The Effects of Environmental Structure on the Evolution of Modularity in a Pattern Classifier

Jessica Lowell and Jordan Pollack

DEMO Lab, Brandeis University, Waltham, MA 02453
jessiehl@brandeis.edu

Abstract

We examine hierarchical modularity - modularity on multiple levels, in which the modules at a lower level of abstraction can serve as nodes in a network at a higher level of abstraction that also has positive modularity - as well as degree of modularity on a single level of abstraction, in evolved neural networks in single-task, parallel-subtask environments, and sequential-subtask environments, using a common benchmark problem. We determine that top-performing networks evolved in the sequential-subtask environment have both more levels of hierarchical modularity, and a higher degree of modularity within levels, than those involved in either the single-task or parallel-subtask environment. In the single-task environment, both single-level and hierarchical modularity tend to rise initially before stagnating and even declining, while in the sequential-subtask environment, both single-level and hierarchical modularity tend to rise throughout the period of evolution.

Introduction

In this paper, we examine the influence of an environment in which a neural network must perform sequential subtasks on the emergence of modularity, both single-level and hierarichical. Both many natural and many engineered systems display modularity, the organization of a system into a hierarchy of independent but interacting subparts (Koza, 1992; Hartwell et al., 1999). Understanding how modularity emerges has become increasingly important as evolutionary systems are used in increasingly complex applications. Neural networks are commonly used in control and machine learning applications, and modularity in large neural networks is beneficial, as it allows a large and unmanageable neural network to be reduced to smaller and more manageable subnetworks (Azam, 2000). We briefly discuss modularity in evolution, followed by an overview of modularity in neural networks.

Modularity in Evolution

A variety of natural and engineered systems tend to be modular, including biological, technical, and organizational (Schilling, 2002). Modular biological systems include networks such as bacterial metabolic networks and biological neural networks, as well as other kinds of biological systems that are assembled from smaller parts, such as tissues, which are assembled from cells. The definition of modularity is not formalized, and is field-dependent and even subfield-depdendent, but generally refers to the degree to which a system is composed of separable, recombinable components. Schilling (2002) reviews elements of modularity that are common to different fields, including technology, biology, American studies, psychology, and mathematics. Bolker (2000) attempted to synthesize definitions of modularity across different abstraction levels and subfields in biology by defining a list of characteristics of modularity, including the ability to delineate modules from their surroundings, greater internal integration within modules than external integration between modules, and module function/performance exceeding the sum of the module's compoonents.

The highest-performing solutions produced by runs of evolutionary algorithms tend to be nonmodular, though there are some exceptions, such as the coevolutionary algorithm of Juillé and Pollack (1996), which found modular solutions to the intertwined spirals problem using genetic programming. The nonmodular solutions tend to be connected in complicated ways, and perform better on the tasks for which they are optimized than modular solutions designed by humans (Thompson, 2012; Vassilev et al., 2000). While this produces good results for simple problems, the lack of modularity makes it difficult for these evolved solutions to solve complex problems (Kashtan and Alon, 2005). Modular architectures are better at certain kinds of modular problems, such as learning piecewise control structures, than nonmodular ones (Jacobs and Jordan, 1993). Designers can address this issue by building the encapsulation of modules into their algorithms, but this sheds no light on how modularity evolves in nature. It also precludes any design benefit that may arise from modularity emerging naturally rather than being hard-coded. For example, there is some evidence (Calabretta et al., 1998) that modules that emerge naturally split tasks differently than hard-coded modules. Juillé (1999) demonstrated a version of this by developing a Mod-

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

ular Inductive Learning system featured automated, rather than hard-coded, decomposition of tasks. Fitness-sharing between simulated symbiont organisms that become a composite, which has some conceptual similarity to modularity in that it combines the functionalities of multiple subsystems, promotes evolution on hierarchical problems (Watson and Pollack, 2000, 2001). Finally, allowing modules to emerge through the iterative process of evolution may allow for high-performing nonmodular candidate solutions to develop modularity over time without compromising their strong performance, resulting in the benefits of both evolutionary algorithms and modularity.

Because of these benefits of understanding and implementing the evolution of modularity, in recent years there have been several studies on the emergence of modularity in both natural and artificial systems. These have taken two main approaches - examining the kinds of environments that cause modularity to evolve, and the kinds of selection factors that cause modularity to evolve. In the former category, Lipson et al. (2002) suggested using variable rather than fixed criteria for evolutionary optimization after finding, in a study of minimal substrate modularization, that modular separation is logarithmically proportional to the environmental variation rate. Further work on environmental variation and modularity in computational evolution studies (Kashtan and Alon, 2005) and natural evolution studies (Kashtan et al., 2007; Parter et al., 2007) found that modularity evolves in response to varying environments in which an individual switches between optimizing for different tasks that are decomposable into common subtasks (modularly varying goals). In the latter category, Clune et al. (2013) proposed that modular networks evolve in response to selection pressure to minimize the number of connections between nodes, representing the energy cost of forming a link in a physical network. A differently-implemented energy cost imposed on the NEAT neuroevolution algorithm, on a problem in which some solutions that evolve are relatively modular and some are not, has been found to increase consistency in the degree of modularity that emerges (Lowell and Pollack, 2014). Calcott (2014) reported the emergence of first-level modularity in a sequential-subtask environment, but did not elaborate on details of the experiments done or the results found.

While (Schilling, 2002) found that nearly all fields examined defined hierarchical nesting as part of the definition of modularity, and (Variano et al., 2004) found that hierarchical modularity in networks improved the robustness of network stability, it has not traditionally been examined in simulated evolution studies of modularity. While there are a few recent exceptions to this - a study by Lowell and Pollack (2016) found that developmental encodings promoted the emergence of hierarchical modularity, and work byMengistu et al. (2016) found that connection cost between nodes in a network promoted its emergence - it is still not well-studied. Furthermore, this existing work on evolving hierarchical modularity focuses on the hierarchy rather than the degree to which specific levels of abstraction are modular, and work on single-level degree of modularity, by necessity, has looked at modularity only on the lowest level of abstraction. This paper addresses the emergence of a hierarchy of modules as well as single-level modularity on each level of abstraction. To clarify our terminology: at the lowest level of abstraction, or level of modularity, in a modular network, the network's nodes form modules of some levels of strength. If some or all of these modules can in turn be treated as nodes in a new network, that is a second level of abstraction, or level of modularity. The strength of the modules within a given level, the calculation for which is discussed below, is the single-level modularity for that level. Hierarchical modularity is defined in this paper as the presence of multiple levels of abstraction with positive modularity, with a more hierarchically modular network having more levels.

Methods

NEAT

There are many algorithms for evolving neural networks. Some of these evolve network topology only, some evolve weights only, and some evolve both. NeuroEvolution of Augmenting Topologies, or NEAT (Stanley and Miikkulainen, 2002), is an example of a neuroevolution algorithm that evolves both. It begins with simple, minimal networks, and gradually generates more and more complex ones through three forms of mutation: adding neurons, adding connections, and modifying connection weights. It temporarily protects innovations that are sufficiently different from existing candidate solutions, in order to give them the opportunity to cultivate niches and prove evolutionarily useful, by using a speciation mechanism, which isolates subsets of the population into different reproductive groups based on topological dissimilarity. Species that perform well grow and persist, perhaps with new species splitting off, while poorly-performing species gradually die off. Another important aspect of NEAT is that it tracks the history of innovations through assigning them persisting numbers, which allows for a working crossover operator by making it possible to determine which subnetworks can be recombined without producing nonviable neural networks. NEAT remains one of the most popular neuroevolution algorithms, and has proven effective in several problem domains, including vehicle collision avoidance, evolving a roving eye for Go, training AI teammates to work together in a video game (Stanley and Miikkulainen, 2002; Stanley et al., 2005), and strategic decision-making (Lowell et al., 2011).

The standard NEAT algorithm does not tend to evolve modularity (Reisinger et al., 2004). Reisinger et al. (2004) created Modular NEAT, a version of NEAT intended to produce modular solutions, by requiring their algorithm to reuse

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

subnetworks in different spatial locations to form complete neural networks, thereby forcing a predisposition toward modularity. HyperNEAT (Stanley et al., 2009), an extension of NEAT that evolves compositional pattern-producing networks for an indirect encoding, also does not tend to produce modular networks on its own Clune et al. (2010), though Verbancsics and Stanley (2011) were able to influence it toward modularity by seeding with a bias to connect components which are spatially near each other, a mechanism that, like the previously-discussed work on connection costs and modularity, relates to the energy costs of connecting nodes, and may play a role in biological modularity. In this study, we use the NEAT4J open source Java implementation of NEAT (Simmerson, 2006).

Probability of mutation, crossover, and other network-related and evolution-related parameters, is set by the user in NEAT. In Table 1 we list key NEAT parameters used in all experiments in this paper.

Parameter	Value
Population size	500
Max number of generations	2000
P(Mutation)	0.75
P(Crossover)	0.25

Table 1: Parameters used in all experiments.

Calculating Modularity

Many studies of modularity use the metric of Q to represent modularity, defined by the approach of (Newman and Girvan, 2004). This method is well-suited for networks in which connections are undirected and binary (i.e. networks where there is either a connection between two nodes or there is not) and the phenomenon being studied is single-level modularity (the modularity at one layer of abstraction). It does not account for different edge weights between nodes, as exist in neural networks, nor for directed networks, as neural networks are. In addition, it is not intended to calculate the number of levels of hierarchical modularity in a network. In order to better understand the modularity of our evolved neural networks, we used the "Louvain method," which was designed to detect hierarchical modularity and maximize community detection while also having a low runtime and not requiring a predefined number of communities, and which has versions for both undirected and directed graphs, to determine Q (Blondel et al., 2008). This algorithm initially assigns each node in the network to its own module. For a weighted graph, it calculates modularity Q using the definition of modularity:

$$Q = \frac{1}{2m} \sum ij \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta\left(c_i, c_j\right) \tag{1}$$

where A_{ij} represents edge weight between nodes i and j, m represents half the sum of the graph's edge weights, δ is a delta function, c_i and c_j are node communities, and k_i and k_j are the sums of the weights of all edges attached to node i and node j respectively.

Once this value is found, for each node, the algorithm calculates the change in modularity achieved by moving that node into the module of each of its neighbors. This calculation of change in Q is done for each module that the node is connected to. The node is then moved into the module that would result in the greatest modularity increase (or left in place if no modularity increase is possible). Once there is no further possible increase, the first result of the algorithm - the modularity at the lowest level of abstraction of the network - is equal to the current Q. These modules can then be used as nodes in a network one level of abstraction higher, with subsequent, hierarchical levels of Q being calculated in the same fashion. For the directed version of the Louvain method, we used Antoine Scherrer's MATLAB implementation of directed Louvain (Scherrer, 2008), which uses a modified definition of modularity developed by Arenas et al. (2007), and consequently modified change equations, for directed networks:

$$Q = \frac{1}{2m} \sum ij \left[A_{ij} - \frac{k_i^{out} k_j^{in}}{2m} \right] \delta\left(c_i, c_j\right) \tag{2}$$

The Retina Problem

We tested the effect of sequential subtasks on modularity was the eight-pixel retina problem, a pattern classification problem developed by Kashtan and Alon (2005) to study the emergence of modularity on a single level of abstraction, and which has been used in other modularity studies (Clune et al., 2010; Verbancsics and Stanley, 2011; Lowell and Pollack, 2014). In this problem, a pattern recognition system such as a neural network must recognize two-pixel by two-pixel patterns in either half of a four-pixel by two-pixel retina, where certain patterns represent objects and the rest do not, and return true or false based on whether an object is present, with performance evaluated by the mean squared error across all possible combinations of inputs. There are differences between which patterns are objects in each half of the retina, as illustrated in Fig.1. These differences make for a more complex task, as the classifier must discover both functions and benefits from the ability to separate into different functional substructures, and advanced neuroevolution algorithms such as HyperNEAT have struggled with it (Clune et al., 2010). We compare three different versions of the retina problem. In the first version, there is only a single task; it is not modularly decomposable, as an object need only exist on one side of the retina for the correct answer to be "true." The second version of the problem requires an object to be present on both sides of the retina

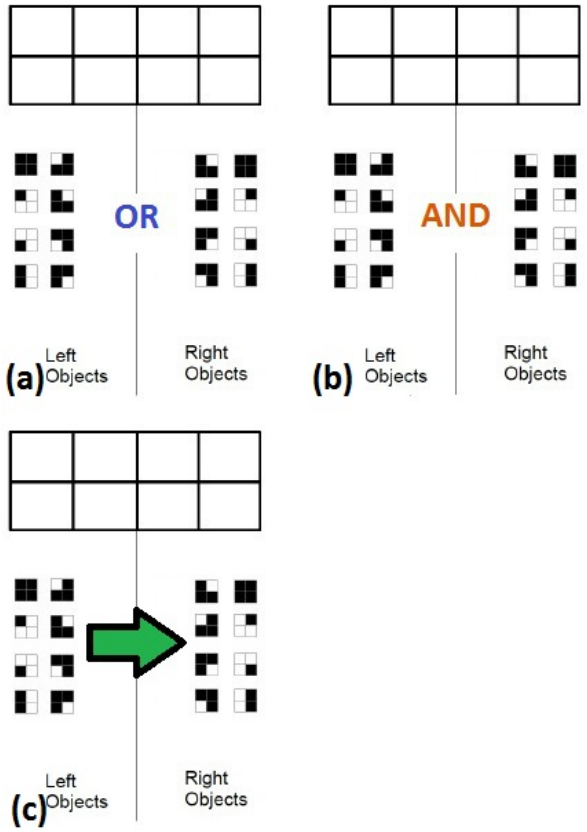

Figure 1: An illustration of patterns used in the retina problem, adapted from (Kashtan and Alon, 2005). a) The single-task version, in which an object need only exist on one side of the retina for the correct answer to be "true." b) The parallel-subtask version, in which the classifier must combine appropriate functions for the left and right sides. c) The sequential-subtask version, in which the classifer must assess the left and right sides in sequence.

for the correct answer to be "true," meaning that it can be divided into two parallel subtasks. The third version requires the pattern recognition system to perform the subtasks in sequence instead of in parallel. The pattern recognition system first evaluates the left half of the retina. The error from the evaluation of the left half of the retina then propagates into the inputs of the neural network for the evaluation of the right half of the retina - for example, if input I into the neural network for the evaluation of the right half of the retina was to be 0, and the value predicted by the network when evaluating the left half of the retina was 0.15 off of the expected value, then I would be assigned a value of 0.15 during evaluation of the right half of the retina.

Results

Our comparisons were between a set of 10 trials of NEAT (with recurrency allowed) on the single-task environment

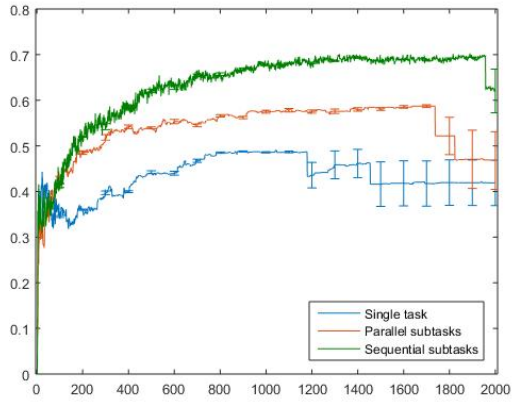

Figure 2: The mean Louvain modularity value Q in the first level of modularity, on three different retina problem environment versions, for the top-performing neural network of each generation. Error bars every 100 generations show variance. At the start, networks produced by NEAT are unmodular, and become more modular in all environments, with the sequential-subtask environment producing greater modularity than the single-task or parallel-subtask environments with $P < 0.01$.

(problem version), 10 trials on the parallel-subtask environment, and 10 trials on the sequential-subtask environment, each across 2000 generations, as described earlier. In Fig. 2, which shows mean modularity of the best-performing member of the population across trials at the lowest level of abstraction across all 2000 generations, we see that at first, NEAT is producing nonmodular solutions for all three environments, and that modularity rises rapidly to moderate levels (in the Q = 0.3 to Q = 0.4 range) in early generations. Starting as the generation number approaches 100, the single-task environment begins to show a drop, followed by a gradual low rise, followed by a series of drops. The parallel-subtask environment shows a steady moderate rise in best-individual modularity, while the sequential-subtask environment shows a steeper rise. Looking at modularities across the 2000 generations, the sequential-subtask environment has greater best-individual modularity than the parallel-subtask environment, which has greater best-individual modularity than the single-task environment, with $P < 0.01$. While fitness is not a primary focus of our study, it was generally similar in parallel-subtask and sequential-subtask environments, and superior in single-subtask environments, as the relative difficulty of the different tasks would suggest.

In addition to viewing modularity at this lowest level of abstraction, we also viewed both the emergence of higher levels of abstraction, and the modularity of those higher levels of abstraction - the emergence of both hierarchical mod-

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

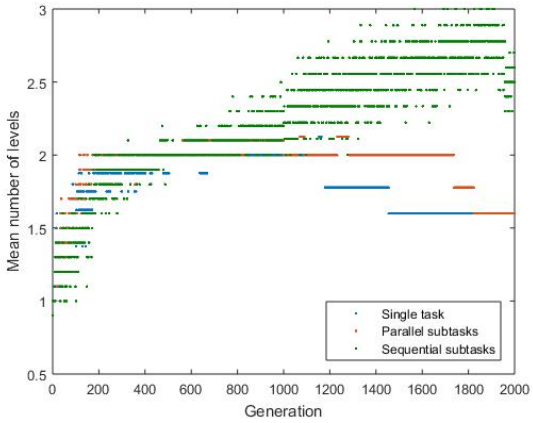

Figure 3: The mean number of levels of modularity for three different retina problem environment versions.

ularity, and modularity in the hierarchical modules themselves. We see the first of these, the emergence of hierarchical modularity, in Fig. 3, which shows the mean number of levels of modularity in the best-performing individual in each environment for every generation, as indicated by 2000 data points plotted for each environment's set of trials. In all three environments, some cases of hierarchical modularity start to emerge within the first 200 generations. However, in the single-task environment, this emergence stagnates and is then mostly lost. In the parallel-subtasks environment, a second level of hierarchical modularity gradually emerges, but stagnates. In the sequential-subtasks environment, we start seeing a third level at earlier generations than we do in either of the other two environments, even reaching the point of a mean number of three levels at points later in the simulation. So we can see that sequential subtasks, in this problem space, are more effective at causing hierarchical modularity to emerge than a single task or parallel subtasks are.

The degree of modularity in these emerging levels of hierarchical modularity is depicted in Fig. 4. In this figure, if a level does not exist, it is assigned a value of zero. The second level, in Fig. 4a, shows a similar pattern to what existed in the first level of modularity. Mean modularity for best individuals rises in all three environments, but levels out and then declines in the single-task environment. It rises in the parallel-subtask environment, but in the sequential-subtask environment, it rises more quickly and has a higher asymptote value. In the third level, shown in Fig. 4b, we can see the starkest differences between the results produced by different environments, in part because the sequential-subtask environment is the only one in which a third level of hierarchical modularity consistently emerges. The single-task environment shows almost no third level. For the parallel-subtask environment, there is a period of several hundred generations, late in the evolutionary process, in which there

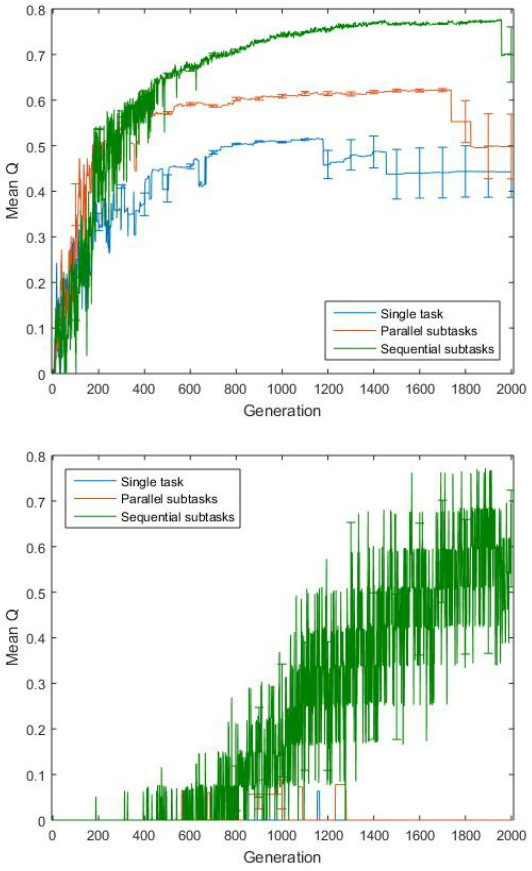

Figure 4: Mean Louvain modularity values at upper levels of abstraction. Error bars represent variance. At the start, networks produced by NEAT are unmodular at these levels of abstraction. At the second level, they become more modular in all environments, with the sequential-subtask environment producing greater modularity than the single-task or parallel-subtask environments with $P < 0.01$, while at the third level, the sequential-subtask environment is the only one to produce modules of nontrivial strength. a) The mean Louvain modularity value Q in the second level of modularity, on three different retina problem environment versions. If a second level does not exist, the value is set equal to 0. b) The same value for the third level of modularity.

is a third level, but the modules that have emerged are weak. While the result for the sequential-subtask environment is very noisy, it shows a strong upward trend, reaching, at its peaks, similar mean Q values as were found in the first two levels.

In all environments, there is a pattern at both the first and second levels of abstraction where, late in the trial, mean modularity drops and variance increases. This occurs earliest in the single-task environment, in which averages drop to near what they were at the start of the evolutionary pro-

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

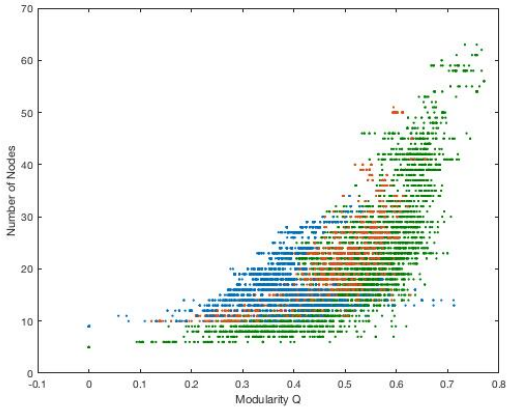

Figure 5: Network size (number of nodes) as a function of first-level modularity for all top-performing networks produced in the first 500 generations of evolution. As in other figures, the colors blue, orange, and green represent networks evolved in single-task, parallel-subtask, and sequential-subtask environments, respectively.

cess, and latest in the sequential-subtask environment. At least some of both the drop in mean modularity and the drop in variance appear to be due to nonmodular outliers. Given the tendency, in the absence of modularity-promoting factors, for evolutionary algorithms to produce nonmodular solutions, these outliers may be cases where methods of modularity promotion have gradually been overcome by this tendency. In the single-task environment, with no environmental modularity promotion mechanism, this tendency would make it unsurprising that evolved solutions, whose single-level modularity never got very much higher than it was at the start of the evolutionary process, would regress. In other environments, the environmental modularity promotion mechanism appears to provide at least a partial buffer against regression.

One potential factor contributing to increased modularity in more complex environments, with more difficult tasks, is network size, with greater single-level modularity, or more levels of hierarchical modularity, appearing as NEAT adds more nodes to the networks in an attempt at better performance. Network size in networks with emerged hierarchical modularity has previously found not to be a primary factor in that hierarchical modularity (Lowell and Pollack, 2016). To assess impacts of network size on single-level modularity, we plotted network size and first-level modularity (as second-level modularity closely tracked first-level) for best-of-generation networks in the first 500 generations of every trial, or 5,000 networks per environment, in Fig. 5. We show the first 500 generations because differences in module strength between environments had already clearly emerged after 500 generations, and first-level modularity was close to its apparent asymptote for all environments, as seen in Fig. 2, while spikes in network size caused by the stalling of fitness improvements in a difficult environment had not yet emerged. While there is some correlation between network size and first-level modularity, there is a visible difference in the first-level modularity of networks of similar size evolved in different environments, with those evolved in sequential-subtask environments being the most modular at this level and those evolved in single-subtask environments being the least modular.

Discussion, Conclusion, and Future Work

One interesting aspect of the results is that the neural networks in the single-task environment show an early peak in their first-level modularity, and in their second-level modularity as well as the second level emerges, but they are unable to sustain this as evolution progresses, and stagnate or lose ground. In the sequential-subtask environments, single-level modularity early in the evolutionary process is actually somewhat lower than it is in single-task environments. However, this modularity rises from its lower beginnings to a much higher value as evolution progresses. The slope of the curve is steeper, and it approaches what appears to be a higher asymptote. When we analyzed the effects of a parallel-subtask environment, single-level network modularity shows a similar but less drastic pattern to that of the sequential-subtask environment, rising gradually toward a middle-ground asymptote. It is possible that some factor causing the early modularity bump in the single-task environment then reduces the further evolvability of modularity.

Intuitively, it makes sense that a task that can be decomposed into multiple parts would be more likely to induce the evolution of modularity, or would promote it to a greater degree, than would be the case for a non-decomposable task, with subnetworks evolving to handle different subtasks. The compartmentalization of one large task into multiple subtasks limits the possible connections that each node can make, and as (Clune et al., 2013) demonstrated, fewer connections between nodes is associated with greater modularity. It is also associated with greater hierarchy (Mengistu et al., 2016). However, this does not explain why sequential subtasks promote modularity to a greater degree than parallel subtasks. One key difference between sequential-subtask and parallel-subtask environments is that in a parallel-subtask environment, networks are evolving to perform well on one *simultanous* task, whereas in sequential-subtask environments, networks are performing two different subtasks *at two different times*, which requires them to be reusable. Recombinability and decomposability are considered aspect of modularity in most fields that study it (Schilling, 2002) A reason that modular design is considered desirable by engineers is that parts can be reused. This selection pressure for reusability may also be a selection pressure for recombinability and decomposability, pro-

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

ducing modularity.

The modularity that emerged was not simply a case of two modules emerging, one for each subtask, but a hierarchical network of many modules, sometimes dozens for larger networks at the lowest level of abstraction. This may be an effect of NEAT's bottom-up approach, in which neural networks start as small as possible and build up gradually through random mutations adding nodes and edges. The randomness, combined with gradual increase in network size, and environmental inducing of modularity, could cause small modules to form gradually in different parts of the network.

The tendency of single-level modularities, in all environments, to rise toward apparent asymptotes, suggests that environments can impose constraints on modularity as well as promoting or not promoting it. As previously mentioned, the difference in neural networks evolved in parallel-subtask vs sequential-subtask environments on the retina problem is that in the latter the apparent asymptote is greater. How these asymptotes form, how environments constrain as well as promote modularity, may prove a fruitful area for further research.

Our results suggest other potentially fruitful areas of future research as well. One possibility would be to adjust parameters within NEAT, or compare it with other neuroevolution methods, in order to identify algorithm-specific factors that interact with different environments to promote modularity. Another possibility would be to combine our environmental method with a network-intrinsic method such as the connection cost method developed by (Clune et al., 2013). Finally, it would be interesting to study the effects of sequential subtasks in the evolution of actual biological systems, as (Kashtan et al., 2007) did to complement computational studies on the effects of varying environments on the evolution of modularity.

References

Arenas, A., Duch, J., Fernández, A., and Gómez, S. (2007). Size reduction of complex networks preserving modularity. *New Journal of Physics*, 9(6):176.

Azam, F. (2000). *Biologically inspired modular neural networks*. PhD thesis, Citeseer.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008.

Bolker, J. A. (2000). Modularity in development and why it matters to evo-devo. *American Zoologist*, 40(5):770–776.

Calabretta, R., Nolfi, S., Parisi, D., and Wagner, G. P. (1998). Emergence of functional modularity in robots. *From animals to animats*, 5:497–504.

Calcott, B. (2014). Chaining distinct tasks drives the evolution of modularity. In *ALIFE 14: The Fourteenth Conference on the Synthesis and Simulation of Living Systems*, volume 14, pages 701–702.

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

Clune, J., Mouret, J.-B., and Lipson, H. (2013). The evolutionary origins of modularity. *Proceedings of the Royal Society of London B: Biological Sciences*, 280(1755):20122863.

Hartwell, L. H., Hopfield, J. J., Leibler, S., and Murray, A. W. (1999). From molecular to modular cell biology. *Nature*, 402:C47–C52.

Jacobs, R. A. and Jordan, M. I. (1993). Learning piecewise control strategies in a modular neural network architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(2):337–345.

Juillé, H. (1999). *Methods for statistical inference: extending the evolutionary computation paradigm*. dissertation, Brandeis University.

Juillé, H. and Pollack, J. B. (1996). Co-evolving intertwined spirals. In *in Proceedings of the Fifth Annual Conference on Evolutionary Programming*. Citeseer.

Kashtan, N. and Alon, U. (2005). Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences of the United States of America*, 102(39):13773–13778.

Kashtan, N., Noor, E., and Alon, U. (2007). Varying environments can speed up evolution. *Proceedings of the National Academy of Sciences*, 104(34):13711–13716.

Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press.

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

Lowell, J., Grabkovsky, S., and Birger, K. (2011). Comparison of neat and hyperneat performance on a strategic decision-making problem. In *Genetic and Evolutionary Computing (ICGEC), 2011 Fifth International Conference on*, pages 102–105. IEEE.

Lowell, J. and Pollack, J. (2014). The effect of connection cost on modularity in evolved neural networks. In *ALIFE 14: The Fourteenth Conference on the Synthesis and Simulation of Living Systems*, volume 14, pages 726–733.

Lowell, J. and Pollack, J. (2016). Developmental encodings promote the emergence of hierarchical modularity. In *ALIFE 15: The Fifteenth Conference on the Synthesis and Simulation of Living Systems*, volume 15, pages 344–352.

Mengistu, H., Huizinga, J., Mouret, J.-B., and Clune, J. (2016). The evolutionary origins of hierarchy. *PLoS Comput Biol*, 12(6):e1004829.

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

Newman, M. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113.

Parter, M., Kashtan, N., and Alon, U. (2007). Environmental variability and modularity of bacterial metabolic networks. *BMC evolutionary biology*, 7(1):169.

Reisinger, J., Stanley, K. O., and Miikkulainen, R. (2004). Evolving reusable neural modules. In *Genetic and Evolutionary Computation Conference*, pages 69–81. Springer.

Scherrer, A. (2008). Matlab louvain implementation. online, 2008.

Schilling, M. A. (2002). Modularity in multiple disciplines. *Managing in the modular age: Architectures, networks and organizations*, pages 203–214.

Simmerson, M. (2006). Neat4j homepage. online, 2006.

Stanley, K. O., Bryant, B. D., and Miikkulainen, R. (2005). Evolving neural network agents in the nero video game. *Proceedings of the IEEE*, pages 182–189.

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

Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.

Thompson, A. (2012). *Hardware Evolution: Automatic design of electronic circuits in reconfigurable hardware by artificial evolution*. Springer Science & Business Media.

Variano, E. A., McCoy, J. H., and Lipson, H. (2004). Networks, dynamics, and modularity. *Physical review letters*, 92(18):188701.

Vassilev, V. K., Job, D., and Miller, J. F. (2000). Towards the automatic design of more efficient digital circuits. In *Evolvable Hardware, 2000. Proceedings. The Second NASA/DoD Workshop on*, pages 151–160. IEEE.

Verbancsics, P. and Stanley, K. O. (2011). Constraining connectivity to encourage modularity in hyperneat. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1483–1490. ACM.

Watson, R. A. and Pollack, J. B. (2000). Symbiotic combination as an alternative to sexual recombination in genetic algorithms. In *International Conference on Parallel Problem Solving from Nature*, pages 425–434. Springer.

Watson, R. A. and Pollack, J. B. (2001). Symbiotic composition and evolvability. In *European Conference on Artificial Life*, pages 480–490. Springer.

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).