)}|)/36 \tag{12}$$

$$c_{h,h} = (\sum_{i=4}^{7}\sum_{j=1}^{2}\sum_{m=4}^{7}\sum_{n=1}^{2} |w_{im}^{(jn)}|)/64 \tag{13}$$

$$c_{a,h} = (\sum_{i=1}^{3}\sum_{j=1}^{2}\sum_{m=4}^{7}\sum_{n=1}^{2} |w_{im}^{(jn)}|)/48 \tag{14}$$

$$c_{h,a} = (\sum_{i=4}^{7}\sum_{j=1}^{2}\sum_{m=1}^{3}\sum_{n=1}^{2} |w_{im}^{(jn)}|)/48 \tag{15}$$

where $c_{a,a}$ is the fraction of connections that exist between

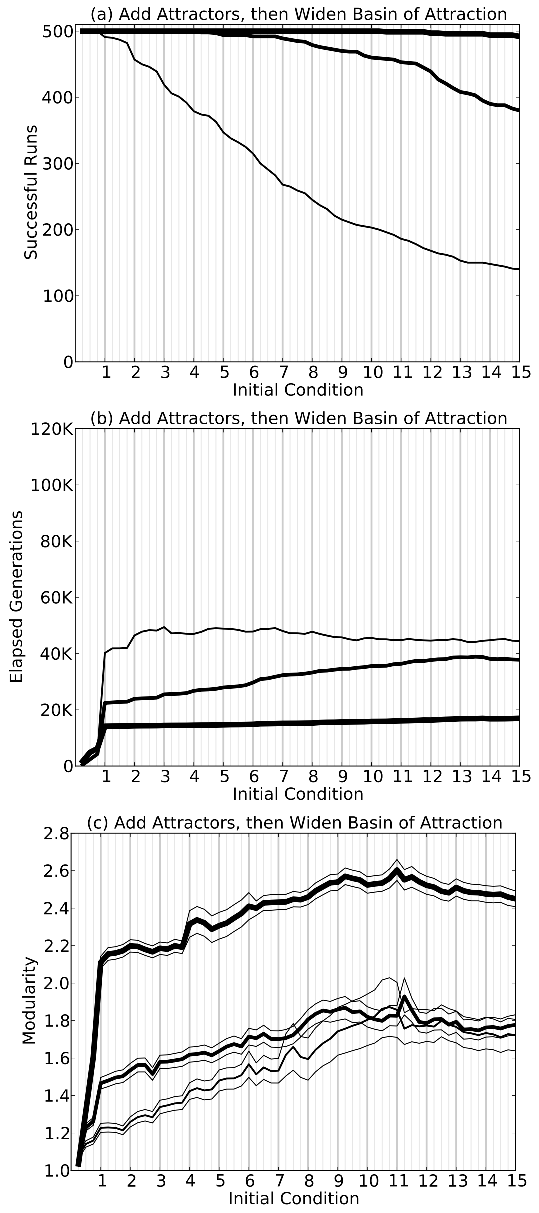

Figure 5: Results from evolving controllers to succeed in all four environments first, and then expanding their ability to handle different initial conditions. Thick vertical line=new initial condition added; thin vertical line=new environment using that initial condition added. (a) Number of runs (out of 500) that produced successful controllers for each evolutionary stage. Thick line=controllers evolved to converge on a point attractor corresponding to the correct pose; medium line=controllers evolved to converge on any attractor that corresponded to the correct pose; thin line=controllers evolved to produce the correct pose after 12 time steps. (b) Mean generations required to discover a successful controller. (c) Mean modularity of successful controllers at that stage. Thin surrounding lines denote one unit of standard error of the mean.

each pair of nodes in the arm, $c_{h,h}$ is similarly the connectivity density with the hand, $c_{a,h}$ is the connectivity density between each node in the arm to each node in the hand, and $c_{h,a}$ is the connectivity density between each node in the hand to each node in the arm. Thus $m >> 1$ indicates the presence of structural modularity: there is significantly greater connectivity density with the arm and within the hand than there is between the two body parts.

As can be seen in Fig. 5c, modularity increases significantly in all three sets as the controllers evolve to succeed in all four environments. However, modularity increases significantly more when point attractors are selected for. In all three sets, modularity then increases slightly as the basins of attraction (or convergence to the correct behavior for set 3) are widened. For set 1, modularity rises such that by the end of the runs the average connectivity within body parts is nearly 2.5 more dense that it is between body parts.

When networks are first evolved for large basins of attraction around a single attractor, after which more attractors are added (sets 4,5 and 6), the evolutionary dynamics are quite different. Fig. 6a shows that for all three sets, the runs are less evolvable: fewer runs (compared to Fig. 5a) complete before the maximum allowed time elapses. Thin vertical lines now represent the addition of new initial conditions to an environment in which the controllers have already succeeded, and thick vertical lines indicate the addition of new environments to the controllers' training set.

Fig. 6b shows the reason for this decrease in evolvability. Controllers rapidly evolve to succeed for all 15 initial conditions of environment 1, but then a significant period of time is required to further evolve them to succeed for just one initial condition in environment 2. This pattern is repeated at the transition from two environments to three, and so on.

Fig. 6c shows that modularity does not emerge during the expansion of the first basin of attraction, but does rise during the transition from successful behavior in one environment to successful behavior in two environments. There is again no change in modularity until controllers are evolved to succeed in three environments. After succeeded in all four environments, in set 1 there is a significant increase in modularity as the basin of attraction for this new, fourth attractor is widened. As in the first scaffolding schedule, significantly more modularity evolves when point attractors are selected for, compared to the two less restrictive fitness functions.

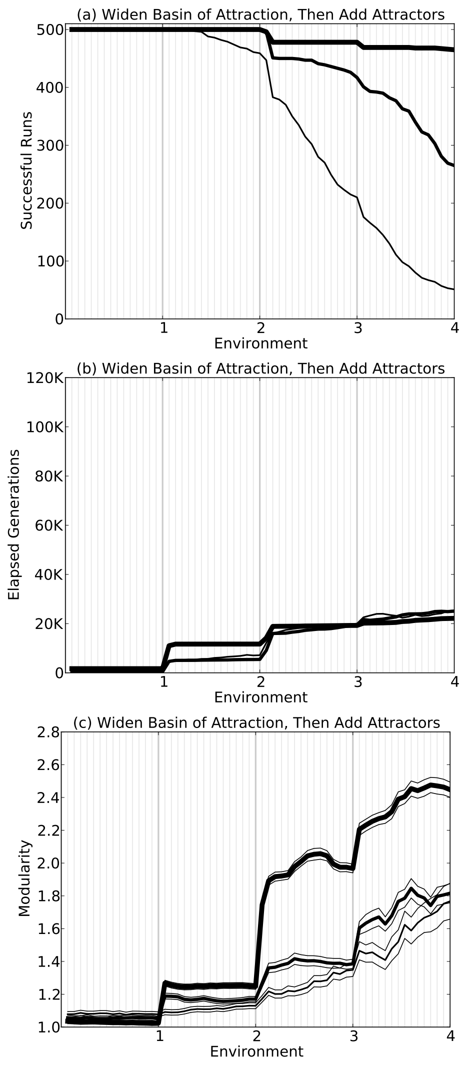

Figure 6: Results from evolving controllers to succeed in each of the 15 initial conditions for environment 1 first, and evolving them further to succeed in the other three environments. Thin vertical line=new initial condition added; thick vertical line=new environment using the first initial condition added.

4. ANALYSIS AND CONCLUSIONS

This work demonstrates that structural modularity may arise in robot controllers under the right conditions. This approach does not require building modules in *a priori* for evolution to make use of [18], nor does it require explicitly bringing selection pressure to bear differently on different parts of the network [8]. Also, it does not require that explicit evolutionary operators be added that create modules, as is the case in indirect or recursive genotype-to-phenotype mappings [10] in which duplication of phenotypic structures is common. Figs. 5 and 6 make clear that particular kind of dynamics—in this, the convergence to point attractors—greatly influence the amount of modularity that arises. This suggests that it may be easier to evolve modularity in dynamical systems, compared to systems that do not have their own intrinsic dynamics (as were employed in [12]), but this conclusion requires further evidence. The explanation for why structural modularity arises in the controllers describe here derive not just from their dynamics, but because of the structure of the tasks. Here, as in [8], different selection pressures come to bear differently on different parts of the network. For pairs of conditions in which the same-sized object is placed on the left or right, the arm should behave differently (rotate to the left or right), while the arm should exhibit the same posture. For pairs of conditions in which differently-sized objects are placed all on the same side of the robot, the arm should behave the same (rotate to that side), but the arm should behave differently. This causes evolution to favor adaptive dissociation between the arm nodes and the hand nodes so that they may act independently under different conditions. Presumably, selecting for larger basins of attraction toward a given attractor further increases modularity because greater connectivity within these groupings is favored: connections between nodes within a group can help correct for sensor noise.

Much work remains to be done to investigate under what conditions structural modularity arises in robot controllers.

First, the system as described here will be re-run for different placements and sizes of objects to determine which conditions favor modularity and which do not, and to verify that the proposed method discovers non-modularity controllers for the former cases and modular controllers for the latter cases. Second, the importance of the biased mutation operator (Eqn. 8) will be investigated: it is possible that this operator may keep networks close to the critical regime [14, 2], and that this regime more easily admits the addition of attractors and the expansion of their basins of attraction [1]. Third, robot controllers cannot remain autonomous dynamical systems: in order to perform complex tasks robots must be able to continuously sense their environment. This will require maintaining the desirable properties of this approach while transitioning to drive dynamical systems. This may require the use of shaping techniques [7, 21, 3, 4] in which robots gradually sense more of their environment, for longer periods of time, than their ancestors did. The motor system of the robots will likewise have to be enhanced. Finally, a better coupling between a binary controller and a robot operating in a continuous environment must be investigated. This may require assigning increasing numbers of groups of binary nodes to sensors and motors, rendering some of the discrete aspects of random boolean networks continuous, or some combination of the two approaches.

5. ACKNOWLEDGMENTS

This work was supported by National Science Foundation Grant CAREER-0953837.

6. REFERENCES

[1] M. Aldana, E. Balleza, S. Kauffman, and O. Resendiz. Robustness and evolvability in genetic regulatory networks. *Journal of theoretical biology*, 245(3):433–448, 2007.

[2] P. Bak and K. Sneppen. Punctuated equilibrium and criticality in a simple model of evolution. *Physical Review Letters*, 71(24):4083–4086, 1993.

[3] J. Bongard. Behavior chaining: incremental behavioral integration for evolutionary robotics. In S. Bullock, J. Noble, R. Watson, and M. A. Bedau, editors, *Artificial Life XI*, pages 64–71. MIT Press, Cambridge, MA, 2008.

[4] J. C. Bongard. Innocent until proven guilty: Reducing robot shaping from polynomial to linear time. *IEEE Transactions on Evolutionary Computation*, 2011. To appear.

[5] R. Calabretta, S. Nolfi, D. Parisi, and G. P. Wagner. Duplication of modules facilitates the evolution of functional specialization. *Artificial Life*, 6(1):69–84, 2000.

[6] J. Clune, B. Beckmann, P. McKinley, and C. Ofria. Investigating whether hyperneat produces modular neural networks. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 635–642. ACM, 2010.

[7] M. Dorigo and M. Colombetti. Robot Shaping: Developing Autonomous Agents Through Learning. *Artificial Intelligence*, 71(2):321–370, 1994.

[8] C. Espinosa-Soto and A. Wagner. Specialization can drive the evolution of modularity. *PLoS Comput Biol*, 6(3):e1000719, 2010.

[9] M. Girvan and M. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12):7821, 2002.

[10] F. Gruau. Automatic definition of modular neural networks. *Adaptive Behaviour*, 3:151–183, 1994.

[11] I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi. Evolutionary robotics: the Sussex approach. *Robotics and Autonomous Systems*, 20(2-4):205–224, 1997.

[12] N. Kashtan and U. Alon. Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences of the United States of America*, 102(39):13773, 2005.

[13] S. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of theoretical biology*, 22(3):437–467, 1969.

[14] S. A. Kauffman. *The Origins of Order*. Oxford University Press, Oxford, UK, 1993.

[15] H. Lipson, J. Pollack, and N. Suh. On the origin of modular variation. *Evolution*, 56(8):1549–1556, 2002.

[16] G. Massera, A. Cangelosi, and S. Nolfi. Evolution of prehension ability in an anthropomorphic neurorobotic arm. *Frontiers in Neurorobotics*, 1(4):1–9, 2007.

[17] O. Miglino, H. Lund, and S. Nolfi. Evolving mobile robots in simulated and real environments. *Artificial Life*, 2:417–434, 1995.

[18] S. Nolfi. Using emergent modularity to develop control systems for mobile robots. *Adaptive Behavior*, 3–4:343–364, 1997.

[19] S. Nolfi and D. Floreano. *Evolutionary Robotics*. MIT Press, Boston, MA, 2000.

[20] A. Roli, M. Manfroni, C. Pinciroli, and M. Birattari. On the design of Boolean network robots. In *Proceedings of EvoApplications 2011*, Lecture Notes in Computer Science. Springer, 2011.

[21] L. Saksida, S. Raymond, and D. Touretzky. Shaping robot behavior using principles from instrumental conditioning. *Robotics and Autonomous Systems*, 22:231–250, 1997.

[22] A. Slocum, D. Downey, and R. Beer. Further experiments in the evolution of minimally cognitive behavior: From perceiving affordances to selective attention. *The Sixth International Conference on the Simulation of Adaptive Behavior*, 6:430–439, 2000.

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

[24] J. Sun and M. Deem. Spontaneous emergence of modularity in a model of evolving individuals. *Physical review letters*, 99(22):228107, 2007.

[25] G. Wagner. Homologues, natural kinds and the evolution of modularity. *Amer. Zool.*, 36:36–43, 1996.

[26] G. Wagner, M. Pavlicev, and J. Cheverud. The road to modularity. *Nature Reviews Genetics*, 8(12):921–931, 2007.

[27] R. Watson. *Compositional Evolution*. MIT Press, 2006.

[28] G. Werner and M. Dyer. Evolution of Communication in Artificial Organisms. *Artificial Life II, SFI Studies in the Sciences of Complexity*, X:659–687, 1991.