2019

# Exploring the Modularity and Structure of Robots Evolved in Multiple Environments

Collin Cappelle
*University of Vermont*

# Exploring the Modularity and Structure of Robots Evolved in Multiple Environments

A Dissertation Presented

by

Collin Cappelle

to

The Faculty of the Graduate College

of

The University of Vermont

In Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
Specializing in Computer Science

August, 2019

Defense Date: April 26th, 2019
Dissertation Examination Committee:

Joshua C. Bongard, Ph.D., Advisor
Charles J. Goodnight, Ph.D., Chairperson
James Bagrow, Ph.D.
Laurent Hébert-Dufresne, Ph.D.
Cynthia J. Forehand, Ph.D., Dean of Graduate College

# Abstract

Traditional techniques for the design of robots require human engineers to plan every aspect of the system, from body to controller. In contrast, the field of evolutionary robotics uses evolutionary algorithms to create optimized morphologies and neural controllers with minimal human intervention. In order to expand the capability of an evolved agent, it must be exposed to a variety of conditions and environments.

This thesis investigates the design and benefits of virtual robots which can reflect the structure and modularity in the world around them. I show that when a robot's morphology and controller enable it to perceive each environment as a collection of independent components, rather than a monolithic entity, evolution only needs to optimize on a subset of environments in order to maintain performance in the overall larger environmental space. I explore previously unused methods in evolutionary robotics to aid in the evolution of modularity, including using morphological and neurological cost.

I utilize a tree morphology which makes my results generalizable to other morphologies while also allowing in depth theoretical analysis about the properties relevant to modularity in embodied agents. In order to better frame the question of modularity in an embodied context, I provide novel definitions of morphological and neurological modularity as well as create the sub-goal interference metric which measures how much independence a robot exhibits with regards to environmental stimulus.

My work extends beyond evolutionary robotics and can be applied to the optimization of embodied systems in general as well as provides insight into the evolution of form in biological organisms.

# Citations

Material from this dissertation has been published in the following form:

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

AND

Cappelle, C.K., Anton, B., Bongard, J.C.. (2017). "Reducing Training Environments in Evolutionary Robotics Through Ecological Modularity." In Conference on Biomimetic and Biohybrid Systems, pp. 95-106. Springer, Cham.

AND

Cappelle, C.K., Bongard, J.C.. (2018). "Embodied Embeddings for Hyperneat." In Artificial Life Conference Proceedings, pp. 461-468. One Rogers Street, Cambridge, MA 02142-1209 USA journals-info@ mit. edu: MIT Press.

AND

Cappelle, C.K., Bongard, J.C.. (2019). "Morphological Cost Facilitates the Evolution of Modularity." *In review for PLoS computational biology.*

# Acknowledgements

Thanks to my advisor Josh Bongard for this amazing opportunity to explore the fascinating world of virtual creatures. I would like to thank James Bagrow, Laurent Hébert-Dufresne, and Charles Goodnight for taking the time to help me on this journey.

Thanks to Katie Ettman for being the best partner a human could ask for.

Thanks to my parents Frank and Jill, and my sister Kelsey for their unending support.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

A continual challenge in robotics is the design of both the robot's morphology (i.e. body) and controller to adequately accomplish some set of tasks. An even more difficult challenge is to create an algorithm which automatically designs a portion or entirety of the robot based on the task or tasks at hand. One field which aims to automate the design and creation of robots is Evolutionary Robotics (ER). Briefly, ER uses heuristics taken from biological evolution (known as Evolutionary Computation) and applies them to populations of robots in order to increase the ability of the robots to accomplish a given task. One way to evolve more complex robots is to evolve them in many different environments, thereby exposing the robots to a range of different initial conditions. However, this leads to a tremendous increase in the number of evaluations in a process which already requires a large number of assessments of candidate solutions.

This thesis was born out of exploration into methods to increase the efficiency in which robots are evolved in multiple environments.

Specifically, the focus of this thesis is the ability of evolved robots to break down

their environment into *unfamiliar combinations* of *familiar percepts*. This means the robot is able to break down the environment into the independent components or modules which it has experienced before, possibly in a different organization. By evolving robots which reflect the modularity and structure of the environment they act within, this thesis finds a scalable path forward for increasing the complexity of evolved robots while limiting the number of evaluation instances.

## 1.1 Evolutionary Computation

Evolutionary Computation is a class of optimization methods which draw inspiration from the evolution of biological organisms [34, 43]. Using basic concepts such as mutation, sexual recombination, and fitness based survival, Evolutionary Computation searches for increasingly better solutions for some desired problem.

One basic form of Evolutionary Computation is the Genetic Algorithm (GA). In its most basic form, a GA starts with a population filled with random individuals [64, 34]. The individuals are evaluated against some criteria in order to determine their fitness. Next, parent selection occurs in which members of the population are chosen to be parents of the next generation. Parents can be chosen randomly, based on fitness, or based on some other heuristic. After parents are chosen, offspring are created from the parents through the use of mutation and/or crossover.

When an individual undergoes mutation to create one or more offspring, the parents genotype is slightly altered. For example, in many cases the genotype may be a bit vector and a mutation is represented by a bit flip at a random location. Mutation allows for the introduction of new genetic material into the population and generally

occurs in a similar fashion to the mutation of DNA where a single base pair is altered. This allows for more exploration of the search space.

Whereas mutation is a unary operator requiring only one parent to occur, crossover is an $n$-ary operator combining genetic material from multiple sources [34]. Although crossover can occur with more than two parents, it is generally used as a binary operator. In this case, two parents are selected for crossover and a portion of each parents genetic material is combined in some manner to create a wholly new individual which maintains aspects of its genotype from its parents. This is similar to the way sexual recombination works where the child has half its DNA from one parent and half from the other. Going back to the example of the bit vector, one simple method of crossover is single-point crossover. In single-point crossover, a location in the bit vector is chosen splitting the vector into left and right portions. The offspring is then created by combining the left portion of one parent and the right portion of another. The goal of crossover is to maintain small, high-fitness portions of the genome, called building blocks, and combine them with other building blocks in the genomes of other individuals. Crossover is aimed to aid in the discovery of high-fitness solutions to the original problem, whatever it might be.

While the work presented here mainly uses various forms of mutation operators to evolve populations of robots controlled by neural networks, it does suggest how successful crossover can occur for neural networks which is known to be a difficult task [46, 102]. This thesis uses evolutionary algorithms to both present a scalable way forward in the realm of evolutionary robotics as well as to propose possible pressures for the evolution of modularity in biology.

Evolutionary Computation algorithms are primarily used in numerical or design

problems, generally with little to to no relation to evolutionary biology. In this case, the goal is not to answer deeper questions about the nature of evolution but rather use general concepts from natural evolution to provide a heuristic method of optimization. These algorithms and methods have been used in the design of antenna [52], to solve symbolic regression tasks [3], and in the creation of art [97].

As a numerical optimization technique, two important factors to consider are the rate of a convergence to a satisfactory solution and overall scalability to larger problems [43]. This work addresses both issues by showing that modular robots which correctly reflect the modularity of the environment and desired tasks are both more evolvable on a given set of environments and more scalable when considering future unknown environments.

Evolutionary Computation has also been used to answer general questions regarding evolutionary biology. Specifically, evolutionary methods have been used to address the evolution of modularity [23, 36]. A more detailed explanation of modularity and previous work is presented in Section 1.4. This work extends previous work into the origins and evolution of modularity in organisms by considering embodied agents represented by robots.

## 1.2 Evolutionary Robotics

Evolutionary Robotics applies Evolutionary Computation methods to the design and construction of robots [82]. Evolution can be applied to any or all aspects of the robots controller and morphological design.

In most cases researchers will start with a fixed morphology, already knowing the

form of the robot they want to evolve, some of which are already physical robots built by the researchers themselves [59] or bought off the shelf [77]. In these cases the robots controller, which is generally represented by an artificial neural network, is under evolutionary control. Even within these experiments there is a wide variety of choice for how to evolve the robot [32]. In the most basic approaches only the synaptic weights of the neural network are mutated over an evolutionary run [81]. This manner of evolution has proven effective for a variety of experiments [22, 41]. Although evolution plays a limited role in these experiments, there is still a general need to simulate the robots in order to reduce the amount of computational time to perform multiple independent trials [32]. Even in cases where only the controller is being evolved, reducing the number of evaluations is crucial to ensure computation times are not prohibitively long.

In 1994, Karl Sims introduced the first artificial evolution of both morphology and control in order to create lifelike behavior in artificial agents [98]. This allowed a wider variety of behavior and form while also tremendously increasing an already large search space. The utilization of co-evolution of morphology and control has continued with examples such as [71] which aimed to design body and control while also creating a physical instance of the robot through the use of a 3D printer and [20] which applied protection to genomes which recently underwent a morphological change. All of these experiments focused on single simple tasks such as locomotion. The agents found in these examples are not robust to changes in environment nor do they even perceive the environment directly. Alterations in the environment or the introduction of the robot to wholly new environments would have drastic impacts on the robots behavior.

Many ER experiments still limit robots to a single or a small number of environments and tasks [32]. Even less actually address the long term scalability of the evolved solutions to new environments. Although the original issues of scalability were raised in 1996 [74], to date no long term plan has been proposed for the future of ER to scale to large numbers of environments and tasks. This thesis presents a first step to answer the original concerns raised by Matarić and Cliff [74]. Using their characterization of the environment space, this thesis shows that not all environments need to be evaluated during evolution, thereby increasing the scalability of ER to larger sets of environments.

## 1.3 Characterization of Environments

When training any agent, embodied or otherwise, it is important to consider the necessary information the agent needs to be exposed to in order to optimize for adequate behavior. In image recognition tasks, this could be the number and diversity of images needed in order to train a classifier to recognize the difference between photos with and without a cat present [66]. In evolutionary robotics, the information required to evolve adequate behavior is the environment in which the robot is situated [50]. The environment of the real world exists in a continuum. This is infeasible to replicate in simulation so instead the environments must be discretized and simplified. One method is to describe the set of environments by the free parameters and variations of said free parameters [74].

Free parameters are the qualities of the environment which vary independently of one another. For example, consider the task of a robot moving towards an object.

One free parameter could be the location of the object. Another free parameter could be the color of the object. Both the location and color of the object can vary independently without influencing the value of one another. The number of free parameters in an environment is denoted $f$.

The variations of the free parameters are the specific quantities each free parameter takes in a given environment. Following from the previous example, the object could be located to the left or right of the robot and can be red or blue. Each free parameter then has two variations. We denote the number of variations as $n$.

When both $f$ and $n$ have been ascertained, the resulting number of environments is $n^f$. The total environment space can be represented by a hypercube where each dimension is a free parameter, each position along a dimension is a variation of that free parameter, and hypervoxel is a single environment (Fig. 1.1).

The independent free parameters can be thought of as the independent modules which exist in the environment. The overall goal is then for the robot to accurately distinguish the modules in the environment.

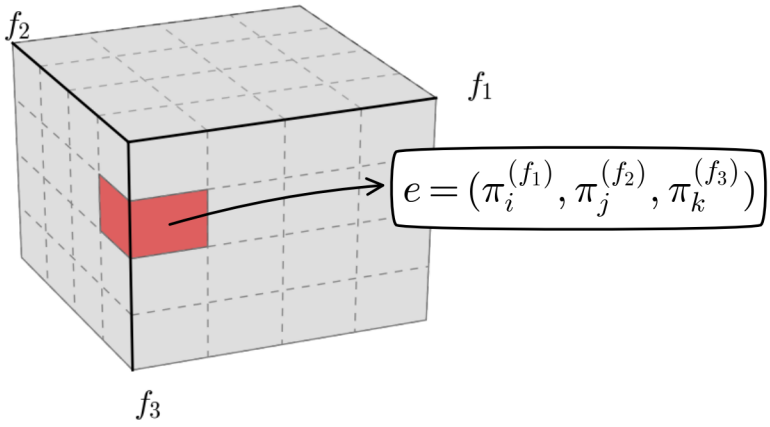



*Figure 1.1: Sample environment space with four variations of three free parameters* $\{f_1, f_2, f_3\}$ *resulting in* $4^3 = 64$ *total environments. The red highlighted voxel represents the single environment* $e$ *consisting of* $\pi_i$ *variation of parameter* $f_1$*, the* $\pi_j$ *variation of* $f_2$*, and the* $\pi_k$ *variation of* $f_3$*.*

## 1.4 Modularity

The presence of modularity is ubiquitous in nature [16, 110]. From organelles within cells to macro-scale interactions on the World Wide Web, modularity has been used to describe natural and human phenomena.

While the specific definitions of modularity are limited to the domain they describe, most definitions rely on modularity requiring some level of independence and separation. In networks, this independence comes from limited connections between communities. This thesis expands beyond network approaches and explores embodied approaches which require mechanical independence as one aspect contributing to a robot's modularity.

### 1.4.1 Network Approaches

The primary method of studying modularity in artificial agents has been through the use of networks which represent the neural controllers of those agents. Networks are powerful tools because they can represent social relationships [111, 96], infrastructure [29], and biological processes [51, 87].

The prevalence of networks in nature has led to a wealth of research into the quantification of various network properties [83, 15, 89]. One such important metric is $Q$ [79]. $Q$ is measured in two stages. First the network is split into groups of nodes called the community partition, where each group is a separate community. Then this partition's score is assigned based on the fraction of internal community connections compared to a random network with the same degree distribution. A network's $Q$ value is the score given to the community partition which is maximal [79].

$Q$ has been widely implemented to measure the modularity present in various types of networks [39, 61, 75, 100]. Most relevant to this thesis is the use of $Q$ used as a metric of modularity in the computational networks of artificial agents [57, 38, 23]. The retina task in particular played an important part towards the formation of this thesis.

**Retina Task**

The retina task was set forth as a basic method to explore the evolution of modularity in networks (Fig. 1.2) [57].

The task consists of a network rewarded for evaluating logical functions of recognized patterns in the left and right of a $4 \times 2$ pixel array. Kashtan and Alon [57] found that when the logical function was altered in a specific way, thereby exposing the networks to modularly varying goals, modular networks adapted faster to the change compared to non-modular networks, allowing the modular offspring drive the latter to extinction.

Further work using the retina task has explored the use of connection cost to more easily enable the evolution of modularity without the need for modularly varying goals over time [24].

The use of $Q$ to explore structural modularity is only relevant when the agent can be perfectly described as a network. For a computational network with inputs and outputs, $Q$'s measure of structural modularity may not align with the functional modularity present (or not) as determined by the relationship between input and output. One can imagine modules existing in a computational network which do not connect to either input or output. Thereby $Q$ can be fooled into showing modularity

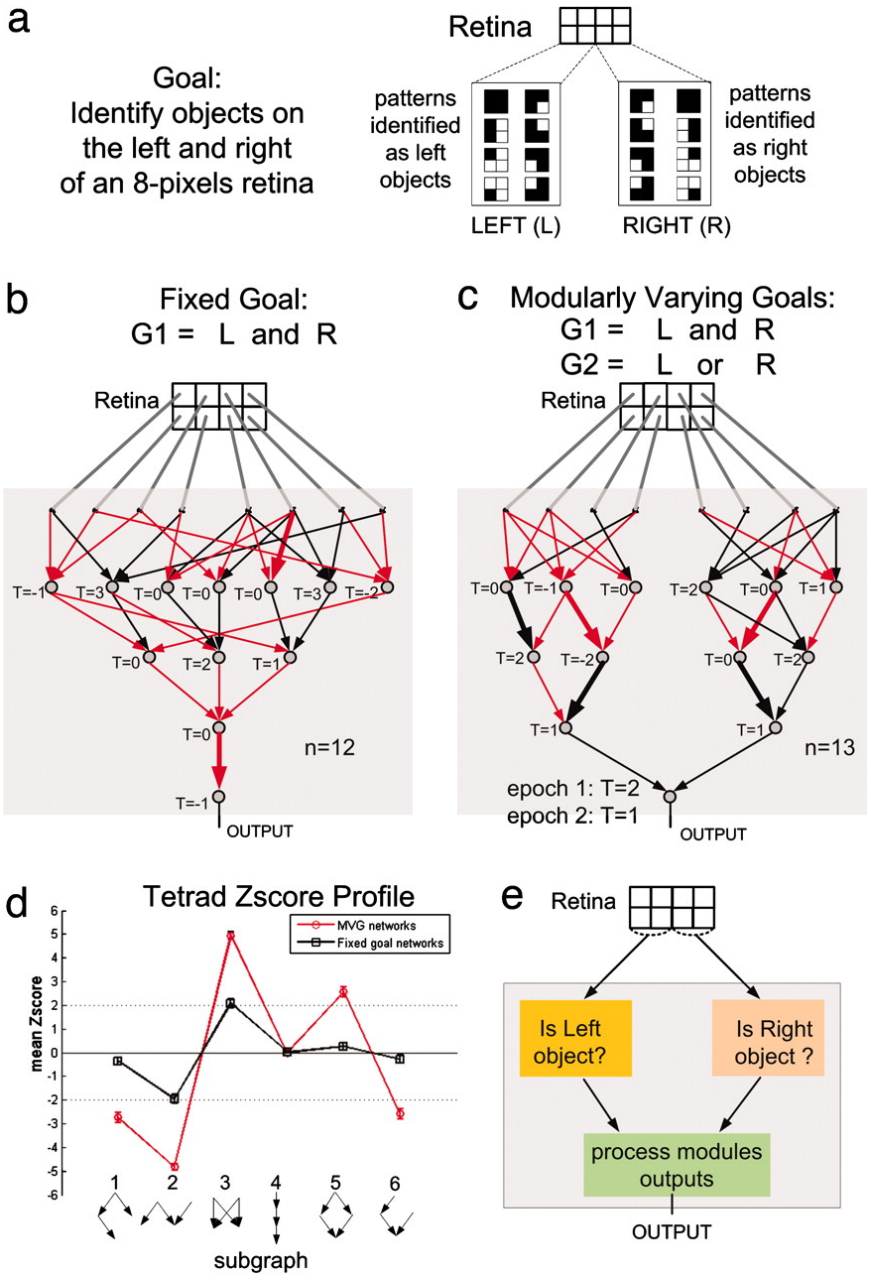



*Figure 1.2:* ***Retina Task****. Diagram of the retina task and resulting modularity of an evolved agent when the population is exposed to modularly varying goals (From [57])*

.

when there is no meaningful modularity present. That is $Q$ can report high modularity in an agent even though the agent may not exhibit any of the benefits of being modular.

$Q$ further fails to describe the impact of a computational network's output if the network is embodied. An embodied network's motoric output has the potential to influence future input sensations. For example, consider a global position sensor on a humanoid robot. As the robot's arms are actuated, it is unlikely that the position of the robot's torso will change, however if the legs are actuated there is a complex relationship between the motoric output represented by the angles of the joints in the legs and the reported position of the sensor in the torso. These complex interactions defined by the motor to sensor relationship, as guided by the environment the agent is situated in, are not captured by $Q$. Contrast this with the sensor to motor relationship as defined by synaptic connections in the neural network of the agent whose modularity is captured by $Q$. Figure 1.3 details four extremes of sensory-motor independence in embodied networks, all of which potentially have high $Q$ modularity. The training of these embodied networks represent an embodied approach towards artificial intelligence.

### 1.4.2 Embodied Approaches

Researchers have explored modularity within embodied agents, however it has largely been limited to neural modularity [13]. The relationship between adaptable morphology and the evolution of populations with high degrees of modularity has not been directly explored.

In [1] robots were exposed to environments with varying degrees of difficulty and

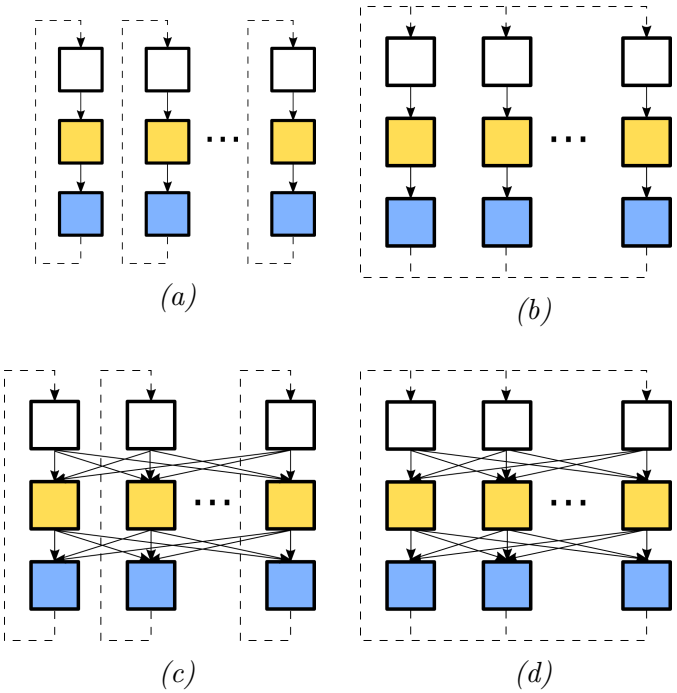



*Figure 1.3: Four embodied networks displaying different levels of functional independence while potentially all receiving high Q scores. White, yellow, and blue squares denote banks of sensor, hidden, and motor neurons respectively. These banks may contain one or many neurons each. Solid arrows between the squares represent direct synaptic connections between neurons contained in the squares. Dashed arrows represent the impact of a motors actuation in the environment on the resulting sensation of the network. Networks* $(a)$ *and* $(b)$ *have high sensor to motor independence because banks of sensors only have the potential to impact a subset of motors. Alternatively, changes in sensor values in networks* $(c)$ *and* $(d)$ *propagate to all motors indicating high sensor-to-motor dependence. Panels* $(a)$ *and* $(c)$ *contain networks with high levels of motor-to-sensor independence while panels* $(b)$ *and* $(d)$ *contain networks of low independence.*

as such, robots are evolved which reflect the complexity in the world. The robots were tasked to move across an environment which contained blocks of low friction ice. The larger and farther apart the the ice blocks were, the more complex the environment was considered to be. These more complex environments led to the evolution of robots with a more complex morphology. However, this work only considers populations exposed to a single environment over the course of an evolutionary trial instead of exposing the populations to many different environments.

Due to the aforementioned time complexity of evaluation in multiple environments (i.e. $O(n^f)$), many evolutionary robotics experiments limit evaluation to a single environment. This leads to robots which are fragile with regards to their environments: as soon as any aspect of their environment is altered, it is highly likely they are unable to exhibit the behavior for which they were evolved.

In pursuit of more robust controllers, novelty search and quality diversity (QD) have proven to be effective methods [90, 67]. These methods discover controllers which, when placed in novel environments, achieve comparatively high performance when compared to controllers optimized for performance alone. In these experiments, each environment was represented as a maze that the robot was tasked to solve by reaching the end. The robustness to novel environments exhibited by the robot does not indicate the robot can break apart and reconstruct each maze into its independent parts, rather it shows that these mazes can all solved by a robot exhibiting a single general maze-solving behavior.

No methods have yet explore the specialization of both morphology and control of a robot with respect to environmental stimulus.

Further, past experiments do not address what it means to be morphologically

modular. In many ways, morphological modularity can be thought of as mechanical dexterity (here used as the ability to isolate independent movement within the body to accomplish a task). By having a body which enables independent movement without impacting sensation in other parts of the body, the robot is more able to break down the environment along the lines of the free parameters.

## 1.5 Contribution Outline

Research in this thesis began by exploring methods to reduce the computational complexity necessary to evolve robots in multiple environments. In order to accomplish this reduction, modularity in both body and brain were examined as methods by which a robot is able to recognize the independence present in the free parameters of the environment. Overall, this thesis provides an important novel framework for discussing the relationship between morphology, control, and environment by first providing definitions for morphological, neurological, and ecological modularity and then tying them all together with the sub-goal interference metric $I$.

Chapter 2 provides the following definitions: **Morphological Modularity** and **Neurological Modularity**. Armed with these definitions, it can be shown that a robot which is morphologically and neurologically modular needs only be evolved in a reduced subset of environments. Specifically this reduction is found to be from exponential to linear ($O(n^f) \rightarrow O(n)$) across multiple values of $n$.

Chapter 3 introduces and defines the novel concept of **Ecological Modularity**. This work states that even if a robot is morphologically and neurologically modular, it may not be able to independently recognize variations across independent free

parameters. This limits the utility of the result in Chapter 2 by finding the reduction in necessary training environments to be $(O(n^f) \rightarrow O(n^{f-m+1}))$ where $m$ is the maximum number of free parameters the robot can independently recognize.

In Chapter 4, in contrast to the previous chapters, evolution is able to alter the relationship between neurons. Chapters 2 and 3 assume a fixed neural topology, thereby fixing which sensor impacts which motor. Chapter 4 explores the impact of having a developmental regime in which spatial relationships in the neural architecture reflect the structure present in the task environment. Using **embodied embeddings** for the well known algorithm HyperNEAT, embeddings which maintain the same modular and hierarchical structure as the task environment are shown to be more evolvable than those that do not.

Finally, Chapter 5 introduces the concept **sub-goal interference** and the corresponding metric $I$. $I$ gives insight into a robot's perception to environmental stimulus. A high value of $I$ means that changes in one independent sub-goal impacts performance in a different independent sub-goal. Therefore a high $I$ indicates the robot cannot break down the environment into its independent parts. A low value of $I$ means the robot perceives each independent sub-goal as independent. When combined with a performance metric, $I$ can be used to accurately determine how well a robot specializes to the specific sub-goals present in the environment. Further, the work of Chapter 5 introduces the use of **morphological cost** as an effective and simple method by which to guide evolution towards highly modular embodied agents.

# References for chapter 1


[1] Joshua E Auerbach and Josh C Bongard. "Dynamic resolution in the co-evolution of morphology and control". In: *Artificial Life XII: Proceedings of the Twelfth International Conference on the Synthesis and Simulation of Living Systems*. CONF. MIT Press. 2010.

[3] Wolfgang Banzhaf et al. *Genetic programming: an introduction*. Vol. 1. Morgan Kaufmann San Francisco, 1998.

[13] Josh C Bongard et al. "Evolving robot morphology facilitates the evolution of neural modularity and evolvability". In: *Proceedings of the 2015 annual conference on genetic and evolutionary computation*. ACM. 2015, pp. 129–136.

[15] Duncan S Callaway et al. "Network robustness and fragility: Percolation on random graphs". In: *Physical review letters* 85.25 (2000), p. 5468.

[16] Werner Callebaut, Diego Rasskin-Gutman, and Herbert A Simon. *Modularity: understanding the development and evolution of natural complex systems*. MIT press, 2005.

[20] Nick Cheney et al. "Scalable co-optimization of morphology and control in embodied machines". In: *Journal of The Royal Society Interface* 15.143 (2018), p. 20170937.

[22] Dave Cliff, Phil Husbands, and Inman Harvey. "Explorations in evolutionary robotics". In: *Adaptive behavior* 2.1 (1993), pp. 73–110.

[23] Jeff Clune, Jean-Baptiste Mouret, and Hod Lipson. “The evolutionary origins of modularity”. In: *Proc. R. Soc. B* 280.1755 (2013), p. 20122863.

[24] Jeff Clune, Jean-Baptiste Mouret, and Hod Lipson. “The evolutionary origins of modularity”. In: *Procs of the Royal Society B: Biological sciences* 280.1755 (2013), p. 20122863.

[29] Paolo Crucitti, Vito Latora, and Massimo Marchiori. “A topological analysis of the Italian electric power grid”. In: *Physica A: Statistical mechanics and its applications* 338.1-2 (2004), pp. 92–97.

[32] Stephane Doncieux et al. “Evolutionary robotics: what, why, and where to”. In: *Frontiers in Robotics and AI* 2 (2015), p. 4.

[34] Agoston E Eiben, James E Smith, et al. *Introduction to evolutionary computing*. Vol. 53. Springer, 2003.

[36] C. Espinosa-Soto and A. Wagner. “Specialization can drive the evolution of modularity”. In: *PLoS Comp Biol* 6.3 (2010), e 1000719.

[38] Carlos Espinosa-Soto and Andreas Wagner. “Specialization can drive the evolution of modularity”. In: *PLoS computational biology* 6.3 (2010), e1000719.

[39] Luca Ferrarini et al. “Hierarchical functional modularity in the resting-state human brain”. In: *Human brain mapping* 30.7 (2009), pp. 2220–2231.

[41] Dario Floreano and Francesco Mondada. “Evolution of homing navigation in a real mobile robot”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.3 (1996), pp. 396–407.

[43] James A Foster. “Computational genetics: Evolutionary computation”. In: *Nature Reviews Genetics* 2.6 (2001), p. 428.

[46] Nicolás García-Pedrajas, Domingo Ortiz-Boyer, and César Hervás-Martínez. "An alternative approach for neural network evolution with a genetic algorithm: Crossover by combinatorial optimization". In: *Neural Networks* 19.4 (2006), pp. 514–528.

[50] Inman Harvey et al. "Evolutionary robotics: the Sussex approach". In: *Robotics and autonomous systems* 20.2-4 (1997), pp. 205–224.

[51] Arend Hintze and Christoph Adami. "Evolution of complex modular biological networks". In: *PLoS computational biology* 4.2 (2008), e23.

[52] Gregory Hornby et al. "Automated antenna design with evolutionary algorithms". In: *Space 2006*. 2006, p. 7242.

[57] Nadav Kashtan and Uri Alon. "Spontaneous evolution of modularity and network motifs". In: *Proceedings of the National Academy of Sciences* 102.39 (2005), pp. 13773–13778.

[59] Sylvain Koos, Jean-Baptiste Mouret, and Stéphane Doncieux. "The transferability approach: Crossing the reality gap in evolutionary robotics". In: *IEEE Transactions on Evolutionary Computation* 17.1 (2012), pp. 122–145.

[61] Anat Kreimer et al. "The evolution of modularity in bacterial metabolic networks". In: *Proceedings of the National Academy of Sciences* 105.19 (2008), pp. 6976–6981.

[64] Manoj Kumar et al. "Genetic algorithm: Review and application". In: *International Journal of Information Technology and Knowledge Management* 2.2 (2010), pp. 451–454.

[66] Quoc V Le et al. “Building high-level features using large scale unsupervised learning”. In: *arXiv preprint arXiv:1112.6209* (2011).

[67] Joel Lehman and Kenneth O Stanley. “Exploiting open-endedness to solve problems through the search for novelty.” In: *ALIFE*. 2008, pp. 329–336.

[71] Hod Lipson and Jordan B Pollack. “Automatic design and manufacture of robotic lifeforms”. In: *Nature* 406.6799 (2000), p. 974.

[74] Maja Matarić and Dave Cliff. “Challenges in evolving controllers for physical robots”. In: *Robotics and autonomous systems* 19.1 (1996), pp. 67–83.

[75] David Meunier et al. “Hierarchical modularity in human brain functional networks”. In: *Frontiers in neuroinformatics* 3 (2009), p. 37.

[77] Francesco Mondada, Edoardo Franzi, and Andre Guignard. “The development of khepera”. In: *Experiments with the Mini-Robot Khepera, Proceedings of the First International Khepera Workshop*. CONF. 1999, pp. 7–14.

[79] Mark EJ Newman. “Modularity and community structure in networks”. In: *Proceedings of the national academy of sciences* 103.23 (2006), pp. 8577–8582.

[81] Stefano Nolfi and Dario Floreano. “Evolutionary Robotics: The Biology”. In: *Intelligence, and Technology of Self-Organizing Machines, Bradford Company, Scituate, MA* (2004).

[82] Stefano Nolfi, Dario Floreano, and Director Dario Floreano. *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT press, 2000.

[83] Lawrence Page et al. *The PageRank citation ranking: Bringing order to the web.* Tech. rep. Stanford InfoLab, 1999.

[87] Massimo Pigliucci. “Is evolvability evolvable?” In: *Nature Reviews Genetics* 9.1 (2008), p. 75.

[89] Nataša Pržulj. “Biological network comparison using graphlet degree distribution”. In: *Bioinformatics* 23.2 (2007), e177–e183.

[90] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. “Quality diversity: A new frontier for evolutionary computation”. In: *Frontiers in Robotics and AI* 3 (2016), p. 40.

[96] John Scott and Peter J Carrington. *The SAGE handbook of social network analysis*. SAGE publications, 2011.

[97] Jimmy Secretan et al. “Picbreeder: evolving pictures collaboratively online”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2008, pp. 1759–1768.

[98] Karl Sims. “Evolving virtual creatures”. In: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM. 1994, pp. 15–22.

[100] Ricard V Solé and Sergi Valverde. “Spontaneous emergence of modularity in cellular networks”. In: *Journal of The Royal Society Interface* 5.18 (2007), pp. 129–133.

[102] Kenneth O Stanley and Risto Miikkulainen. “Evolving neural networks through augmenting topologies”. In: *Evolutionary computation* 10.2 (2002), pp. 99–127.

[110] Günter P Wagner, Mihaela Pavlicev, and James M Cheverud. “The road to modularity”. In: *Nature Reviews Genetics* 8.12 (2007), p. 921.

[111] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*. Vol. 8. Cambridge university press, 1994.

# Chapter 2

# Morphological Modularity Can Enable the Evolution of Robot Behavior to Scale Linearly with the Number of Environmental Features

## 2.1 Abstract

In evolutionary robotics, populations of robots are typically trained in simulation before one or more of them are instantiated as physical robots. However, in order to evolve robust behavior, each robot must be evaluated in multiple environments. If an environment is characterized by $f$ free parameters, each of which can take one of $n$

features, each robot must be evaluated in all $n^f$ environments to ensure robustness. Here we show that, if the robots are constrained to have modular morphologies and controllers, they only need to be evaluated in $n$ environments to reach the same level of robustness. This becomes possible because the robots evolve such that each module of the morphology allows the controller to independently recognize a familiar percept in the environment, and each percept corresponds to one of the environmental free parameters. When exposed to a new environment, the robot perceives it as a novel combination of familiar percepts which it can solve without requiring further training. A non-modular morphology and controller however perceives the same environment as a completely novel environment, requiring further training. This acceleration in evolvability – the rate of the evolution of adaptive and robust behavior – suggests that evolutionary robotics may become a scalable approach for automatically creating complex autonomous machines, if the evolution of neural and morphological modularity is taken into account.

## 2.2 Introduction

Matarić and Cliff [74] pointed out that the time necessary to evolve robots grows with the number of environments in which the robot should behave correctly. Following their work, let $f$ be the number of free parameters in the environmental set and let $n$ be the number of features for each of these free parameters. So the total number of environments is $n^f$. (For example, if a robot must behave appropriately in environments containing two objects ($f = 2$), and each object may be small, medium, or large ($n = 3$), then there are $n^f = 3^2 = 9$ possible environments in which the robot

must perform correctly.) Thus, in order to evolve robots to perform complex behavior (which means increasing $n$, $f$, or both) the number of environments the robot needs to be evolved in scales exponentially.

[88] presented one way to reduce the number of environments evaluated in while still obtaining robust and generalized controllers for evolved robots. Using the *ProGAb* approach they were able to successfully obtain robust and successful controllers with better generalization abilities in less time than other top methods.However, their work did not look specifically at the structure of the controller and morphology as methods to reduce the necessary number of environments, nor did it categorize which environments should be trained on.

The work presented here demonstrates that morphological and neural modularity is one possible way to reduce the number of environments needed for evolving robust behavior.

Modularity is ubiquitous at all levels of biological organization, from cells to distinct species. Explaining why such modularity exists, and how it evolved, remains an important question in biology. Much work has focused on how modularity evolves in non-embodied systems, but relatively little work has focused on the impact of modularity in evolving embodied systems. The work presented here contributes to this latter aim.

### 2.2.1 Non-embodied modularity.

[109] argued that a combination of directional and stabilizing selection, acting on different parts of the organism's phenotype, should lead to modular developmental programs. Such modularity would enable evolutionary changes to that part of

the phenotype experiencing directional selection, while retaining the structure and function of the other parts of the phenotype under stabilizing selection.

This theoretical argument was confirmed by a number of computational experiments. [72] showed that environmental change can be a catalyst for the evolution of modularity. That work was followed by experiments in which non-embodied Boolean networks [38] or neural networks [57, 23] were evolved to perform various tasks. The tasks and fitness functions were chosen in such a way as to favor networks that computed partial results using separate genetic or neural modules; changes to the fitness function over evolutionary time favored networks that could rapidly change how those partial results were combined. Thus, stabilizing selection came to bear on the partial results, while directional selection acted on how those partial results were combined.

More recently, it has been shown that selecting for sparse networks helps to favor the evolution of modular networks. [38] accomplished this by formulating a biased mutation operator that favors low in-degree network nodes. [23] used a multiobjective approach in which one objective was to minimize the number of edges in the network. [4] showed that this relationship between sparsity and modularity can be exploited to enhance the evolution of modular networks by seeding the initial population with sparse rather than random networks.

Modularity is a desirable property of artificial systems for a number of reasons, beyond just the desire to create biologically-inspired artifacts. First, modular systems possess a form of robustness: modular systems can more rapidly adapt to certain kinds of changes in their environments, compared to non-modular systems. Second, modular neural networks are better able to avoid catastrophic forgetting than non-modular networks [35]. Catastrophic forgetting [44] is a common problem in ma-

chine learning whereby a learner must forget something in order to learn something new. Third, complex predictive models and dense, non-modular networks can suffer from the pathology of overfitting: they fail to generalize to novel environments [60]. Modular networks can avoid overfitting by internally reflecting the modularity in its environment: it responds appropriately in a 'new' environment which is actually just an unfamiliar combination of familiar percepts.

### 2.2.2 Embodied modularity.

A modular robot may likewise be robust, and avoid catastrophic forgetting and overfitting, but there are additional challenges that arise when evolving embodied agents compared to non-embodied networks and morphologies.

Embodied cognition is a particular approach to the understanding of intelligence which holds that the body must necessarily be taken into any account of adaptive behavior [14, 21, 86]. One repercussion of the embodied cognitive stance is that, if neural controllers are evolved for artificial embodied agents (i.e. robots), a given robot body plan may facilitate or hinder the evolution of desirable traits. In the context of modularity, previous work showed that there do exist body plans in which modular neural controllers will evolve [10].

Follow-on work demonstrated that, given appropriate conditions, evolution will find such body plans [13]. However, in [13], the morphology itself was not modular, only the neural networks that evolved to control it.

Here we investigate another aspect of the relationship between morphology and modularity: For a given task environment, must both the body and neural controller be modular, and if so, in what way? Before addressing these issues however, we must

define both neural modularity and morphological modularity.

### 2.2.3 Neural modularity and morphological modularity

In this work we investigate robots controlled by artificial neural networks. A common approach to measuring the amount of modularity in a network is to investigate its connectivity: a network that has dense connectivity within subsets of nodes, and relative sparsity between those subsets, is said to be modular [79]. Following this approach, we here investigate modular neural controllers in which subsets of sensor, internal, and motor neurons are connected, but there are no synaptic connections between these subsets. We compare these to non-modular networks in which any sensor can influence any motor.

In a neural controller in which sensor information flows from sensor neurons to internal neurons to motor neurons, this structural approach to modularity implies a functional repercussion. If subsets of sensors and motors are completely structurally independent, they will be functionally independent as well: changes to a subset of sensors will only have an influence on a subset of motors.

Thus, we here define neural modularity in the following manner.

**Neural modularity:** A neural network with $i$ sensor neurons $S = \{s_1, s_2, \ldots, s_i\}$ and $j$ motor neurons $M = \{m_1, m_2, \ldots, m_j\}$ is defined to be modular if every possible change to less than $i$ of the sensors results in changes to less than $j$ of the motors.

Conversely, in a non-modular neural controller, it is possible for a change to fewer than $i$ sensors to influence the new values of all $j$ motors. It is possible that a non-modular neural controller may internally extinguish certain sensor dynamics from reaching some motors, but we disregard this case in the present work. This results in a simplified, binary definition of modularity: either a neural controller is modular or it is not. Here, we investigate robots with both types of neural controller.

This approach to defining the modularity of robot neural controllers suggests a similar approach for defining the modularity of a robot's body plan:

**Morphological modularity:** A robot is defined to be morphologically modular if a change in less than $j$ of its motors results in a change in the state of the world registered by at least one and strictly less than $i$ of the robot's sensors.

One common definition of morphology is any agent sub-system that mediates between its controller and its environment. More specifically, when an agent acts, it alters its relationship with its environment. If it is equipped with sensors, it can register this change. The above definition of morphological modularity captures the intuition that structural independence of the body, like structural independence of a neural network, implies functional independence: if a robot moves one part of its body that is independent of the rest of its body, local sensors will register the action, but more sensors on other morphological modules will not.

Armed with these two definitions, one can investigate four classes of robots:

1. those that are morphologically and neurally non-modular;

2. those that are morphologically modular but neurally non-modular;

3. those that are morphologically non-modular but neurally modular; or

4. those that are morphologically and neurally modular.

In this study we evolve robots belonging to the first, second, and fourth class. One can deduce that robots which belong to the third class are functionally equivalent to those which belong to the first class: If a morphologically non-modular robot moves, its motion will affect all of its sensors. These sensors will then affect all motors, regardless of whether its neural controller is modular or not. Further, for this instance of the treebot, there is no design of a robot of the third class with a completely modular controller where both leaf sensors influence the motor neuron. If the controller was modular only one or none of the leaf sensors would influence the motor neuron.

Although modular robots have been the focus of a number of studies [113, 40], here we compare morphologically modular and non-modular robots to investigate a specific and new question: if modular and non-modular robots are evolved in an increasing number of environments, are the robots with modular controllers able to detect familiar percepts combined in unfamiliar ways, and, with a modular morphology, respond appropriately?

This question brings to light a challenge for modular, embodied agents that modular, non-embodied systems do not experience. Even if an embodied agent has a modular neural controller with which it detects novel combinations of familiar percepts in a new environment, once it moves, its perceptions will change, and the environment may no longer 'look' modular. We show here that movement in a new environment continues to appear modular from the robot's point of view only if it also has a modular morphology: it is free to move in response to independent per-

cepts as it did previously, without disrupting the sensory signals arriving at other morphological modules.

The methods employed for investigating this issue are described in the next section. Section 2.4 reports our results, while sections 2.5 and 2.6 provide some analysis and concluding remarks, respectively.

## 2.3 Material & Methods

This section describes the body plans of the simulated robots (Sect. 2.3.1), their various controllers (Sect. 2.3.2), the task environments they operated within (Sect. 2.3.3), the evolutionary algorithm used to optimize their controllers in those environments (Sect. 2.3.4), and the experimental design (Sect. 2.3.5).

### 2.3.1 The robot morphologies.

Two robot morphologies were considered: one which is modular and one which is non-modular. Figures 2.1a and 2.1b represent robots with modular morphologies while Figure 2.1c represents the non-modular one.

Robots were instantiated as trees composed of hierarchically-branching segments. Both robot morphologies considered here were composed of one root branch and two leaf branches. Each branch had length one, and the leaf branches were placed at 45 degree angles from the base. The robot contained three joints: one connecting the base branch to the environment itself (the base joint), and two that connect the base of each leaf branch to the tip of the root branch (the leaf joints). In the modular robot, the leaves were free to move independently of one another and the root was

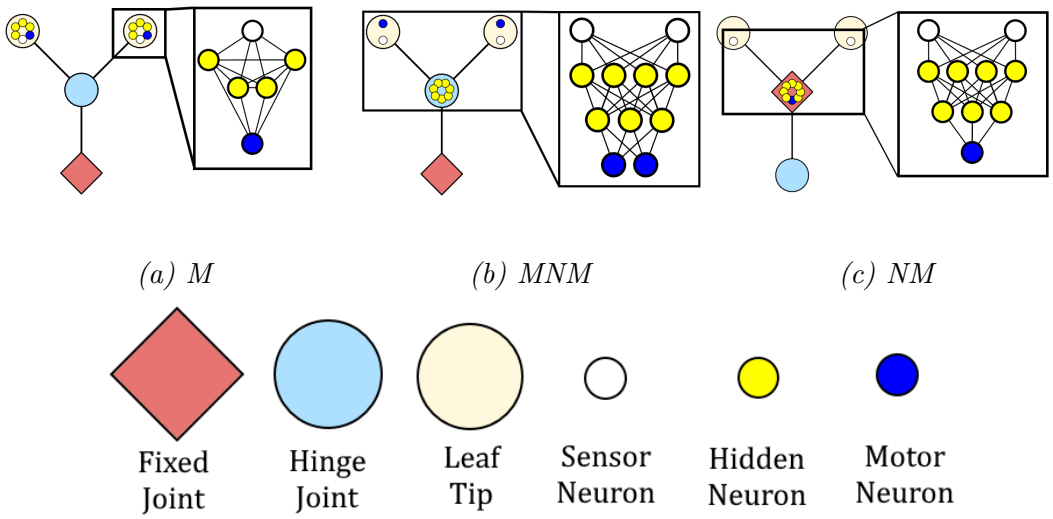



*Figure 2.1: The controllers and morphologies for each of the three robots. The red diamonds indicate that branches connected at that position are fixed relative to one another. Large blue circles indicate that the branches rooted at the circles are free to move independently of one another. Beige circles represent leaf tips. The small circles represent neurons: Blue neurons represent motor neurons, white represent sensor neurons, and yellow represent hidden neurons. The modular robot is represented by (a) on the left, the mod-nonmodular (b) is in the middle and the non-modular robot is represented by (c) on the right. Blow ups of the network structure are included. All hidden neurons and motor neurons have recurrent self-connections which are not depicted. Further, all connections except those with the sensor neurons are two-way.*

fixed, whereas in the non-modular robot the leaves were fixed and the root was free to move.

In the non-modular robot, this was accomplished by instantiating the base joint as a rotational hinge joint and the two leaf joints as fixed joints. In the modular robot, the base joint was fixed and the two leaf joints were rendered as rotational hinge joints. The base hinge joint movement was restricted to rotations of $[-120^o, 120^o]$ and the leaf hinge joints restricted movement to rotations of $[-45^o, +45^o]$ around the vertical axis. These angles are relative to the initial angle of the joint, which is treated as zero degrees.

The robots were designed in this way such that a single parameter could dictate how modular the robot's body plan was. If we define $[-45\alpha^o, +45\alpha^o]$ as the range of rotation of the leaf joints, $[-120(1-\alpha)^o, +120(1-\alpha)^o]$ as the range of rotation of the base joint, and restrict $\alpha$ to $[0, 1]$, then higher values of $\alpha$ create more modular robots, with $\alpha = 0$ and $\alpha = 1$ corresponding to the maximally non-modular and maximally modular robots investigated here. The robot with $\alpha = 0$ is considered morphologically non-modular according to the definition above, and any robot with $\alpha > 0$ is considered morphologically modular.

### 2.3.2 The robot controllers.

Three robot controllers were considered in this work. The first makes the robot neurologically modular (Fig.2.1a) while the second and third makes the robot neurologically non-modular (Fig. 2.1b,c). All controllers contain two distance sensors (the small blue circles if Fig. 2.1), one in each of the two branches of the robot's body plan. These sensors emit a beam that enables the robot to sense the distance from a

branch to any objects in the environment. The value returned by this sensor is the length of the beam. The maximum length of the beam, if unobstructed, was set to 10 units so the largest value the sensor neuron could have is 10.

Controller M (Fig 2.1a) consists of a sensor neuron, a motor neuron, and four hidden neurons in each leaf branch. The sensor feeds into all of the hidden neurons which are completely interconnected with each other. All of the hidden neurons also have connections to the motor neuron, which also is connected back to all of the hidden neurons. Finally, all of the hidden neurons and the motor neuron are self connected, giving the M robot a total of 12 neurons and 50 synapses.

Controller MNM(Fig2.1b) consists of two sensor neurons, seven hidden neurons, and two motor neurons. The hidden neurons are in a two layer structure. The input from the sensors is passed into each of the four neurons in the first hidden layer. They, in turn, feed forward into the second hidden layer. The second layer has synapses connected back to the first one and also forward to the motor neurons. The motor neurons are also connected back to the second hidden layer. Finally all of the hidden neurons and the motor neurons are self connected. Therefore MNM has 11 neurons and 53 synapses.

Controller NM (Fig 2.1c) consists of two sensor neurons, seven hidden neurons, and one motor neuron. The hidden neurons are organized in a two layer structure. The sensor values input into the four neurons in the first layer which then feed forward into the three neurons in the second layer. The second layer has synapses going back to the first layer and forward to the motor neuron. The motor neuron is also connected back to the second layer. Finally all of the hidden neurons and the motor neuron are self connected. Therefore NM has 10 neurons and 46 synapses.

During evaluation, each sensor neuron received the raw distance value from its sensor. The hidden and motor neurons were updated using

$$n_i = \tanh\left(\sum_{n_j \in I_{n_i}} w_{ji} n_j\right) \tag{2.1}$$

where $I_{n_i}$ is the set of incoming synapses to neuron $n_i$ and $w_{ji}$ is the weight of the synapse from neuron $n_j$ to neuron $n_i$. The hyperbolic tangent function limits the hidden and motor neurons to floating point values in $[-1, +1]$.

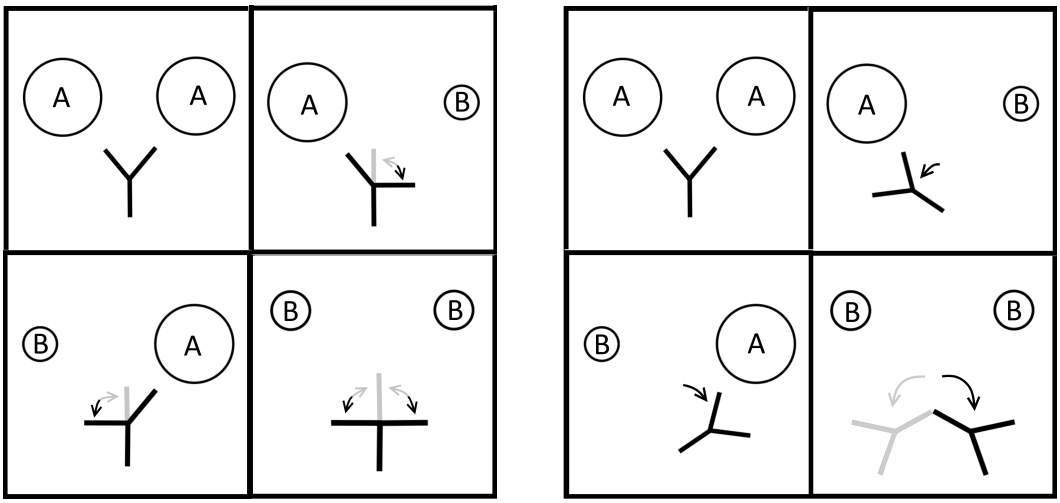
Movement was controlled using proportional difference control. The values output by the motor neurons were scaled to the range $[-45, +45]$ and treated as desired angles. The rotational velocity of a branch at each time step was thus determined by the difference between the desired angle determined by the value of the motor neuron in that branch (or at the root) and the current angle of that branch (or root).

### 2.3.3 The task environments.

The robots were evolved for a simple embodied categorization task: the robots were evolved to 'point at' Type **A** spheres and 'point away' from Type **B** spheres (Fig. 2.2). Each environment that a robot was placed in contained a pair of spheres. Following [74], this corresponds to two free parameters ($f = 2$): the object on the left and the object on the right.

Three environment spaces were considered.

The first was the simplest consisting of a $2 \times 2$ environment space, giving four separate environments (Fig. 2.3a). Each sphere could be Type **A** or Type **B** ($n = 2$). For this environment the type $A$ sphere had a radius of 3.5 and the type $B$ sphere




*(a) Desired modular morphology behaviors*

*(b) Desired modular non-morphology behaviors*

*Figure 2.2: Drawings of desired behavior for the modular morphology (2a) and non-modular morphology (2b) in each of the four environments in the $2 \times 2$ environment space. Arrows indicate desired movement away from the base position (the base position is shown in the top left panels). Gray segments and arrows indicate other acceptable behaviors.*

had a radius of 0.5.

The second environment space contained $3 \times 3$ environments, meaning nine total environments to consider (Fig. 2.3b). A sphere could be one of two instances of Type **A** (either $A$ or $a$) or Type **B**. For this environment space $A$ had a radius of 3.5, $a$ had a radius of 0.5, and $B$ was in the middle with a radius of 2.0. Thus, for this environment space, $n = 3$.

Finally, the last environment space considered contained $4 \times 4 = 16$ different environments (Fig. 2.3c). A sphere could be one of two instances of Type **A** ($A$ or $a$) or one of two instances of Type **B** ($B$ or $b$). For this environment space, spheres of type $A$, $B$, $a$, and $b$ had radii of 3.5, 2.5, 1.5, and 0.5, respectively. Therefore, $n = 4$ for this environment space.

Open Dynamics Engine was used to simulate the robots and the environment. A

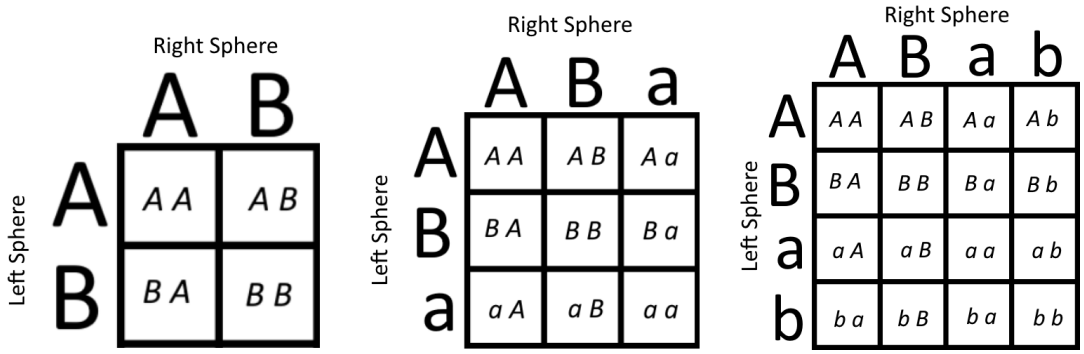



*Figure 2.3: The three environment spaces considered. $A, B, a, b$ represent spheres positioned on the left or right of the robot. Uppercase letters represent bigger spheres than their lowercase counterparts. Robots were evolved to 'point' at $\boldsymbol{A}$=$\{A, a\}$ spheres and away from $\boldsymbol{B}$=$\{B, b\}$ spheres.*

time step size of 0.05 was used.

### 2.3.4 Evolutionary optimization

The robots were trained using Age-Fitness Pareto Optimization (AFPO; [95]). AFPO is a multi-objective optimization algorithm which is designed to maintain diversity in an evolving population by periodically injecting new random individuals into the population and restricting the ability of older individuals to unfairly compete against younger individuals. In all of the experiments reported herein, a population size of 40 was employed.

Mutations in the population occurred in the form of choosing a new weight for a synapse from a normal distribution with mean of the current weight and a standard deviation proportional to the absolute value of the current weight. This mutation operator enables evolution to rapidly incorporate high magnitude weights if required, while also being able to fine tune weights with low magnitude. Mutation rates were

set to be the reciprocal of the number of synapses, thus yielding an average of one synapse change per mutation.

The optimization function used was an error function which averaged the error of the robot when exposed to each environment in the environment list $E_{\{\}}$:

$$\mathrm{Err}(E_{\{\}}) = \frac{1}{||E_{\{\}}||} \sum_{o_\ell o_r \in E_{\{\}}} \frac{e(o_\ell o_r) - e_{\min}(o_\ell o_r)}{e_{\max}(o_\ell o_r) - e_{\min}(o_\ell o_r)} \tag{2.2}$$

$$e(o_\ell o_r) = \frac{g(o_\ell) + g(o_r)}{2} \tag{2.3}$$

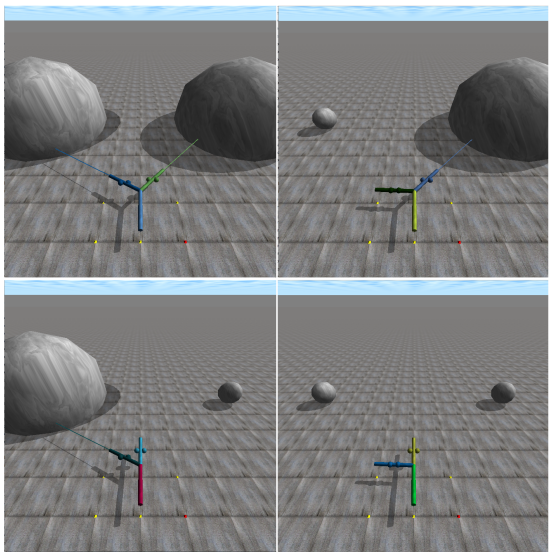
$$g(\mathbf{A}) = \begin{cases} 1, \text{ if } d(\mathbf{A}) > d_{\max}(\mathbf{A}) \\ \frac{d(\mathbf{A}) - d_{\min}(\mathbf{A})}{d_{\max}(\mathbf{A}) - d_{\min}(\mathbf{A})}, \text{ otherwise} \end{cases}, g(\mathbf{B}) = \begin{cases} 0, \text{ if } d(\mathbf{B}) > d_{\max}(\mathbf{B}) \\ \frac{d(\mathbf{B}) - d_{\min}(\mathbf{B})}{d_{\max}(\mathbf{B}) - d_{\min}(\mathbf{B})}, \text{ otherwise} \end{cases} \tag{2.4}$$

where

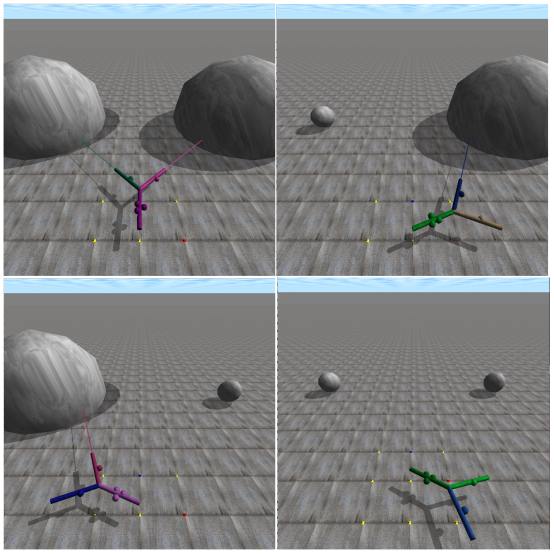
- $E_{\{\}}$ is a single environment, e.g. $(AA, bA, Bb$, etc.)
- $o_\ell \in \{\mathbf{A}, \mathbf{B}\}$ indicates the type of the object on the left. Either $(o_\ell = \mathbf{A})$ or $(o_\ell = \mathbf{B})$;
- $o_r \in \{\mathbf{A}, \mathbf{B}\}$ indicates the type of the object on the right. Either $(o_r = \mathbf{A})$ or $(o_r = \mathbf{B})$;
- $e(o_\ell o_r)$ indicates the robot's error incurred in environment $o_\ell o_r$ during the last time step;
- $e_{\min}(o_\ell o_r)$ and $e_{\max}(o_\ell o_r)$ indicate the minimum and maximum possible error the robot can incur in environment $o_\ell o_r$ during any one time step, respectively. These were calculated based on the environment present and the geometry of the robot;

- $g(o_\ell)$ and $g(o_r)$ denote the errors incurred as a result of the left-hand and right-hand objects, respectively;
- $g(\mathbf{A})$ and $g(\mathbf{B})$ denote the errors incurred as a result of the objects of each type.
- $d(\mathbf{A})$ and $d(\mathbf{B})$ denote the distances from the midpoint of the closest leaf to the center of the object considered.
- $d_{\max}(\mathbf{A}), d_{\min}(\mathbf{A}), d_{\max}(\mathbf{A}), d_{\min}(\mathbf{B})$ denote the maximum and minimum distance values for the **A** and **B** environments. Because the motion range of the modular and non-modular robots are inherently different, these values are necessarily different. Further, the $d_{\max}(\mathbf{A})$ and $d_{\max}(\mathbf{B})$ values could be set artificially lower than the actual maximums in order to create weighting which more heavily considered $g(\mathbf{A})$ term over $g(\mathbf{B})$. $d_{\min}(\mathbf{A})$ and $d_{\min}(\mathbf{B})$ represent the actual observable minimums depending on the geometry of the robot. The values are presented in Table 2.1.

For the modular morphologies, $d_{\max}$ was set to the actual limit of motion of the branch. For the non-modular morphologies, $d_{\max}$ was set to less than the actual range of the motor to produce the desired behavior. By setting $d_{\max}$ less than the actual range, any robot that goes past a certain distance away from the **A** sphere would have an error of 1 for that object. Similarly, in the **B** sphere, if the robot moved far enough away to be past $d_{max}$ it was considered to have 0 error. This effectively created a weighting to the influences between the **A** and **B** spheres which corresponded to the robot learning the desired behavior as seen in Figures 2.2 and 2.4.



*(a) An evolved controller for the M robot*



*(b) An evolved controller for the M robot*

*Figure 2.4: The behaviors generated by two controllers that evolved to succeed in each of the four environments in the 2x2 environment space. Lines emanating from the leaf branches represent the distance sensors embedded in each.*

| | Actual Max. Distance | $d_{\max}(B)$ | $d_{\max}(S)$ | $d_{\min}(B)$ | $d_{\min}(S)$ |
|---|---|---|---|---|---|
| M Morphology | 5.315 | 5.315 | 5.315 | 5.157 | 5.157 |
| NM Morphology | 7.596 | 6.002 | 7.200 | 5.028 | 5.028 |

*Table 2.1: Table of maximum and minimum distance values for each morphology type.*

### 2.3.5 Experimental Design

The first set of experiments consisted of evolutionary trials made up of fixed length epochs in the 2x2 environment space. The robot starts by training on one environment for the duration of the epoch. At the end of each epoch a new environment is added for the robot to be trained on. By the last epoch the robot is trained on all four environments. So, from the robots' perspective, when each new environment was introduced, the environment space changes by becoming more complex. The epoch length was set to 100 generations, thus each evolutionary run lasted for 400 generations. If a robot survived from the last generation of one epoch into the first generation of the next epoch, its fitness was recomputed against this expanded set of environments.

In the second set of experiments, the robots were evolved in a pre-determined subset of the environment space. Unlike the previous experiment, the robot is introduced to all of the environments in the subset at the same time instead of sequentially. After the best robot in the population achieved a pre-specified error threshold in all of the environments in the chosen subset, it was tested in the remaining environments not in the subset without any further evolution to see how well it performed.

| | $\{AA, AB, BA\}$ | $\{AA, AB, BB\}$ | $\{AB, BA, BB\}$ |
|---|---|---|---|
| NM | $BB$: 0.465 ($\pm$0.0445) | $BA$: 0.593 ($\pm$0.0385) | $AA$: 0.388 ($\pm$0.0232) |
| MNM | $BB$: 0.665 ($\pm$0.00614) | $BA$: 0.580 ($\pm$0.0168) | $AA$: 0.586 ($\pm$0.0269) |

*Table 2.2: Mean values of the error for the non-modular robot types in the unseen environment after achieving an error of at most 0.15 in the three seen environments. Values in the parenthesis represent one standard error of the mean.*

## 2.4 Results

Experiment 1, described in Section 2.3.5, was run 50 times for all three robots in four environments in the 2x2 environment space, yielding a total of $50 \times 2 = 100$ independent evolutionary runs. The order of the environments was $AA, BB, AB, BA$. Figure 4.3 shows that at the start of each epoch there is a spike in the error in the case of both the MNM and NM robot. In the case of the M robot, there is no spike in error when the third ($AB$) and fourth ($BA$) epochs are introduced.

Experiment 2, described in section 2.3.5, was also run 50 times for the 2x2, 3x3, and 4x4 environments on all of the robots. Thus there were $50 \times 2 \times 3 = 300$ independent trials. For the first set of trials only the 'diagonal' of the environment space was considered. For the 2x2 environment space this consisted of $\{AA, BB\}$. For the 3x3 environment space the diagonal was $\{AA, BB, aa\}$. Finally, for the 4x4 environment space the diagonal was $\{AA, BB, aa, bb\}$. The error threshold was set to 0.15. Figure 2.6 shows the results for these trials.

The next test using this experimental set-up considered another subset other than the diagonal which had the same number of elements as the diagonal. Specifically, the

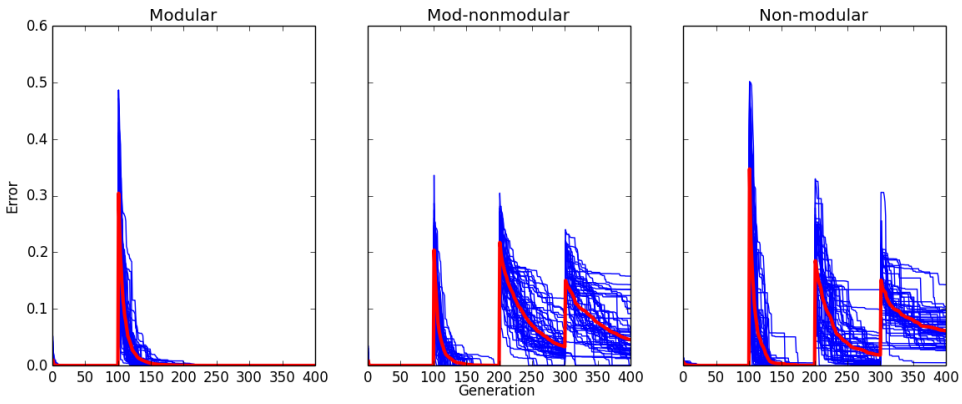



*Figure 2.5: Errors of controllers evolved for the M robot (left column), MNM robot (middle column), and the NM robot (right column) in fixed epoch training (experiment 1 as described in Sect. 2.3). New environment regimes occurred every 100 generations. Robots were evolved along the diagonal of the environment space meaning the order presented to the robot was $AA, BB, AB, BA$. Each blue curve corresponds to an individual evolutionary run: it reports, at each generation, the controller with the lowest error in the population at that time. The red curve reports the average of these runs.*

'corner' of the environment space was considered Fig. (2.7). All three environment spaces were considered. For the 2x2 environment space, the corner was designated to be the top row of the environment space $\{AA, AB\}$. For the 3x3 environment space, the corner was set as $\{AA, AB, BA\}$. Finally, for the 4x4 environment space, the corner was $\{AA, AB, BA, BB\}$. Fifty trials of each robot in each environment space were performed, yielding $50 \times 2 \times 3 = 300$ independent trials. Again the error threshold was set to 0.15.

The last test performed using this experimental set-up looked at how well the MNM and NM robots respond to an unseen environment in the 2x2 case when evolved in three out of the four environments. The robots were evolved in three different subsets: $\{AA, AB, BA\}, \{AA, AB, BB\}$, and $\{AB, BA, BB\}$. Because of the inherent symmetry in the problem, $\{AA, AB, BB\}$ is the same as $\{AA, BA, BB\}$, so only one

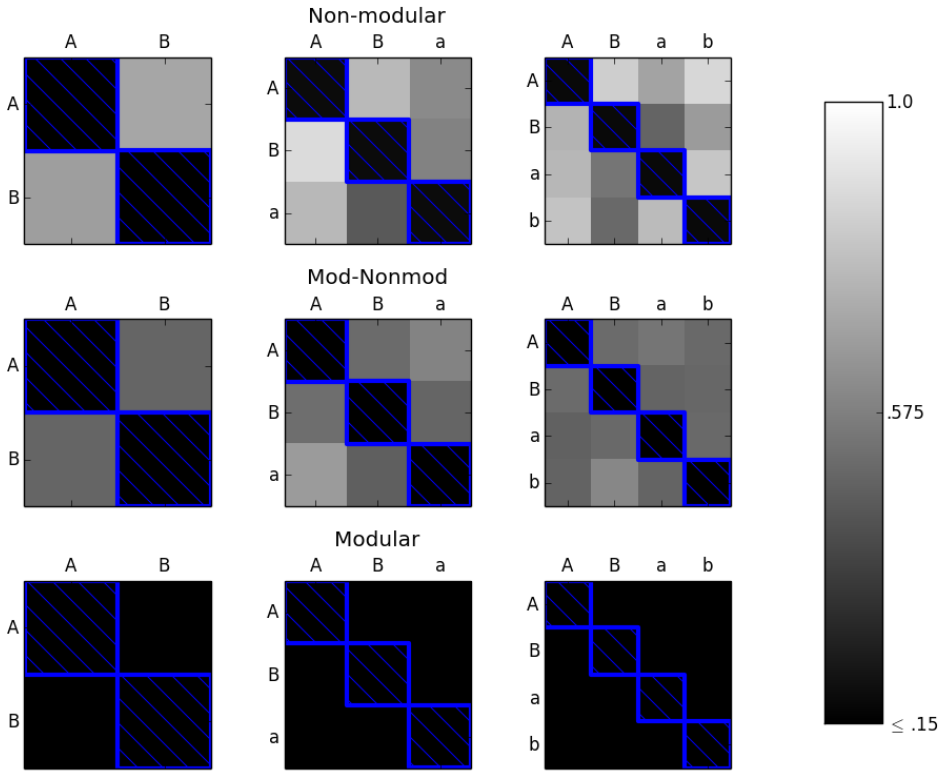



*Figure 2.6: The M (bottom row), MNM (middle row), and NM (top row) robots were evolved along the diagonal (environments with blue boxes around them) until they achieved an error of less than or equal to 0.15 in each environment in the subset considered. The robots were then tested in the remaining environments. The color of the box of each environment represents the average reported error in that environment. The lighter the color, the greater the average error with white representing an error of 1.0 and black representing an error of* $\leq 0.15$. *In the modular case, every robot achieved an error of less than or equal to 0.15 on the off diagonal environments. Over the 50 trials, both the non-modular and mod-nonmodular robot averaged an error greater than 0.15 in all of the off diagonal environments.*

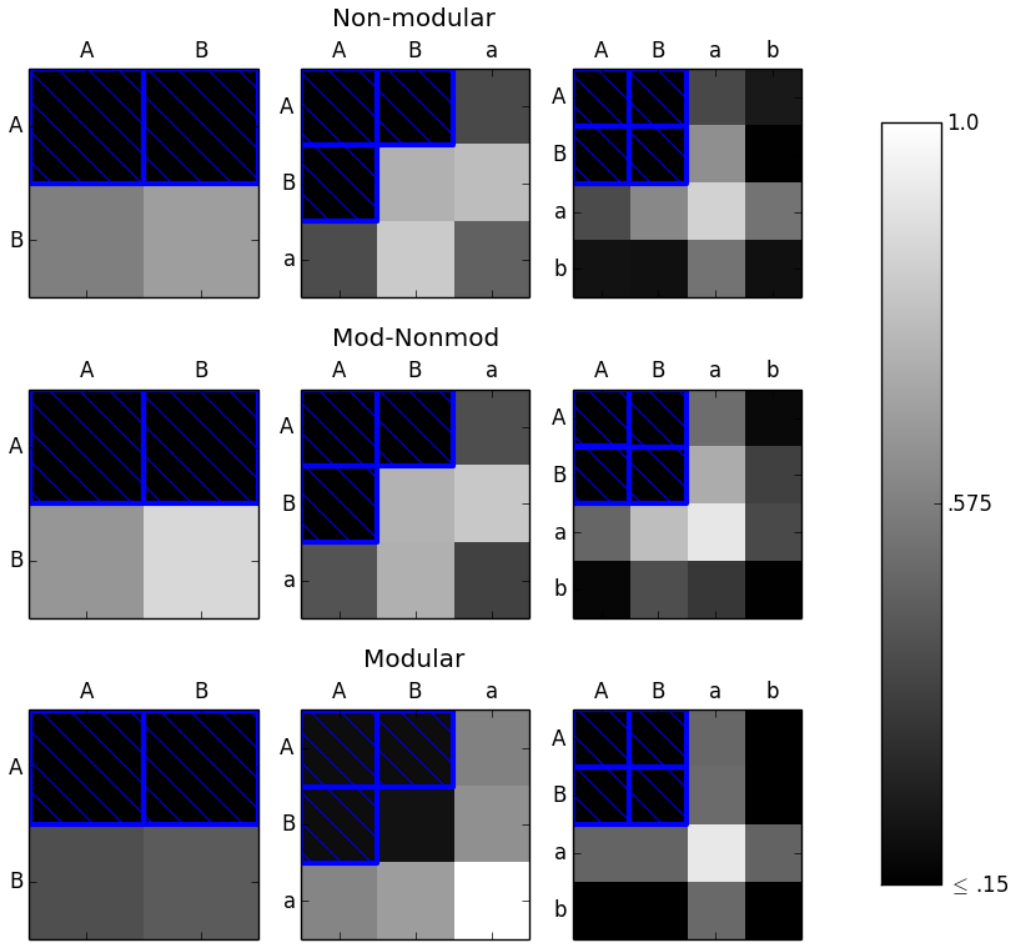



*Figure 2.7: The M (bottom row), MNM (middle row), and NM (top row) robots were evolved in the corners (environments with blue boxes around them) until they achieved an error of less than or equal to 0.15 in each environment in the subset considered. The robots were then tested in the remaining environments. The color of the box of each environment represents the average reported error in that environment. Here we see that it is necessary to use completely independent subsets of the environment space to ensure linear scaling in the modular case.*

was chosen to be tested. Results are presented in Table 2.2.

## 2.5 Discussion

When the modular robot is presented with a new environment, it is able to break down that environment into a combination of percepts. If the robot has seen those percepts before, even if the combination of those percepts is unfamiliar, it is able to act appropriately. Evidence for this is shown in Fig.4.3. There is no spike in error in the modular case at the start of the third and fourth epochs when the $AB$ and $BA$ environments are introduced. In contrast, the non-modular robots cannot see the environment in this manner, as is shown by the presence of error spikes at each new epoch.

Fig. 2.6 shows that when the modular robot is evolved along the diagonal of the environment space, it is able to achieve acceptable error levels, that is at or below the pre-determined cut off threshold (0.15), in the remaining environments in the environment space. This suggests that, for this specific task, the number of environments needed to evolve a robot with a modular morphology and controller scales with the size of the diagonal of the environment space. Therefore the necessary number of environments for the modular robot seems to scale linearly with $n$, where $n$ is equal here to the number of variations in the size of the spheres.

Conversely, the robots with the non-modular morphologies or controllers do not achieve acceptable, at or below 0.15, errors in the other environments in the space by simply evolving along the diagonal, as seen in Fig. 2.6. This means that for this task, the number of environments the robot needs to be evolved in before achiev-

ing adequate fitness for the whole environment space is greater than the number of environments along the diagonal.

Table 2.2 shows that even when either of the non-modular robots is presented with three out of the four environments in the 2x2 environment space they cannot use what it has seen in previous environments to help them in the unseen environment. Thus, at least for the 2x2 environment space case, the non-modular robots need to be evolved in each environment in the entire space in order to achieve adequate fitness.

Figure 2.7 indicates that just choosing any subset of environments to evolve in does not guarantee adequate fitness in the remaining unseen environments. Specifically, the results point to choosing a subset of environments in which each environment is completely independent from every other environment in the subset. In this context, completely independent environments are those which do not share the same row or column. For example, $AB$ would be completely independent from $aa$ since both the right ($A \neq a$) and left ($B \neq a$) spheres are different. As a converse example, $AB$ and $Aa$ are not completely independent since the left sphere is the same in both environments, namely $A$. These results further suggest that a modular robot can recognize familiar precepts from previous environments and respond appropriately to them, even when they are presented in an unfamiliar combination. This is seen in the result from Fig. 2.7, which shows that in the 3x3 environment space case, when the robot is tested in the $BB$ environment, it reacts appropriately without requiring further evolution.

Figure 2.7 also shows the side result that evolution will generally find the simplest action to solve the problem at hand. In the 4x4 environment space case both the modular and non-modular robot evolve to act on any sphere of size $B$ or smaller (the

$a$ or $b$ sizes) as an instance of the $B$ sphere. Thus, the robots do well in the remaining environments comprised of $b$ spheres and poorly in the environments containing $a$ spheres since the action desired for $B$ sizes is the same as $b$ and different than the action desired for $a$.

## 2.6 Conclusions

This paper has shown that a modular morphology combined with a modular neural control can enable a robot to break down seemingly novel environments into combinations of familiar percepts. Moreover, if robots possess both this morphological and neural modularity, these robots are also likely to move in a similar manner in these environments, thus continuing to perceive the environment as a combination of familiar percepts. Assuming that the robot should always react the same way to each of these local percepts, it follows then that such a robot is likely to exhibit a successful behavior in this novel environment without requiring further training.

Robots with either non-modular morphologies or non-modular neural controllers cannot easily exhibit this phenomenon and, as a result, are likely to require additional training even in environments that contain individually familiar percepts. Given this, we have shown that, for this task, robots with a modular morphology, combined with a modular neural controller, need to be evolved only in a linearly growing number of environments, whereas the number of environments non-modular robots require grows superlinearly. Our results indicate that it is likely that non-modular robots will require evolution in all of the possible environments in the space.

In future work, we would like to investigate specifically how the amount of evolu-

tionary time necessary to evolve adequately fit robots scales for both the modular and non-modular robots. We plan to accomplish by completely evolving both the modular and non-modular robots in the 2x2, 3x3, and 4x4 environment spaces. Further, we will look into scaling both $f$ and $n$ instead of just $n$, as was presented in this work.

If we consider our entire environment space to be a hypercube composed of $n^f$ hypervoxels representing each individual environment, then there will be $n$ voxels along the diagonal of the hypercube. If it is sufficient for a modular robot to simply evolve along this diagonal, then it is possible for time complexity, in this case the number of evolutionary time steps, necessary to evolve a given robot in an $n^f$ sized environment space to decrease from $O(n^f)$ to $O(n)$. However, this ideal case holds only if the robots are already morphologically and neurologically modular.

If robots begin with little or no morphological or neural modularity, it follows from [57] that if environments are added in a modularly-varying way, more modular robots should evolve. This can be accomplished in this framework by ensuring that each newly-added environment contains just one new feature of one of the free parameters describing the environments, while the other free parameters hold to a feature against which the robots have already been trained. This would require environments to be added to the training set along each of the edges of the environment hypercube in sequence, thus reducing $O(n^f)$ to $O(nf)$. Determining whether this theoretical result holds in practice, and under what conditions, is another worthy target of future investigation.

There are many other problems to investigate, including how these results here can be generalized to more complex and realistic robots and task environments. Furthermore, under what conditions would the evolved modularity be maintained when

the evolved robots are instantiated as physical robots.

Ultimately, this work thus suggests that there may exist a relationship between morphology, modularity, evolvability, and scalability, which may in future enable the automated optimization of increasingly complex robots that perform appropriately in increasingly complex environments.

# References for chapter 2


[4] Anton Bernatskiy and Joshua C Bongard. "Exploiting the relationship between structural modularity and sparsity for faster network evolution". In: *Proceedings of the companion publication of the 2015 annual conference on genetic and evolutionary computation*. ACM. 2015, pp. 1173–1176.

[10] Josh C Bongard. "Spontaneous evolution of structural modularity in robot neural network controllers". In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM. 2011, pp. 251–258.

[13] Josh C Bongard et al. "Evolving robot morphology facilitates the evolution of neural modularity and evolvability". In: *Proceedings of the 2015 annual conference on genetic and evolutionary computation*. ACM. 2015, pp. 129–136.

[14] Rodney A Brooks. "Elephants don't play chess". In: *Robotics and autonomous systems* 6.1 (1990), pp. 3–15.

[21] Andy Clark. *Being there: Putting brain, body, and world together again*. MIT press, 1998.

[23] Jeff Clune, Jean-Baptiste Mouret, and Hod Lipson. "The evolutionary origins of modularity". In: *Proc. R. Soc. B* 280.1755 (2013), p. 20122863.

[35] Kai Olav Ellefsen, Jean-Baptiste Mouret, and Jeff Clune. "Neural modularity helps organisms evolve to learn new skills without forgetting old skills". In: *PLoS Comput Biol* 11.4 (2015), e1004128.

[38] Carlos Espinosa-Soto and Andreas Wagner. “Specialization can drive the evolution of modularity”. In: *PLoS computational biology* 6.3 (2010), e1000719.

[40] Robert Fitch et al. “Reconfigurable modular robotics”. In: *Robotics and Autonomous Systems* 7.62 (2014), pp. 943–944.

[44] Robert M French. “Catastrophic forgetting in connectionist networks”. In: *Trends in cognitive sciences* 3.4 (1999), pp. 128–135.

[57] Nadav Kashtan and Uri Alon. “Spontaneous evolution of modularity and network motifs”. In: *Proceedings of the National Academy of Sciences* 102.39 (2005), pp. 13773–13778.

[60] Kostas Kouvaris et al. “How Evolution Learns to Generalise: Principles of under-fitting, over-fitting and induction in the evolution of developmental organisation”. In: *arXiv preprint arXiv:1508.06854* (2015).

[72] Hod Lipson et al. “On the origin of modular variation”. In: *Evolution* 56.8 (2002), pp. 1549–1556.

[74] Maja Matarić and Dave Cliff. “Challenges in evolving controllers for physical robots”. In: *Robotics and autonomous systems* 19.1 (1996), pp. 67–83.

[79] Mark EJ Newman. “Modularity and community structure in networks”. In: *Proceedings of the national academy of sciences* 103.23 (2006), pp. 8577–8582.

[86] Rolf Pfeifer and Josh Bongard. *How the body shapes the way we think: a new view of intelligence*. MIT press, 2006.

[88] Tony Pinville et al. “How to promote generalisation in evolutionary robotics: The ProGAb approach”. In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM. 2011, pp. 259–266.

[95] Michael Schmidt and Hod Lipson. “Age-fitness pareto optimization”. In: *Genetic Programming Theory and Practice VIII*. Springer, 2011, pp. 129–146.

[109] Günter P Wagner. “Homologues, natural kinds and the evolution of modularity”. In: *American Zoologist* 36.1 (1996), pp. 36–43.

[113] Mark Yim et al. “Modular self-reconfigurable robot systems [grand challenges of robotics]”. In: *Robotics & Automation Magazine, IEEE* 14.1 (2007), pp. 43–52.

# Chapter 3

# Reducing Training Environments Through Ecological Modularity

## 3.1 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.

## 3.2 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 [44], 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. [74] 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 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 $n = 2$. [74] 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.

### 3.2.1 Robustness

Much work has been done to increase the robustness of evolved behavior in robots. For instance, Jakobi [54] investigated the introduction of noise to guard against evolutionary exploitation of any inaccuracies in the simulator used to the evolve the behaviors. Lehman [68] 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 [8] 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.* [35], 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 [24], and, in [35], this neural modularity enabled evolved networks to rapidly adapt to new environments without losing their ability to succeed in the original environments.

### 3.2.2 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 [108] 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 [72], neural networks [56], [24], and models of gene networks [36].

Investigations into the evolution of modularity in embodied systems begin with Gruau [48], who employed an indirect genotype to phenotype mapping that allowed for the construction of neural modules in a robot. Yamashita et al. [112] demonstrated robots capable of learning independent motor primitives and then combining them in novel sequences. In [11] and [13], 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.

### 3.2.3 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 [35]. Catastrophic forgetting is a problem when, in order to learn a new task, an agent must forget what it previously learned [44].

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 [80]. $Q$ 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 [17]. 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'.

### 3.2.4 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 $F \setminus U$ 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.

## 3.3 Methods

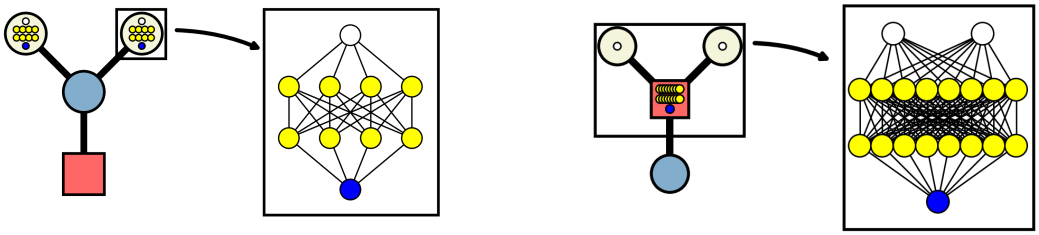
In this section we describe the structure of the environment spaces, robot design, evolutionary algorithm, and experimental design.

### 3.3.1 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 $+45^o$ and $-45^o$ with respect to the root node. In this paper we explored two variations of robot morphology.



(a) $\mathcal{M}$

(b) $\mathcal{NM}$

*Figure 3.1: The modular (3.1a) and non-modular (3.1b) 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.*

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 $[-45^o, +45^o]$ 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 $[-120^o, +120^o]$.

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 Figure 3.1. 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. 3.1a). 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. 3.1b).

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

$$y_i^{(t)} = \tanh\left(y_i^{(t-1)} + \sum_{j \in J} w_{ji} y_j^{(t-1)}\right) \tag{3.1}$$

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 $i$.

### 3.3.2 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. 3.2. Cylinders were organized on a line segment perpendicular to the direction of the leaf nodes. Clusters were placed such that the

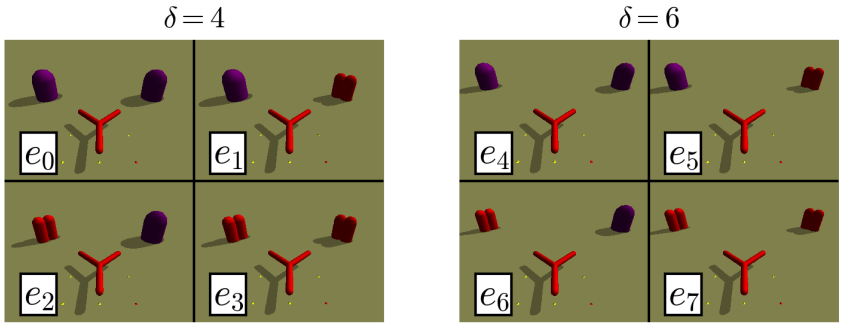



*Figure 3.2: The starting point of the robots in simulation for each environment. The environment space $E_2 = \{e_0, e_1, e_2, e_3\}$ is shown by the four left environments in the figure and $E_3 = \{e_0, e_1, \ldots, e_7\}$ is shown by all eight environments which make up the figure. The $\delta$ variable defines the initial distance of both clusters from the robot.*

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 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. $c_R \in \{1, 2\}$.
- $\delta$: 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 $(\delta, c_L, c_R)$.

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:

*Figure 3.3: Physical robot made out of Legos. Can represent the $\mathcal{M}$ or $\mathcal{NM}$ robot by fixing/freeing motors.*

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

where $*$ indicates that parameter is free to vary. From this we see $E_2$ is a $2 \times 2$ 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 Figure 3.2.

### 3.3.3 Physical implementation

The robot was also made in out of Legos as shown in Figure 3.3. 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.

### 3.3.4 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 equation (3.4).

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. 3.3).

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. 3.2). 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.

$$\text{Overall Fitness} = \sum_{i \in ||O||} w_i \ \text{fit}(e_{(i)}) \tag{3.2}$$

$$\text{fit}(e_i) = \frac{1}{\text{normalize}(e_i)} \sum_{t=T/2}^{T} s_L(t) + s_R(t) \tag{3.3}$$

$$s_{\{L,R\}}(t) = \begin{cases} 1 & \text{if the sensor is pointing at} \\ & \quad \text{an even cluster at time } t \\ 0 & \text{if the sensor is not pointing at} \\ & \quad \text{an object at time } t \\ -1 & \text{if the sensor is pointing at} \\ & \quad \text{an odd cluster at time } t \end{cases} \tag{3.4}$$

Evolution was performed using Age Fitness Pareto Optimization (AFPO) with a population size of 50 [95]. 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.

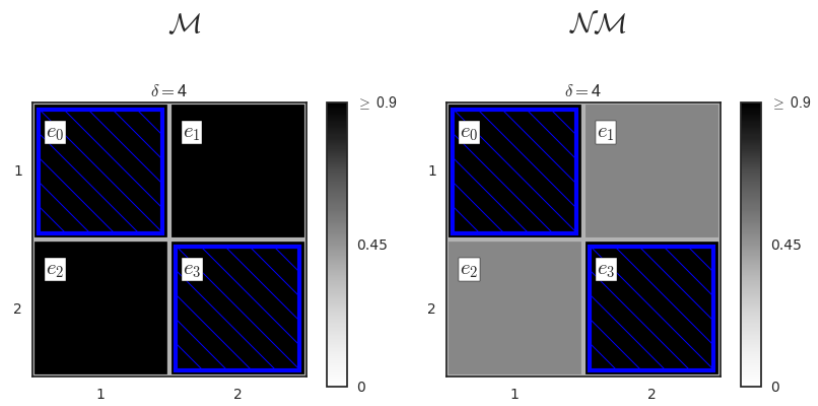
### 3.3.5 Experimental setup

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.

## 3.4 Results

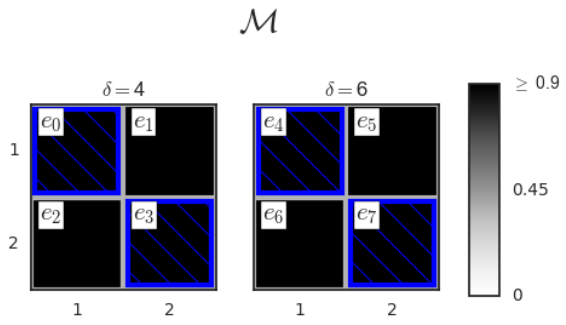
The first environment space explored was $E_2$, the $2 \times 2$ environment space where only $c_L$ and $c_R$ were varied. The training set of the robots was $O_{2,2} = \{e_0, e_3\}$. In $E_2$, $O_{2,2}$ represents the grand diagonal of the space. Figure 3.4a shows that $\mathcal{M}$ was able to achieve sufficient fitness in the entirety of $E_2$ when the robot achieved sufficient fitness in $O_{2,2}$. Figure 3.4b 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.

The second environment space we explored was $E_3$, the $2 \times 2 \times 2$ environment space where $c_L, c_R$ and $\delta$ 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 $\delta$. 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. 3.5a). 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. 3.5b). 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. 3.6 show the $\mathcal{M}$ was not able to be sufficiently fit.

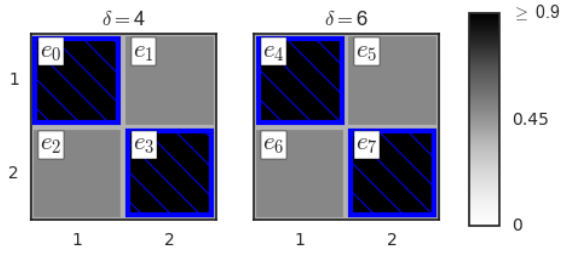



*(a) Modular robot in $E_2$* *(b) Non-modular robot in $E_2$*

*Figure 3.4: Average fitness scores for $\mathcal{M}$ (3.4a) and $\mathcal{NM}$ (3.4b) 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.*




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




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

*Figure 3.5: Average fitness scores for $\mathcal{M}$ (3.5a) and $\mathcal{NM}$ (3.5b) 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.*

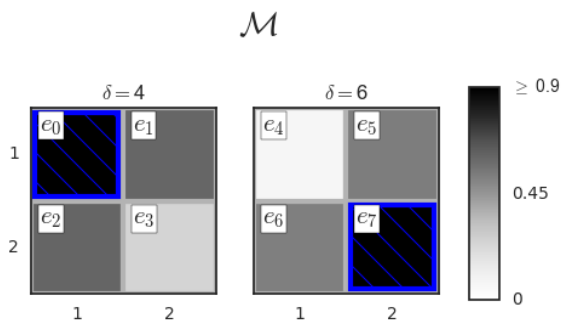
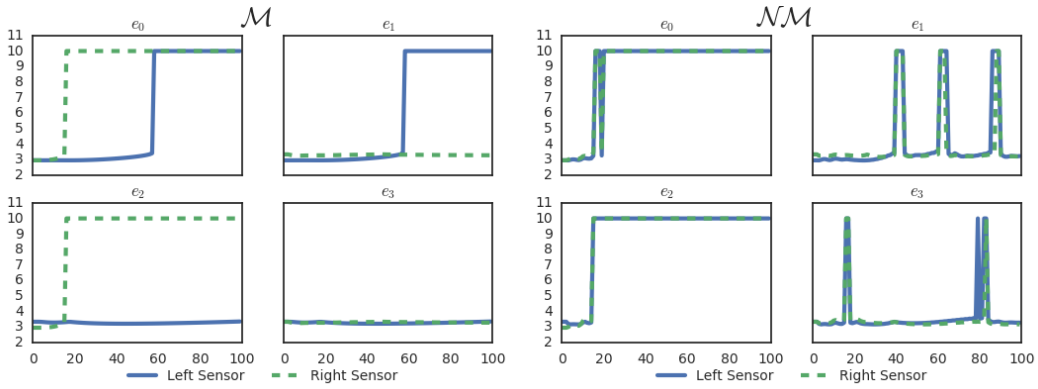



*Figure 3.6: The $\mathcal{M}$ robot evolved with training set $O_{3,2} = \{e_0, e_7\}$. We see that the robot does not achieve adequate fitness when only evolved in $O_{3,2}$.*




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

*Figure 3.7: 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.*

## 3.5 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 3.4 and 3.5.

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. 3.7a ). The $\mathcal{NM}$ robot cannot break down its environment into left and right percepts because it is morphologically and neurologically non-modular (Fig. 3.7b). 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 $\delta$. 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. 3.5 and 3.6). 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.

# References for chapter 3


[8] Josh Bongard. "Morphological change in machines accelerates the evolution of robust behavior". In: *Proceedings of the National Academy of Sciences* 108.4 (2011), pp. 1234–1239.

[11] Josh C Bongard. "Spontaneous evolution of structural modularity in robot neural network controllers". In: *Proceedings of the 2011 Genetic and Evolutionary Computation Conference*. Dublin: ACM, 2011, pp. 251–258.

[13] Josh C Bongard et al. "Evolving robot morphology facilitates the evolution of neural modularity and evolvability". In: *Proceedings of the 2015 annual*

*conference on genetic and evolutionary computation*. ACM. 2015, pp. 129–136.

[17] Collin K Cappelle et al. "Morphological modularity can enable the evolution of robot behavior to scale linearly with the number of environmental features". In: *Frontiers in Robotics and AI* 3 (2016), p. 59.

[24] Jeff Clune, Jean-Baptiste Mouret, and Hod Lipson. "The evolutionary origins of modularity". In: *Procs of the Royal Society B: Biological sciences* 280.1755 (2013), p. 20122863.

[35] Kai Olav Ellefsen, Jean-Baptiste Mouret, and Jeff Clune. "Neural modularity helps organisms evolve to learn new skills without forgetting old skills". In: *PLoS Comput Biol* 11.4 (2015), e1004128.

[36] C. Espinosa-Soto and A. Wagner. "Specialization can drive the evolution of modularity". In: *PLoS Comp Biol* 6.3 (2010), e 1000719.

[44] Robert M French. "Catastrophic forgetting in connectionist networks". In: *Trends in cognitive sciences* 3.4 (1999), pp. 128–135.

[48] F. Gruau. "Automatic definition of modular neural networks". In: *Adaptive Behaviour* 3 (1994), pp. 151–183.

[54] Nick Jakobi, Phil Husbands, and Inman Harvey. "Noise and the reality gap: The use of simulation in evolutionary robotics". In: *Advances in artificial life*. Springer, 1995, pp. 704–720. DOI: 10.1007/3-540-59496-5_337.

[56] N. Kashtan and U. Alon. "Spontaneous evolution of modularity and network motifs". In: *PNAS* 102.39 (2005), p. 13773.

[68] Joel Lehman et al. “Encouraging Reactivity to Create Robust Machines”. In: *Adaptive Behavior* (2013).

[72] Hod Lipson et al. “On the origin of modular variation”. In: *Evolution* 56.8 (2002), pp. 1549–1556.

[74] Maja Matarić and Dave Cliff. “Challenges in evolving controllers for physical robots”. In: *Robotics and autonomous systems* 19.1 (1996), pp. 67–83.

[80] Mark EJ Newman and Michelle Girvan. “Finding and evaluating community structure in networks”. In: *Physical review E* 69.2 (2004), p. 026113.

[95] Michael Schmidt and Hod Lipson. “Age-fitness pareto optimization”. In: *Genetic Programming Theory and Practice VIII*. Springer, 2011, pp. 129–146.

[108] G.P. Wagner, M. Pavlicev, and J.M. Cheverud. “The road to modularity”. In: *Nature Reviews Genetics* 8.12 (2007), pp. 921–931. ISSN: 1471-0056.

[112] Yuichi Yamashita and Jun Tani. “Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment”. In: *PLoS Comp Bio* 4.11 (2008), e1000220.

# Chapter 4

# Embodied Embeddings for HyperNEAT

## 4.1 Abstract

A long time goal of evolutionary roboticists is to create ever-increasing lifelike robots which reflect the important aspects of biology in their behavior and form. One way to create such creatures is to use evolutionary algorithms and genotype to phenotype maps which act as proxies for biological development. One such algorithm is HyperNEAT whose use of a substrate which can be viewed as an abstraction of spatial development used by Hox genes. Previous work has looked into answering what effect changing the embedding has on HyperNEAT's efficiency, however no work has been done on the effect of representing different aspects of the agents morphology within the embeddings. We introduce the term *embodied embeddings* to capture the idea of using information from the morphology to dictate the locations of neurons in the substrate. We further compare three embodied embeddings, one which uses

the physical structure of the robot and two which use abstract information about the robot's morphology, on an embodied version of the retina task which can be made modular, hierarchical, or a combination of both.

## 4.2 Introduction

The ultimate goal of evolutionary artificial intelligence and automated machine research is to have the complex forms and behaviors of animals reflected in the agents created by algorithms. Modularity, hierarchy, and regularity are important factors to consider when creating agents which reflect the complexities seen in biology [19, 94, 49, 12].

In mammalian fetal development, HOX genes are activated at different times in development based on the strength of chemical gradients physically present in the morphology of the organism [33, 63]. In one example, Hox genes at the beginning of the genome are activated first near the anterior end of the organism while Hox genes at the end of the genome are activated later in development along the posterior of the organism. This allows development to easily generate symmetry and structure based on a local interaction of chemicals. This type of development helps generate the repeated, hierarchical, and modular structures present in a biological agents body.

Indirect encodings which incorporate development in some manner can help artificially act as development similar to how Hox genes behave during mammalian development[101, 7]. Indirect encodings also have the benefit of only needing to optimize a set of parameters existing in a smaller dimension than a counterpart direct encoding. Instead of choosing the weight of each synapse, an indirect encoding instead

provides rules or functions for how the weights should be set [47]. Evolution then only has to optimize the rules for determining the weights and not the weights themselves meaning indirect encodings can be applied to arbitrarily tasks with arbitrarily large networks without increasing the search space evolution actually exists in. Indirect encodings can also introduce regularity into the phenotype of the agent, similar to what is found in natural agents [28]. If the rules used by the indirect encoding contain symmetry, it is likely symmetry will be reflected in the final individual. In the effort to create more natural evolved robots being able to codify natural development processes in the generation of said robots is an important step.

HyperNEAT, explained in more detail in the next section, is a direct encoding which acts as a proxy for development for artificial agents. Similar to Hox genes, HyperNEAT takes locality into account in its genotype to phenotype map. We call an *embodied embedding* when the locality present in the substrate is based on some aspect of the morphology of the agent. We provide evidence that different embodied embeddings provide differences based on the objective the robot is tasked to perform.

### 4.2.1 HyperNEAT

HyperNEAT is a genetic algorithm specialized in creating gradients and patterns along substrates [101]. When used on neural networks, Hyperneat uses Compositional Pattern Producing Networks (CPPNs) to determine the synaptic weight between two neurons by taking in the embedded locations of the source and target neuron as input. The output of the CPPN is then used to determine the weight of the synapse. HyperNEAT uses the NEAT algorithm to determine the structure and topology of the CPPN. The initial population for NEAT is a simplistic network directly connecting

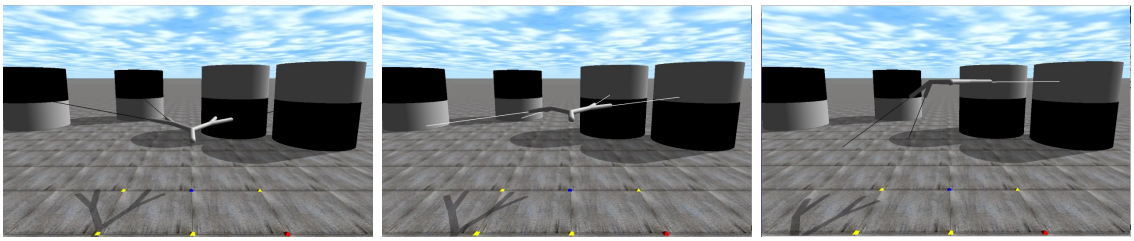


*Figure 4.1: A depth 2 robot in an environment consisting of two Far cylinders on the left and two Near cylinders on the right. The robot is tasked with a local and a global objective. The local objective consists of pointing at the white portions of the cylinders while the global objective consists of moving its root node upwards because there are an even number of each type of cylinder. The initial position of the robot is shown on the left. At the midpoint of evaluation (middle image) the robot is correctly pointing at the correct portions of the cylinders. Rays coming out of the robots leaves are purely graphical to help indicate where the robot is pointing. Towards the end of simulation (right) the robot has completed the global task at the cost of half of the local task.*

inputs to outputs with randomly determined weights. Over evolutionary time the network is made more complex by adding nodes and edges to the network, as well as changing the weights of the edges. The activation functions of the nodes are chosen from a predetermined list of functions which can be chosen to reflect the relevant regularity needed in the specific task at hand. NEAT further uses diversity preservation techniques which give new genotypes a chance to optimize to the task before removing them from the population due to fitness.

HyperNEAT has been shown to be effective in a wide range of domains which require regularity from determining weights in neural networks, to painting images, and determining an agent's morphology [30].

There are many variants and modifications one can make to HyperNEAT. The one we use throughout the experiments in this paper is HyperNEAT with Link Expression Output (HyperNEAT-LEO). HyperNEAT-LEO uses two outputs to the CPPN, one to determine the weight of the synapse in question and one to determine whether the

synapse is expressed or not in the final neural network [106]. By allowing evolution to control whether the synapse is expressed, HyperNEAT-LEO can control the final topology of the produced neural network and explicitly dictate the presence of modules. To further imbue the concept of modularity and locality for HyperNEAT-LEO, the initial population of CPPNs can be seeded with Gaussian nodes which are activated only when input neurons are close to eachother in their embedded substrate. Thus synapses are initially much more likely to be expressed if the neurons are close in embedded space.

This type of control over synaptic expression can be useful in tasks in which modularity is necessary [106]. One such task is the Retina Task detailed later in the paper. So far the majority of tasks which use HyperNEAT-LEO have been disembodied networks in which modularity is critical or necessary to correct completion of the task [106, 53]. In this work we apply HyperNEAT-LEO to an embodied agent evolved in task environments that are modular and hierarchical.

### 4.2.2 Substrate Analysis

[25] showed that the configuration of the substrate can impact fitness and efficiency when using HyperNEAT. The authors applied different embodied embeddings for a quadruped. In one experiment, the embeddings differed by the dimension of the geometric representation. The authors tested 1,2, and 3-d representations for the neural network controlling the quadruped, the idea being that the 3-d representation more encompasses the actual symmetries and structure present in the physical robot. They showed that the 2 and 3-d representations resulted in similar performance meaning that this increase in dimensionality did not help or hinder HyperNEAT's ability

to evolve a walking gait in the quadruped. Their work did not use information about the abstract structure and relationships between components in the morphology in determining the embedding.

Further work in the impact on of embedding locations has been focused on evolvable-substrate HyperNEAT (ES-HyperNEAT) which evolves the location of neurons in the substrate as well as the CPPN [92, 93]. While these methods have shown to have been effective on benchmark problems, when the designer of the substrate is given a body in which to physically place neurons, it allows the morphology to dictate the structure of the controller.

### 4.2.3 Retina Task

The retina task is an inherently hierarchical and modular task. Two retinas, on the left and right, are fed into a neural network. The modular aspect of the retina task is to distinguish whether each retina contains a target pattern or not. The hierarchical aspect of the retina task is to then take whether each retina is a target pattern or not as a logical input and compute a function on that input, like NAND.

The original goal of the retina task was to show that modularly varying goals causes evolution to generate modular networks whereas fixed goals tend to generate nonmodular networks [57]. In their initial paper [57] used direct encodings to construct the neural networks topology and synaptic weights. By changing the logical function the network needed to compute periodically, they were able to generate networks which exhibited left-right modularity.

[27] then used the same retina task using HyperNEAT to specify the neural network. They found that HyperNEAT performed much more poorly than the direct

encodings used previously as well as finding that the solutions produced by HyperNEAT were not modular.

In response, [106] showed that by using Hyperneat-LEO modular solutions were found to the retina task when the initial population of Central Pattern Producing Networks (CPPNs) used by HyperNEAT-LEO were seeded with an explicit concept of locality. This work further showed that other versions of HyperNEAT including dynamic threshold and LEO without seeding were less effective in generating modular networks which were fit to the required task.

Further modularity has been shown to be evolved by using connection cost along with HyperNEAT-LEO [53]. The authors evolved networks on a variety of variants of the disembodied retina task.

Every work using the retina task has done so in a disembodied way. We present an embodied version of the retina task in which the robot must physically move in order to respond correctly to what it senses in the environment. This movement then changes how the robot perceives its environment causing its sensation of the environment to change. In this manner seemingly modular tasks can have extremely effective non-modular solutions. In future work we plan to examine how adding connection cost can aid modularity in the embodied retina task presented later in the paper.

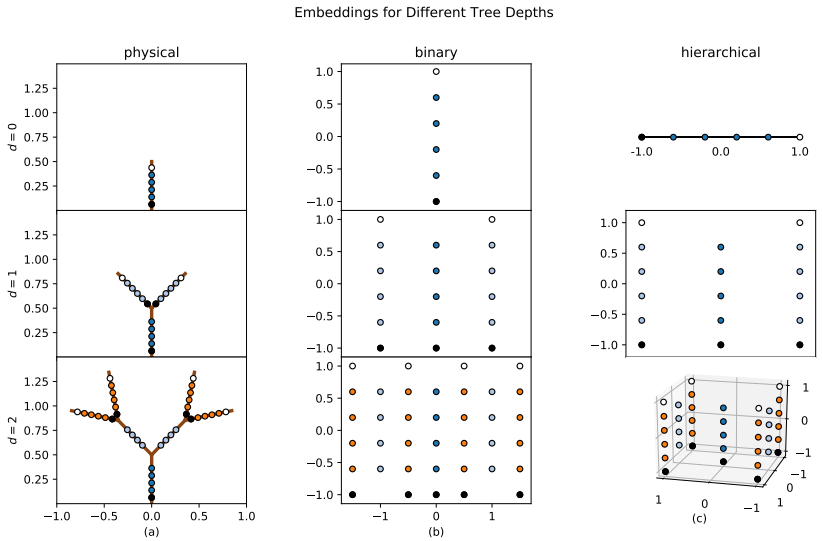



*Figure 4.2: The embeddings shown at three different depths of the tree. White neurons indicate the distance sensor neurons. Sensor neurons are placed in the leaves of the tree and point outward in the direction of the leaf they are contained in. Black neurons indicate motor neurons and are placed in the leaves and the root of the tree. The remaining neurons are hidden neurons colored according to their depth for ease of comparison across embeddings. Each branch consisted of four hidden neurons. Both the physical* ***(a)*** *and binary* ***(b)*** *embeddings exist in 2D-space regardless of the depth of the tree and produce valid embeddings (i.e. no overlapping neurons) for arbitrary depths of the tree. The dimension of the hierarchical embedding* ***(c)*** *grows as the size of the tree grows. Note at $d = 1$ the hierarchical and binary embedding are exactly the same.*

## 4.3 Methods

### 4.3.1 Robot Construction

The robot was a planar tree structure consisting of an actuated root and $2^d$ actuated leaves where $d$ is the depth of the tree. Hence, at $d = 2$ there are two actuated leaves (Fig. 4.2). Each leaf consisted of a distance sensor pointing out from the tree into the environment and a motor which actuates a hinge joint connecting the leaf

to its parent branch. The hinge joint allowed the leaf to rotate up and down with respect to its parent branch. The root consisted of a motor which actuated a linear joint moving the entire tree along the z-axis (up and down). Joint ranges in the leaves were limited to $\pm\frac{\pi}{4}$ from their starting position and the root node could move $\pm 1$ units from its starting position. Each branch was 1/2 unit long with the root base starting at the point $(0, 0, 1.5)$

$$n_i^{(t)} = \sigma\left(n_i^{(t-1)} + \sum_{j \in J} w_{ji} n_j^{(t-1)}\right) \tag{4.1}$$

Neuron activation in the robot was controlled using Equation 4.1. The value of the $i^{\text{th}}$ neuron, $n_i$, at time step $t$ was equal to the value of $n_i$ in the previous time step plus the sum of the incoming synaptic weights multiplied by the corresponding neuron's value. The $\sigma$ in the Eq. (4.1) is the hyperbolic tangent function.

### 4.3.2 Embodied Retina Task

The goal of the embodied retina task is to perceive objects in the environment and react accordingly. Similar to the its disembodied counterpart the embodied retina task requires aspects of modularity and hierarchy in the controller of the agent.

The task environment consisted of cylinders placed along a semi circle four units (Near) and six units (Far) away from the origin. Near cylinders were white on top, black on the bottom and far cylinders were colored black on top and white on bottom. The cylinders were placed such that each leaf of the robot was pointing at the middle of its corresponding cylinder (Fig. 4.1).

There were two objectives for each robot: local and global. The local objective was

to have leaves point at the white region of their corresponding cylinders. The global objective was to determine if there was an even or odd number of Near cylinders. The robot had to respond by moving the root, and thus the whole tree, up if there was an even number and down if there was an odd number of Near cylinders.

For example, given a depth $d = 1$ robot and the the environment {Near, Near}, the robot should move its root up while the leaves of the robot point up towards the tops of the cylinders. Further, given the environment {Near, Far}, the global solution will be to move the root down, while the local solutions will be to point to the top of the left (Near) cylinder and the bottom of the right (Far) cylinder.

The number of cylinders in the environment is dependent on the number of leaves in the tree which is further dependent on the depth of the tree. Specifically, the number of cylinders, $n$, is $n = 2^d$. Each cylinder had two variants. Thus, at depth $d = 1$ there are two cylinders meaning there are $2^2 = 4$ total environments for the robot to be evaluated in. For depth $d = 2$ there are four cylinders giving $4^2 = 16$ total environments for the robot to be evaluated in.

$$\text{eval}(c_e, t) = \begin{cases} 1 & \text{if pointing at white region of} \\ & \quad c_e \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$$

$$g_e = |z_{\text{target}} - z_{\text{root}}| \tag{4.2}$$

$$\ell_e = 1 - \frac{\sum_{t=T/2}^{T} \sum_{c_e \in C_e} \text{eval}(c_e, t)}{\ell_{e,\max}} \tag{4.3}$$

$$Err(\alpha, E) = \sum_{e \in E} \left(\alpha g_e + (1 - \alpha)\ell_e\right)^2 \tag{4.4}$$

The global objective, calculated in Eq. (4.2) as $g_e$, is primarily hierarchical. $g_e$ is the absolute difference between the target $z$-location (1.5 if there is an even number of each cylinder, 0.5 otherwise) and the ending $z$-location of the root at the final time step. A human designer would possibly create a network where information would flow from the leaves to the root where the robot would then aggregate the information to create the correct response.

The local objective, calculated in Eq. (4.3) is primarily modular. $\ell_e$ is found by assessing whether the robot is looking at the white portion of the cylinders in the last half of the evaluation. By expanding out evaluation to the final half of simulation we allow evolution to create a more steady gradient to the optimal solution. At each time step during evaluation, the robot is given a point if it is correctly looking at the white portion of the cylinder and a 0 if it is not. These points are then summed for each cylinder and normalized between $[0, 1]$ and subtracted from 1 to give the error. In correctly assessing a cylinder, each leaf would benefit from having an isolated module which determines whether to move the leaf branch up or down as appropriate.

The overall objective function is a mean squared error consisting of a weighted sum of the error from the two tasks. From these two objectives we explored three tasks determined by Eq. (4.4) using $\alpha = \{0.0, 0.5, 1.0\}$. These values correspond to focusing on only the local objective, both objectives combined, and only the global objective, respectively.

### 4.3.3 Embodied Network Embeddings

Each embedding consisted of four hidden neurons per physical tree branch. The embeddings are embodied because each neuron is placed using information from the

morphology of the robot. The physical embedding uses the actual $(x, y)$ positioning of the robot in its initial state whereas both the binary and hierarchical embeddings use information about the branches location in the abstract tree structure. Each embedding consisted of a number of motor and sensor neurons dependent on the depth of the tree considered. Each leaf branch contained one sensor neuron and one motor neuron. The root contained a single motor neuron. Thus for $d = 1$ there where $3 * 4 = 12$ hidden neurons 2 sensor neurons and 3 motor neurons giving 17 total neurons. For $d = 2$ there where $7 * 4 = 28$ hidden neurons 4 sensor neurons and 5 motor neurons giving 37 total neurons.

The path of a branch is determined by whether the branch is a left or right child or an ancestor. Thus, the path of each branch is a list with length $d$, the depth of the tree. The elements of this list are chosen from $\{-1, 0, 1\}$. A $-1$ indicates the branch is a left child, $+1$ indicates a right child, and 0 indicates the branch is an ancestor to branches in that depth. For example, given a $d = 2$ tree, the leftmost leaf is $[-1, -1]$ because it is left child of the left child of the root. In contrast root's path is $[0, 0]$ because it is the ancestor of both depth one and two branches.

In the physical embedding, neurons were placed along the branches of the physical robot. Thus each neuron had a physical location which corresponded to the robots initial starting position in physical $(x, y)$ space. The physical embedding is in two-dimensional space, regardless of the the depth of the robot.

In the binary embedding, the neurons were placed according to the location of its corresponding branch in the overall tree structure. The $x$ position of the neurons are placed using information about the path and depth of the current branch. Neurons $x = \sum_{i=1^d} p_i * \frac{1}{i}$ where $p_i$ is the $i^{\text{th}}$ index of the path $p$. This results in neurons in

branches of left children being placed to the left of neurons in the parent branches and, conversely, neurons in right child branches are to the right of their parents. The neurons in the root branch are located at $x = 0$. The $y$ coordinate of the binary embedding was chosen to be linear spacing between $[-1, +1]$. Motors were placed at $y = -1$, sensors at $y = +1$ and hidden neurons were linearly spaced in between. In branches without sensors or motors, hidden neurons were still placed as if they existed, meaning hidden neurons from each branch shared $y$ coordinates. This embedding was chosen because it encompasses information about the morphology of the tree, specifically left-right symmetry, while also being extensible to different depths of tree morphologies and remaining in a 2-d embedding. The embedding is shown in more detail in Figure 4.2b.

The hierarchical embedding (Fig. 4.2c) is similar to the binary embedding in that it uses information about the branches location to determine the position of the embedded neurons. However, instead of placing neurons in a 2-d embedding, the dimension hierarchical embedding grows with the depth of the tree, specifically dim $= d + 1$. Each new depth of the tree is a new dimension for the embedding with variations in the positioning at that depth, according to the path of the branch, corresponding to location in that dimension in the embedding. For example, if a branches path in a depth 2 tree is [-1, 1], the corresponding $(x, y)$ embedding for neurons in that branch are $(-1, 1)$ with the $z$ coordinate being determined by the same linear interpolation between $[-1, +1]$ as seen previously. In general, the first $d$ coordinates of the hierarchical embedding are determined by the path with the final coordinate being determined by the linear interpolation. This means at $d = 1$ the binary and hierarchical embeddings are exactly the same. Further, the distance

| | bin/hier | physical |
|---|---|---|
| $\alpha = 0.0$ | **0.0152****** | 0.0185 |
| $\alpha = 0.5$ | **0.0208** | 0.0245 |
| $\alpha = 1.0$ | **0.125** | 0.140 |

*Table 4.1: Depth 1 average minimum fitness at generation 1000. Bolded values indicates minimum across row.* ******** *indicates* $p < 0.0001$ *according to Mann-Whitney U test.*

between in neurons in the hierarchical embedding is reflective of the path distance of the branches within the tree. This means a child is closer to its parent than its sibling and branches with the same parent are closer than branches with different parents.

### 4.3.4 Experimental Parameters

Robots were simulated using Pyrosim, a python interface for Open Dynamics Engine [1]. We used the same parameters for HyperNEAT as in [106] with the exception of changing the population size to 100 due to the computational cost of simulation. Robots were evaluated for 100 time steps in each environment.

The initial population of CPPNs were seeded as in [106]. Because we considered two dimensional substrates, two Gaussian nodes were used and the bias was connected to the LEO output of the CPPN with a -2. This seed means that two neurons which are close together in the embedded $xy$-plane are much more likely to be connected by HyperNEAT.

[1] https://ccappelle.github.io/pyrosim/

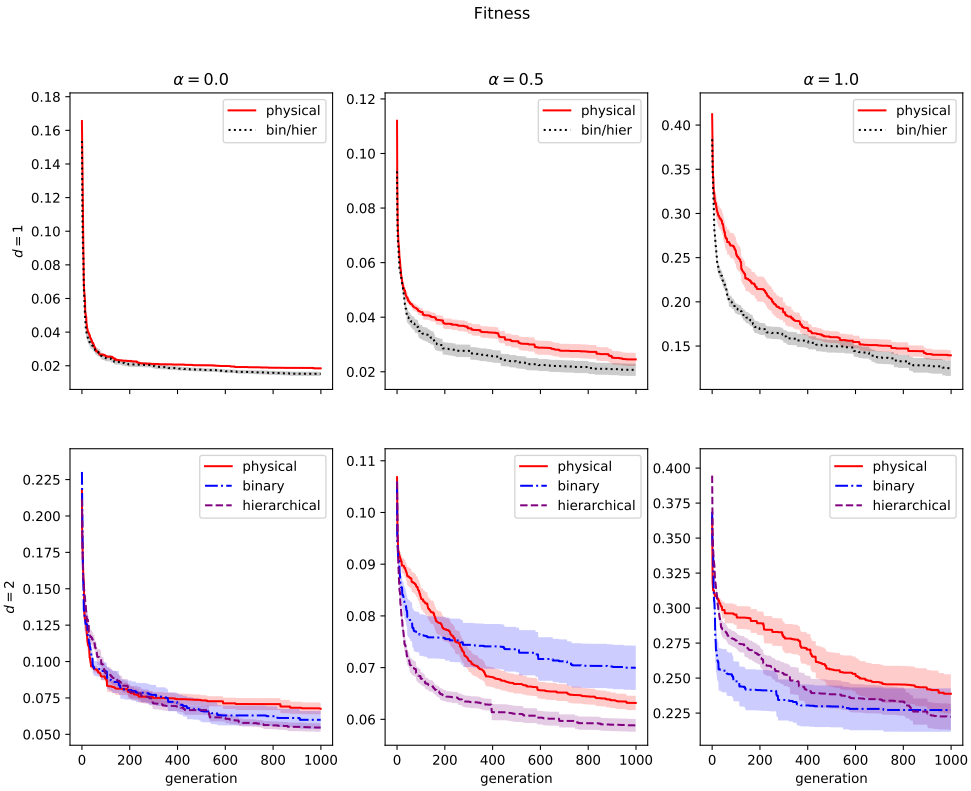



*Figure 4.3: The average minimum error for the three embeddings in the six experiments performed over generational time. The top row contains the average minimum error for depth 1 trees with four total test environments. The bottom row contains the average minimum error for depth 2 trees with 16 total test environments. Each column indicates a different $\alpha$ parameter. $\alpha$ is used to tune the error from the local objective, $\alpha = 0.0$, to the global objective ,$\alpha = 1.0$, as dictated by Equation 4.4. Shaded regions indicate $\pm$ SEM. Only the $d = 1, \alpha = 0.0$ and $d = 2, \alpha = 0.5$ results are significant with $p < 0.05$ according to the Kuskal-Wallis H-test.*

| | binary | physical | hierarchical |
|---|---|---|---|
| $\alpha = 0.0$ | 0.0600 | 0.0675 | **0.0548** |
| $\alpha = 0.5$ | 0.0699 | 0.0632 | **0.0588 *** |
| $\alpha = 1.0$ | 0.227 | 0.239 | **0.222** |

*Table 4.2: Depth 2 average minimum fitness at generation 1000. Bold values indicate minimum across row. $*$ indicates significance at $p < 0.05$ according to Kruskall-Wallis H-Test.*

## 4.4 Results

We ran 30 trials of HyperNEAT-LEO for both the physical and binary embedding for $\alpha \in \{0.0, 0.5, 1.0\}$ with $d \in \{1, 2\}$ for 1000 generations each. The results are presented in Figure 4.3. Amongst the six experiments, two resulted in population in a significant difference in ending average error. In the $d = 2, \alpha = 0.5$ experiment, the physical embedding proved to have significantly better fitness after 1000 generations ($p < 0.05$). Conversely, in the $d = 2, \alpha = 1.0$ experiment, the binary embedding proved to be significantly better error than the physical embedding ($p < 0.05$). All other experiments provided statistically similar ending fitness values for both embeddings.

For every value of $\alpha$ in the Depth 1 experiments, the binary/hierarchical embedding performed better than the physical embedding over 1000 generations however only the $\alpha = 0.0$ results are significant. Complete reporting on ending error of Depth 1 experiments is located in Table 4.1.

For every value of $\alpha$ in the Depth 2 experiments, the hierarchical embedding performed better than both the physical and binary embeddings over 1000 generations

however only the $\alpha = 0.5$ results are significant.

Modularity in the form of network modularity was not present in any of the ending champion networks, every network was completely connected. Regularity was found to be present and varied between the different encodings as seen in Figure 4.4.

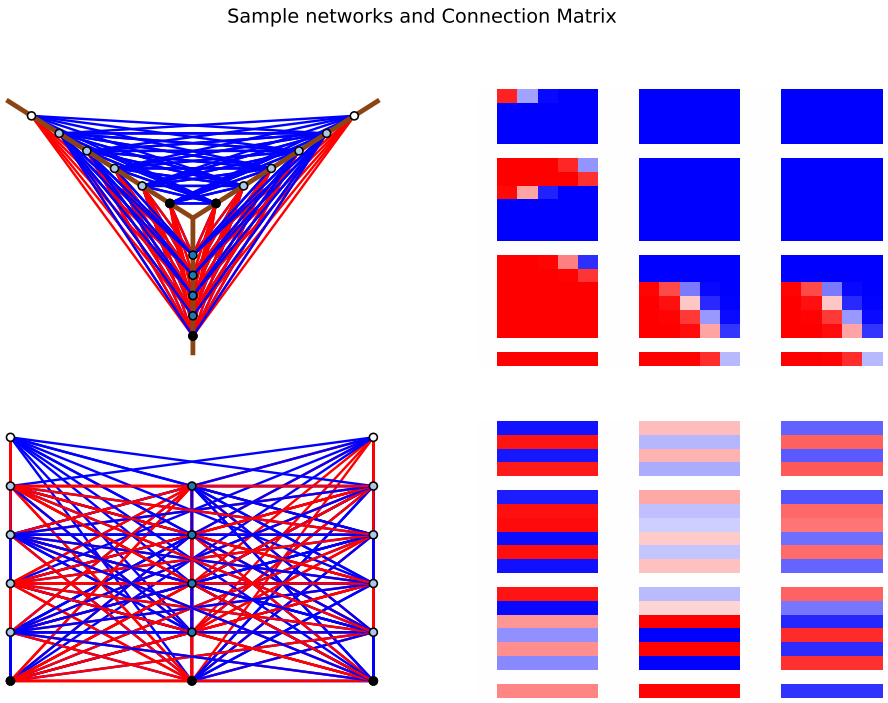



*Figure 4.4: The best run champions from the $\alpha = 0.5, d = 1$ experiment. The physical embedding is shown on the top row and the binary/hierarchical embedding is shown on the bottom row. The left column shows how the network is placed on the embedding and the right column is the same network in adjacency matrix form. Red connections indicate negative synaptic weights while blue indicates positive weights and the alpha of the connection indicates the magnitude of the weight. The adjacency matrix (right) helps show how the positioning of nodes in the embedding impacts the type of connection structure which occurs. The white separations are sensors which cannot be connected to by synapses. The separations help further distinguish the neurons within each branch and how they connect to the neurons in other branches*

## 4.5 Discussion

The location of neurons embedded in the substrate used by HyperNEAT is known to have a difference in the efficiency of evolution to optimize to both embodied and disembodied tasks [25]. By using different *embodied embeddings*, HyperNEAT is able to set connection weights using locality and gradients, present in the morphology, which may stress the importance of certain desirable traits in robot controllers such as hierarchy and modularity. Figure 4.4 shows a clear difference in the types of patterns that are more common given a physical embedding, one where the robots morphology is directly reflected in the location of the neurons, compared to a different type of embodied embedding which uses information about the abstract concept of the structure of the morphology.

Figure 4.3 shows the hierarchical embedding was able to perform better than other embeddingns on a complex task which had both global and local objectives. The increases in dimensionality of the hierarchical embedding helped it compared to the similar binary embedding which uses the same concept of a branches path to determine neuron location but restricts the embedding to two dimensions. These differences are important because it gives an indication as to the nature of the relationship between the morphology of the robot, the structure of the task at hand, and the embedding used.

One important aspect of HyperNEAT is that it is resolution independent meaning it is likely that solutions found are able to scale in a predictable manner [47]. Here we give some insight into how embodied embeddings may be able to be scaled effectively. In either embedding presented in this work, more hidden neurons can easily be placed

by using linear spacing between the end points set by the sensor and motor neurons. In the physical embedding this takes place near the physical $(x, y)$ position of the tips of each branch while in the binary embedding this occurs at $x \in \{-1, +1\}$. Further resolution increases can occur for this robot by increasing the depth of the tree. In other robots this can be thought of as adding different components or sensors and incorporating them into the substrate as prescribed by the emobodied embedding plan. Figure 4.3 shows that while increasing the depth had a impact on the overall error achieved it is important to note that the number of environments between $d = 1$ and $d = 2$ was squared. The increase in depth helped elucidate potential problems with each embedding at these higher depth dimensions.

The disembodied retina task is known to be a benchmark in order to create modularity in evolved neural networks [57, 27]. There are many potential reasons as to why modularity did not form in the experiments performed. One could be that there are plenty of perfectly acceptable non-modular controllers in this task even though, to a human designer, the task seems separable and modular. Another reason could be that there was not enough pressure for modularity to form. [57] only consistently found modularity when the global task the network needed to compute was changed over the course of evolution. This changing every few generations pressured evolution to separate the network into left and right halves. It is possible that for this task, in order for modularity to be present, one would need to change the global objective periodically.

Another way to further constrain connectivity is to explicitly choose for solutions which have less connections through applying a connection cost function. The simplest function to represent cost simply counts the total number of connections present

in the network, however, given the tree structure of the robot, we can direct evolution towards hierarchical solutions by assigning each neuron a physical corresponding branch and computing the path length from the branch of one neuron to the branch of the other. In this manner we could select for controllers which explicitly use a hierarchical structure.

Lastly, it is possible modularity did not occur because this is an embodied task in which speed of movement may play a factor. The more heavily connected a motor neuron is the more likely it is that it will actuate with a higher magnitude velocity. This higher velocity can help the robot achieve its target position more quickly resulting in higher fitness.

## 4.6 Conclusion

In this work we presented the term *embodied embedding* and presented two ways in which it could be performed. One simply took the physical morphology of the robot in space to inform the construction of the substrate. The other used the abstract notion of the robot's structure to create the embedding. Both were able to perform in tasks that, to a human, seem modular and hierarchical. We showed that differences in these embeddings can cause differences in the evolvability of the robots. We further gave insight into the patterns of connections created by certain embeddings and how they can create different networks to complete the same task.

In the future we would also like to investigate if ES-Hyperneat chooses hierarchical embeddings for hierarchical tasks and whether connection cost could help produce modularity in the embodied retina task.

# References for chapter 4


[7] Josh Bongard. "Evolving modular genetic regulatory networks". In: *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*. Vol. 2. IEEE. 2002, pp. 1872–1877.

[12] Josh C Bongard and Rolf Pfeifer. "Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny". In: *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc. 2001, pp. 829–836.

[19] Sean B Carroll. "Chance and necessity: the evolution of morphological complexity and diversity". In: *Nature* 409.6823 (2001), p. 1102.

[25] Jeff Clune, Charles Ofria, and Robert T Pennock. "The sensitivity of HyperNEAT to different geometric representations of a problem". In: *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. ACM. 2009, pp. 675–682.

[27] Jeff Clune et al. "Investigating whether HyperNEAT produces modular neural networks". In: *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM. 2010, pp. 635–642.

[28] Jeff Clune et al. "On the performance of indirect encoding across the continuum of regularity". In: *IEEE Transactions on Evolutionary Computation* 15.3 (2011), pp. 346–367.

[30] David B D'Ambrosio, Jason Gauci, and Kenneth O Stanley. "HyperNEAT: The first five years". In: *Growing adaptive machines*. Springer, 2014, pp. 159–185.

[33] Denis Duboule. "Vertebrate hox gene regulation: clustering and/or colinearity?" In: *Current opinion in genetics & development* 8.5 (1998), pp. 514–518.

[47] Jason Gauci and Kenneth O Stanley. "Indirect encoding of neural networks for scalable go". In: *International Conference on Parallel Problem Solving from Nature*. Springer. 2010, pp. 354–363.

[49] Leland H Hartwell et al. "From molecular to modular cell biology". In: *Nature* 402.6761supp (1999), p. C47.

[53] Joost Huizinga, Jeff Clune, and Jean-Baptiste Mouret. "Evolving neural networks that are both modular and regular: HyperNEAT plus the connection cost technique". In: *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. ACM. 2014, pp. 697–704.

[57] Nadav Kashtan and Uri Alon. "Spontaneous evolution of modularity and network motifs". In: *Proceedings of the National Academy of Sciences* 102.39 (2005), pp. 13773–13778.

[63] Robb Krumlauf. "Hox genes in vertebrate development". In: *Cell* 78.2 (1994), pp. 191–201.

[92] Sebastian Risi, Joel Lehman, and Kenneth O Stanley. "Evolving the placement and density of neurons in the hyperneat substrate". In: *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM. 2010, pp. 563–570.

[93] Sebastian Risi and Kenneth O Stanley. "Enhancing es-hyperneat to evolve more complex regular neural networks". In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM. 2011, pp. 1539–1546.

[94] Gerhard Schlosser and Günter P Wagner. *Modularity in development and evolution*. University of Chicago Press, 2004.

[101] Kenneth O Stanley, David B D'Ambrosio, and Jason Gauci. "A hypercube-based encoding for evolving large-scale neural networks". In: *Artificial life* 15.2 (2009), pp. 185–212.

[106] Phillip Verbancsics and Kenneth O Stanley. "Constraining connectivity to encourage modularity in HyperNEAT". In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM. 2011, pp. 1483–1490.

# Chapter 5

# Morphological Cost Facillitates the Evolution of Modularity

An open area of research in evolutionary biology is the existence of modularity in the structure of organisms. Much of the scientific inquiries into the catalyst for the evolution of modules has ignored the question of morphological modularity–the mechanical independence between portions of the body–and how this modularity relates to the agent's environment. In this study we explore the evolution of populations of embodied agents in a set of environments consisting of variations of modular sub-goals and investigate modularity with respect to the structure present in the environment by introducing a new metric—called sub-goal interference—to measure how environmental variation impacts an agent's response to stimulus. We further compare how implementing a cost associated with an agent's body as well nervous system during evolution impacts the resulting patterns of behavior of fit individuals. We show that our implementation of morphological cost evolves agents with lower sub-goal interference than no cost or neural cost alternatives. With no *a priori* knowledge of the robot

or the environment, we show a simple morphological cost applied during evolution can produce agents which accurately reflect the modularity and structure of their environments. This suggests that for certain tasks and environments, the relationships within an agent's body can be more fundamental towards exhibiting modularity with respect to a set of tasks compared to the structure of its nervous system.

## 5.1 Introduction

Modularity is present at multiple scales and varieties–structural, functional, etc.–within an organism [16, 49, 110, 114]. Consider the breakdown of an organism into its separate organs, specialized tissues within each organ, independent cells within the tissue, and functionally different organelles contained within each cell. While the prevalence of modularity in biological life has made it a popular target of study, quantifying modularity within a system is notoriously difficult due to its inherently subjective description [6, 16]. Despite this difficulty, most definitions of modularity share a reliance on some concept of independent features acting as part of a whole.

Here we explore the modularity of movement and perception with respect to environmental stimulus. Modularity in robotics has been explored through the use of modular robots, however the modules which make up the robot in these studies are predefined and not directly evolved [105, 73]. When considering an example of evolved embodied modularity, it is possibly easiest to imagine this concept as the dexterity present in a hand, or hand equivalent structure. The evolution of hands and limbs has been one of the most studied aspects of evolutionary development research [78, 107, 114].

Consider the movement of one digit of the human hand. For the most part, there is the ability to move one finger without impacting the position and orientation of the remaining fingers in the hand. The movement between left and right hands is even more independent than between fingers within the same hand. This begs for evolutionary explanations for many independent degrees of freedom within the hands of some species, but many coupled degrees freedom in others. Or more generally, what leads to increasing independence of motion within an organism? Given consideration of specific examples we can suggest possible explanations.

The first example to consider is the evolution of whales and other sea mammals which evolved from land mammals. These mammals reverted the previous evolution of mechanically independent fingers and toes into flippers which limit the motion of the still present digits [104]. The evolution of flippers are a product of the animals' aquatic environment where having independent digits without some sort of webbing will naturally be a detriment to swimming. The evolution of whales demonstrates how certain environments impose selective pressure towards more mechanically dependent morphology. However, the environment can also pressure towards a specialization of independent digits. The aye-aye is a lemur with a highly specialized third digit, which aids in foraging for insect larva in the hollows of trees [85]. With this digit, the aye-aye is capable of completing highly dexterous maneuvers in pursuit of food which its other fingers cannot achieve [76]. The specific form of the hands evolved within the whale and aye-aye species provide support to the idea of environment guiding evolution. However there may be other, non-environmental pressures, which can impact the evolution of mechanical independence. We evolve populations of simulated agents in tasks and environments containing modular sub-goals while employing both

morphological and neurological cost to the phenotype in order to explore how the various factors contribute to mechanical modularity.

By simulating the evolutionary process, we can explore evolution at a faster rate while also isolating factors which may cause differences in evolutionary trajectories. Simulation of evolved agents can be separated into two categories: disembodied and embodied. Disembodied approaches (Sec. 5.1.1) treat agents as separate from their environment. That is, while the agents are tasked to receive and react to signals from the environment, there is no action they can take which impacts future perception. Studies of disembodied agents generally implement agents as networks. In contrast, embodied agents have the ability to move through an environment, which in turn affects future behavior (Sec. 5.1.2). While both approaches are valid for exploring the evolutionary process, in this work we implement an embodied agent to study the evolution of mechanical modularity.

### 5.1.1 Disembodied modularity

Computational studies of the evolution of modularity have largely concerned modularity with respect to disembodied networks [57, 58, 23]. There it has been shown that modularity in networks can evolve by exposing an agent to a set of environments which temporally change the global task by varying common sub-goals [57, 38]. This is referred to as modularly varying goals (MVG). Populations of highly modular networks evolve more rapidly to adapt to these changes. While the plausibility for MVG to occur in nature has been shown in certain cases [84], there are most likely many other factors which contribute to the evolution of modularity. Sparsity is an example of another possible factor which has been explored in disembodied networks [23, 45,

4, 37]. Selection for more sparse individuals can be achieved through the implementation of connection cost:a cost levied on the topology of the network, depending on how it is connected. Connection cost can be directly selected for using multi-objective methods [23] or implicitly through the choice of mutation operator [45]. In this work we similarly implement a fitness function which considers both performance and cost while evolving the population of agents in a set of environments consisting of modular sub-goals. However, this work differs from computational studies of disembodied agents because we implement a robot which can move to change future perception of the environment. Further, the use of a body means we cannot use traditional techniques to calculate modularity.

The standard calculation of modularity in a network is through the use of $Q$ [79, 69]. $Q$ is calculated by finding the optimal community structure and then comparing the number of intra-module connections to the number of expected connections of a random graph with the same degree distribution. This means $Q$ only measures structural modularity present in the network; it does not take into account mechanical aspects of the modules nor their physical interactions with the environment [70]. Further, $Q$ has been shown to produce results which are unintuitive [2] and fails to resolve smaller modules as the size of the network increases [42]. The resolution issue of $Q$ has been addressed in more recent work, however it requires the calculation of $Q$ in many random networks [37]. Lastly, $Q$ may only be applied to structures which can be represented as networks. This poses a problem for investigating the modularity present within both an agents' body plan and nervous system. We instead use an embodied measure of modularity called sub-goal interference, $I$, to explore what we believe is a more descriptive metric of modularity for embodied agents. $I$ is fully

defined in Section 5.3.2.

While the investigation of disembodied networks is useful for a more complete understanding of neural pathways, gene regulatory networks, and complex interactions within organisms, actually describing the functional behavior of an agent and how modularity in the body plan impacts evolution requires that the agent be able to affect its perception of its environment.

### 5.1.2 Embodied modularity

Although embodiment is often mentioned in Evolutionary Robotics (ER), many experiments only evolve the synaptic weights in a fixed-topology neural controller, nor is the morphology of the robot evolved [32, 55]. To accurately investigate the evolution of biological organisms we must allow evolution to dictate both neural topology and morphology. Studies which do consider morphological change generally restrict agents to be evolved in a singular environment [98, 9]. Further work into the impact of environment on morphology has been limited to exploring the relationship between environmental complexity and morphological complexity [1], not on how the underlying structure of the environment can be reflected in the body of the evolved agent. These studies do not consider the concept of morphological modularity.

We have previously defined morphological modularity as the amount a subset of motors impacts future values of a subset of sensors [17]. However, modularity within the body plan alone does not imply the robot correctly reflects the structure of the environment [18]. When the structure of a robot accurately reflects the structure of the environment, it is know as *ecologically modular*. When *a priori* knowledge of desired modular behavior within an environment is known, it can be used to increase

evolvability [13]. In this work we show modularity with respect to the environment can be evolved by applying a simple morphological cost which requires no *a priori* knowledge of the environment or robot.

## 5.2 Robot Representation

In this section we detail the structure of the robot's morphology (§5.2.1), controller (§5.2.2), and genome (§5.2.3). The complete design of the robot can also be seen in Figure 5.1. The robot was simulated using *pyrosim*[1], a python interface for Open Dynamics Engine (ODE) [62, 99].

### 5.2.1 Morphology

The robot was expressed as a tree morphology. A tree was chosen because it can exhibit intuitive forms of modularity and hierarchy–movement of leaves of the tree are independent while movement of the root impacts the global state of the tree–and because most relevant robot morphologies are able to be described as trees [65]. The robot consisted of a depth three binary tree consisting of seven total limbs: one root limb, two middle limbs, and four leaf limbs (Fig. 5.1). Each limb contains a one degree-of-freedom rotational motor in the joint connecting it to the parent limb. The root base was connected to an arbitrary point in space allowing for rotation relative to the world. Motors allowed lateral rotation between connected limbs in the horizontal plane. The joints were restricted to a maximum range of motion depending on the depth of the limb containing the joint. These restrictions prevent sibling limbs from

[1] https://ccappelle.github.io/pyrosim/

colliding with one-another. The root joint was limited to rotations in $[-120°, +120°]$, middle limb joints were limited to $[-45°, +45°]$, and leaf limb joints were restricted to $[-20°, +20°]$. Each leaf contained a distance sensor in its tip pointing out into the environment along the long axis of the limb, thus allowing the robot to detect objects in the environment.

### 5.2.2 Neural Controller

Controllers were encoded as Continuous Time Recurrent Neural Networks (CTRNNs), a common formalism when controlling robots in tasks which require memory [5]. Each limb consisted of three hidden neurons and one motor neuron. Each leaf limb was also equipped a sensor neuron. More neurons theoretically allow the robot to complete more complicated tasks however, each additional neuron also increases the search space as parameters describing those neurons and the synapses associated with them, must be optimized as well. The number of neurons used was found to provide sufficient performance in an acceptable given the available computational resources. Four sensor neurons, $3 \times 7 = 21$ hidden neurons, and seven motor neurons were employed, yielding $4 + 21 + 7 = 32$ total neurons. Because recurrent connections were allowed and sensor neurons only obtained input from the distance sensors, there were $32 \times (32 - 4) = 896$ total possible synapses the robot's controller could express.

### 5.2.3 Genome

Each genome consisted of three components: the weight matrix $\mathbf{W} \in \mathbb{R}_{32\times28}$, the network topology matrix $\mathbf{N} \in \{0, 1\}_{32\times28}$, and the joint range vector $\mathbf{r} \in [0, 1]_{7\times1}$.

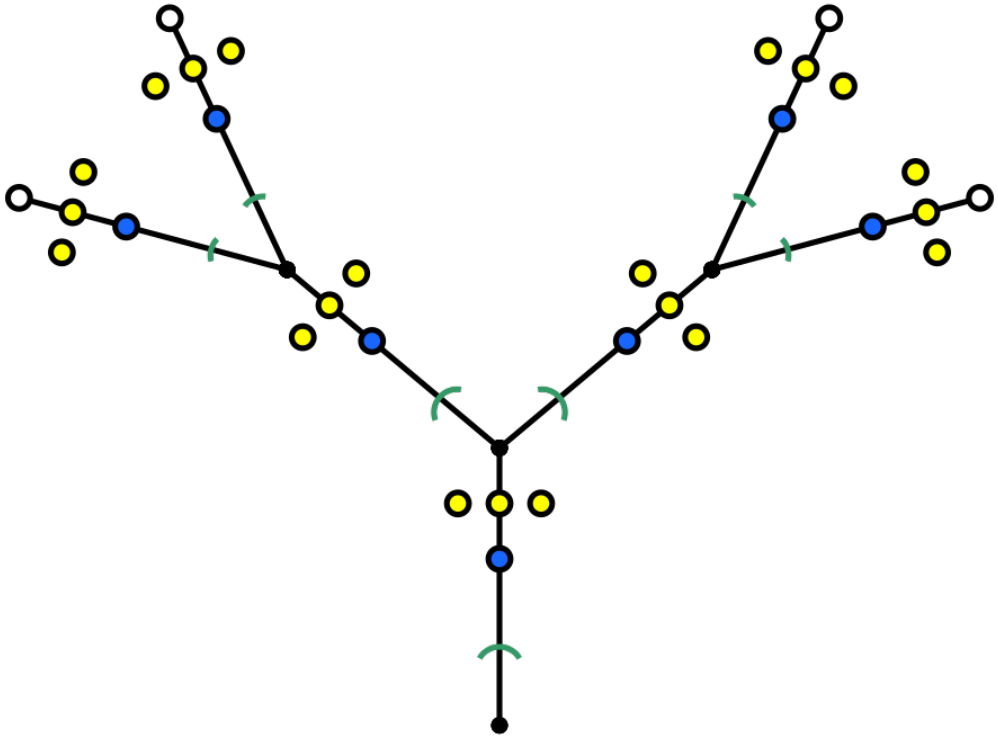


*Figure 5.1: A schematic drawing of the robots morphology and controller. Each robot had seven limbs (black lines). Each limb had three hidden neurons (yellow circles) and one motor neuron (blue circles). Leaf limb also had one sensor neuron (white circle) for a total of 32 neurons. Each sensor neuron had the potential to be connected to every other non-sensor neuron through a synapse. The value of the motor neurons controlled the position of the joints located in the base of each limb (black circles). The green arcs specify the possible movement range of each limb.*

The weight matrix $\mathbf{W}$ consisted of float values which determined the weight of each synapse with entry $\mathbf{W}_{ij}$ corresponding to the synaptic weight from neuron $i$ to neuron $j$. The network topology matrix $\mathbf{N}$ was a matrix of Boolean values and determined whether the synapse from neuron $i$ to neuron $j$ was expressed in the phenotype. Zero dictates that the synapse between neuron $i$ and neuron $j$ is not expressed in the phenotype, regardless of the value of $\mathbf{W}_{ij}$; one indicates the synapse is expressed with weight $\mathbf{W}_{ij}$. Lastly, the joint range vector $\mathbf{r}$ was a vector of float values between 0 and 1. $\mathbf{r}$ determined the proportion of the maximum joint range each joint could rotate through in the phenotype. If we let $\text{range}_{i,\max}$ be the maximum range joint $i$ can achieve, then the actual range of joint $i$ in the phenotype would be $[-\mathbf{r}_i \times \text{range}^{\circ}_{i,\max}, +\mathbf{r}_i \times \text{range}^{\circ}_{i,\max}]$. Thus if $\mathbf{r}$ is a vector of zeros, the robot cannot move at all regardless of $\mathbf{W}$ and $\mathbf{N}$.

## 5.3 Methods

This section provides the methods used in this paper. (§5.3.1) details the environments and task, (§5.3.2) provides the formal definition of sub-goal interference. (§5.3.3) outlines the evolutionary algorithm. (§5.3.4, §5.3.5, §5.3.6) overview the objective, cost, and fitness functions respectively. Lastly, (§5.3.7) presents the experimental configurations.

### 5.3.1 Task Environments

The robot was tasked with classifying cylinders it perceived in its environment,an embodied analog of the retina task used in previous research to investigate disembod-

ied modularity [57, 23, 27, 106].

It is useful to categorize the number of environments in terms of the free parameters in the system and variations on those free parameters [74]. In this study we explore an environment with four free parameters: the locations of four cylinders (Fig. 5.2). Each cylinder had two variations: near or far. Near cylinders were placed three units away from the tip of the closest leaf and far cylinders were placed six units away. Let $f = 4$ be the number of free parameters (one for each cylinder) and $n = 2$ (near, far) be the number of variations. There are $n^f = 2^4 = 16$ total environments.

Let any environment be represented by a $f$-tuple of its $n$ variations where each entry refers to the specific variation of that free parameter. In this case we can express a single environment as a 4-tuple. Thus we write $e = (c_0, c_1, c_2, c_3)$ where $c_0, c_1 \in \{a, b\}$ and $c_2, c_3 \in \{0, 1\}$. For the two cylinders to the robot's left we let $a$ denote 'near' and $b$ denote 'far'. For the right two cylinders we let 0 denote 'near' and 1 denote 'far'. Thus the environment $(a, b, 1, 0)$ corresponds to the environment in which the cylinders are {near,far,far,near} if we start from the left-most cylinder and end with the right-most cylinder.

The robot was tasked with three independent sub-goals. The first two sub-goals relate to the robot's response to $c_0$ and $c_1$ to showcase an example of a sub-goal which is made up of a singular free parameter. The robot should point towards $c_0$ if $c_0 = b$ and point away if $c_0 = a$. Behavior with regards to $c_1$ was the same as $c_0$. The third sub-goal was chosen to be an example of a sub-goal which is a combination of two free parameters. This follows the Retina task approach in which a logical function was computed from the classification of left and right sub-problems. The robot was tasked with pointing towards cylinders $c_2$ and $c_3$ if and only if $c_2 \oplus c_3$ is false where $\oplus$

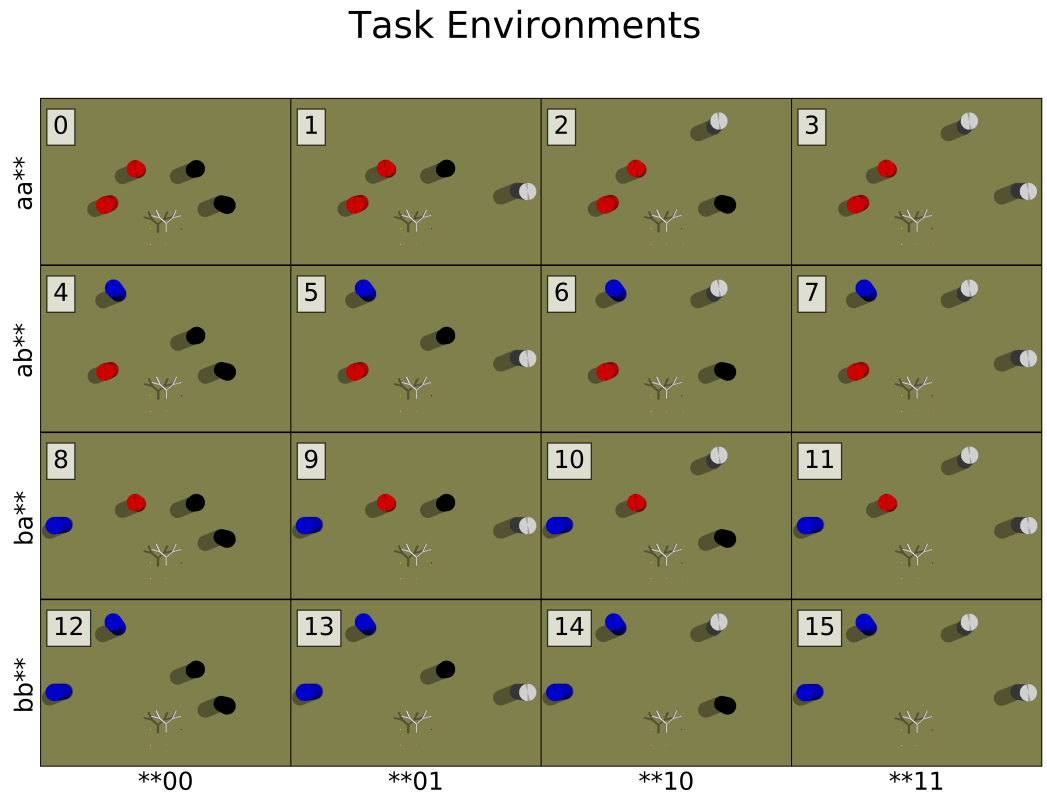



*Figure 5.2: Snapshots of the starting point of the robot in all sixteen possible task environments. Cylinders varied between near (designated a or 0) and far ( designated b or 1). The cylinders in the left half vary along the rows and cylinders on the right half vary along columns. The cylinders are colored as follows to allow for easier distinction: a is red, b is blue, 0 is black, and 1 is white. The environments are further numbered 0-15 for convenience.*

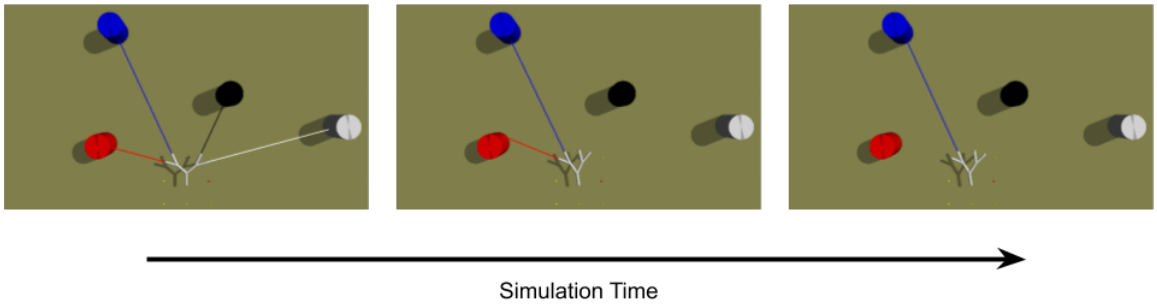



*Figure 5.3: A robot actuating over its lifetime in the $(a, b, 0, 1)$ environment (number 5 in Fig. 5.2). The colored rays emitting from the limbs indicate the robot is looking at that cylinder. One sub-goal is to look away from cylinders labeled a (red), at cylinders labeled b (blue). Another sub-goal is to compute the XOR function on cylinders labeled* 0 *(black) and* 1 *(white). In this case XOR evaluates to true meaning the robot should point away from both cylinders on the right side. This robot exhibits the correct behavior for this environment because it points towards the blue cylinder and away from all others.*

is the XOR function. This setup was chosen because it exhibits environments which contain both modular and non-modular components. Figure 5.3 shows an example robot acting in environment $(a, b, 0, 1)$ over simulation time.

### 5.3.2 Sub-goal interference

In this section we introduce the concept of sub-goal interference, denoted $I$. $I$ is a metric for evaluating to what extent variation in an independent sub-goal impacts movement of the robot related to a different sub-goal over the course of simulation.

To specify $I$ we define the following:

$\mathbf{G} =$ the set of sub-goals

$\mathbf{V_g} =$ the set of variants of specific sub-goal $g \in \mathbf{G}$

$\mathbf{L_g} =$ the set of limbs in the tree associated with sub-goal $g \in \mathbf{G}$

$\mathbf{E_{v,g}} =$ the set of environments associated where variant $v \in \mathbf{V_g}$ and $g \in \mathbf{G}$

$T =$ the total number of simulated time steps

$\alpha_{\ell,t,e} = the$ angle of limb $\ell$ at time $t$ in environment $e \in \mathbf{E_{v,g}}$

$\overline{\alpha_{\ell,e}} =$ the average angle over time of limb $\ell$ in environment $e \in \mathbf{E_{v,g}}$

We then calculate sub-goal interference as

$$I = \frac{1}{|\mathbf{G}|} \sum_{g \in \mathbf{G}} \frac{1}{|\mathbf{V_g}|} \sum_{v \in \mathbf{V_g}} \frac{1}{|\mathbf{L_g}|} \sum_{\ell \in \mathbf{L_g}} \frac{1}{|\mathbf{E_{v,g}}|} \sum_{e \in \mathbf{E_{v,g}}} \frac{1}{T} \sum_{t=0}^{T} |\alpha_{\ell,t,e} - \overline{\alpha_{\ell,e}}|$$

Thus the lower $I$ is, the more invariant a robot's behavior is towards achieving a specific sub-goal to variations in other sub-goals.

To further clarify, in the following example we explore one specific variation of one sub-goal out of our $|G| = 3$ total possible sub-goals. Consider the sub-goal of classifying the left-most cylinder, $c_0$. Let $g$ denote this sub-goal and let $V_g$ denote the set of variations contained in this sub-goal. Then $|V_g| = 2$ because $g$ has two variations: the robot is either presented with $c_0 = 0$ or $c_0 = 1$. Let us choose the case when $c_0 = 0$. If we consider the limbs which contribute to sub-goal $g$, we see that they are the limbs which, when moved, have an impact on limb $\ell_2$, the limb which

starts out the simulation pointing towards $c_0$. These limbs are the root ($\ell_0$) the left limb at depth one ( $\ell_1$) and $\ell_2$ itself. Thus $L_g = \{\ell_0, \ell_1, \ell_2\}$. Next we can specify $E_{v,g}$ by enumerating all of the environments in which $c_0 = 0$. This can be denoted by $(0, *, *, *)$ where $*$ is the wild card operator with two choices. Thus there are $2^3 = 8$ environments where $c_0 = 0$, or exactly half of the total environments the robot is tasked with. Then, at each time step, for each environment, for each limb, we take the absolute difference between the angle of the limb and the average angle of that limb over each of the environments. This measures how similarly the limbs which are relevant to the sub-goal move when cylinders which do not impact the sub-goal are changed.

Therefore, $I = 0$ indicates that the robot displays infinitely conservative behavior: for each sub-goal, the relevant limbs move exactly the same for each variation of that sub-goal.

### 5.3.3 Evolutionary Algorithm

In order to optimize the robot we employed Age-Fitness Pareto Optimization (AFPO) [95] a multi-objective evolutionary algorithm that maintains solution diversity by evolving genetically independent lineages in the same population.

We employed different mutation operators for each part of the genome: synaptic weights, neural topology, and morphology.

For the synaptic weight matrix $\mathbf{W}$ each entry could mutate or not according to some probability, $p_W$. If a location $\mathbf{W}_{ij}$ was chosen for mutation, then the mutation $\widehat{\mathbf{W}_{ij}}$ was calculated by adding a random number drawn from a Gaussian distribution with mean zero and standard deviation proportional to the magnitude of the weight

of the synapse

$$\widehat{\mathbf{W}_{ij}} = \mathbf{W}_{ij} + \mathcal{N}(0, \max(|\mathbf{W}_{ij}|, \varepsilon))$$

where $\varepsilon = 0.01$. This mutational setup allows a mutation in larger synaptic weights to make more drastic changes and smaller weights to undergo smaller mutations. The $\varepsilon$ is used to ensure non-zero standard deviations.

As for the neural topology matrix $\mathbf{N}$, each location undergoes mutation according to probability $p_N$. If a location is selected for mutation, the bit in that location is flipped. So, after mutation we have

$$\widehat{\mathbf{N}_{ij}} = \begin{cases} 0 & \text{if } \mathbf{N}_{ij} = 1 \\ 1 & \text{if } \mathbf{N}_{ij} = 0 \end{cases}$$

Finally, to mutate the joint range vector $\mathbf{r}$, we again decide whether each location in the vector undergoes mutation using some probability $p_r$. If a joint range is selected for mutation, the new joint range value is found by adding a random number from a Gaussian distribution with mean zero and standard deviation of 1/10. A hyper-parameter which was selected after exploratory results that indicated the mutations altered behavior without being too disruptive to evolvability.

$$\widehat{\mathbf{r}_i} = \mathbf{r}_i + \mathcal{N}(0, 0.1)$$

For each portion of the genome, the probability of mutation was such that the expected number of mutations was one. Thus $p_W = 1/(32 \times 28)$, $p_N = 1/(32 \times 28)$ and $p_r = 1/7$.

### 5.3.4 Objective Function

In order to evolve the robots to accomplish their task a fitness function (Sec. 5.3.6) was implemented which incorporated objective performance and cost. The behavior was measured based on how well the robot performed the task of 'looking' towards or away from the appropriate cylinders in each of the test environments denoted $E_{\text{test}}$.

The overall objective performance for a robot (Eq. 5.1) was its average performance in each environment in the test set (Eq. 5.2). The individual environmental objective value of a robot was calculated during the last half of simulation time $t \in [T/2, T]$. By deferring evaluation to the later half of simulation, the robot is allowed to move in the beginning without adversely impacting its performance. During each evaluation time step $t$, each cylinder $c$ was considered. Cylinders could be correctly pointed at, incorrectly pointed at, or not pointed at. If the task specification required cylinder $c_0$ to be pointed at and the robot was pointing at $c_0$, this was denoted as correct pointing and the robot was rewarded. If the task specification required cylinder $c_0$ to not be pointed towards and the robot was pointing at $c_0$, this was denoted as incorrect pointing and the robot received a penalty. When the robot was not pointing at a cylinder, no reward or penalty was implemented (Eq. 5.3). The reward/penalty over the evaluation time was then normalized based on the environment to lie in the range $[0, 1]$ (Eq. 5.2). Normalization was performed based on the minimum and maximum reward the robot could achieve in the environment. For example, if the robot was supposed to look away from all cylinders in the environment, the reward range for that environment would be $[-4T/2, 0]$ because the worst performer would receive a penalty at every time step and the best performer would

receive no reward. Normalization mapped the robot's score to the range $[0, 1]$ and is implement in Equation 5.2.

$$o_{\text{overall}} = \frac{1}{|E_{\text{test}}|} \sum_{e \in E_{\text{test}}} o_{\text{environment}}(e) \tag{5.1}$$

$$o_{\text{environment}}(e) = \frac{1}{T/2} \sum_{t=T/2}^{T} \frac{(\sum_{c \in C_e} o_{\text{cylinder}}(c,t)) - \min(C_e)}{\max(C_e) - \min(C_e)} \tag{5.2}$$

$$o_{\text{cylinder}}(c,t) = \begin{cases} -1 & \text{if cylinder } c \text{ is incorrectly being} \\ & \quad \text{pointed towards at time } t \\ 0 & \text{if cylinder } c \text{ is not being pointed} \\ & \quad \text{towards at time } t \\ 1 & \text{if cylinder } c \text{ is correctly being} \\ & \quad \text{pointed towards at time } t \end{cases} \tag{5.3}$$

### 5.3.5 The cost functions

Two cost functions were considered: one for the neural controller and one for the morphology. Both cost functions were normalized $[0, 1]$.

The cost of a robot's controller, or connection cost (CC) [23], was computed to be the number of total expressed connections present in the network. Specifically

$$\text{CC} = \frac{1}{32 \times 28} \sum_{i=1}^{32} \sum_{j=1}^{28} n_{ij}$$

for $n_i j \in \mathbf{N}$. Therefore, robots with more connections were considered more costly than robots with fewer connections.

The cost of a robot's morphology, or joint cost (JC), was computed as the average of the elements of $\mathbf{r}$, the joint range vector of the robot. Thus robots which had the potential to move more were considered more costly. A robot which could not move therefore had zero cost.

### 5.3.6 The fitness function

The fitness function was a combination of the objective and two cost functions: maximize performance while minimizing cost. This was accomplished by simple multiplication of objective and cost terms (Eq. 5.4). Because each term ranges between zero and one, the overall fitness was also between zero and one.

$$f = o_{\text{overall}} \times (1 - x_{\text{CC}}\text{CC}) \times (1 - x_{\text{JC}}\text{JC}) \tag{5.4}$$

$x_{\text{CC}}$ and $x_{\text{JC}}$ determined the influence of connection and joint cost respectively towards the fitness score. These influence factors were implemented to allow variation in the importance of each cost function.

### 5.3.7 Experimental configurations of mutation and cost

This section details the different evolutionary trials performed. The trials differed in what evolution could alter and which fitness function was used to guide evolution.

The first difference between trials was the mutation function used to alter the population across generations. The specific mutation function used in an evolutionary al-

gorithm is important to the solution forms which result [32]. Many mutation functions in ER are limited to just synaptic weights or neural architecture. Here we test various mutation functions to determine which proves most efficacious for evolving low interference robots. We considered four choices for mutation: weights, topology, joints and all. Each choice differed in which portion of the genome evolution could mutate.

- *weights* could only mutate in **W**.
- *topology* could mutate **W** and **N**.
- *joints* could mutate **W** and **r**.
- *all* could mutate **W**, **N**, and **r**.

**W** is always considered subject to mutation because if **W** is constant, evolution cannot complete the categorization task.

In the cases where **N** and **r** were not under evolutionary control, they were fixed such that all values were equal to one. This means, for example, robots using the *weights* configuration used all synapses ($\mathbf{N} == 1$) and were maximally free to rotate their limbs ($\mathbf{r} == 1$).

The second difference between the configurations was the value of $x_{\mathrm{CC}}$ and $x_{\mathrm{JC}}$ which dictated how the cost function was implemented, and thus what fitness function was used to evaluate robots. These tests explore which cost function aids in the evolution of modularity. We considered four cost options: none, CC, JC, CC+JC.

- none had $x_{\mathrm{CC}} = x_{\mathrm{JC}} = 0$
- JC had $x_{\mathrm{CC}} = 1$ and $x_{\mathrm{JC}} = 0$
- JC had $x_{\mathrm{CC}} = 0$ and $x_{jc} = 1$

| | | Mutation | | | |
|---|---|---|---|---|---|
| | | weights | topology | joints | all |
| Cost | none | (weights,none) | (topology,none) | (joints,none) | (all,none) |
| | CC | | (topology, CC) | | (all, CC) |
| | JC | | | (joints, JC) | (all, JC) |
| | CC+JC | | | | (all, CC+JC) |

*Table 5.1: The nine configurations possible for evolutionary trials. Performed experiments are highlighted in grey and named. White cells of the table represent infeasible configurations. For example, (joints, CC) does not make sense to test because the connection cost would be constant regardless of the genome.*

- CC+JC had $x_{\text{CC}} = x_{\text{JC}} = 1$

Thus each configuration could be written as a pair of what was mutated and what cost function was used. Combinations in which cost remains constant across all genomes in the population were not considered as cost would impart no effect. For example, the combination (weights, CC) makes no sense to explore because connection cost is constant when $\mathbf{N}$ is fixed. This yields nine possible configurations concerning evolution and cost. They are summarized in Table 5.1.

## 5.4 Results

This section details the two main experiments performed and the subsequent results.

In the first experiment, we test the performance and sub-goal interference of each configuration when evolved in all 16 environments. 30 independent populations of each configuration were evolved over 4000 generations. Figure 5.4 compares the joint

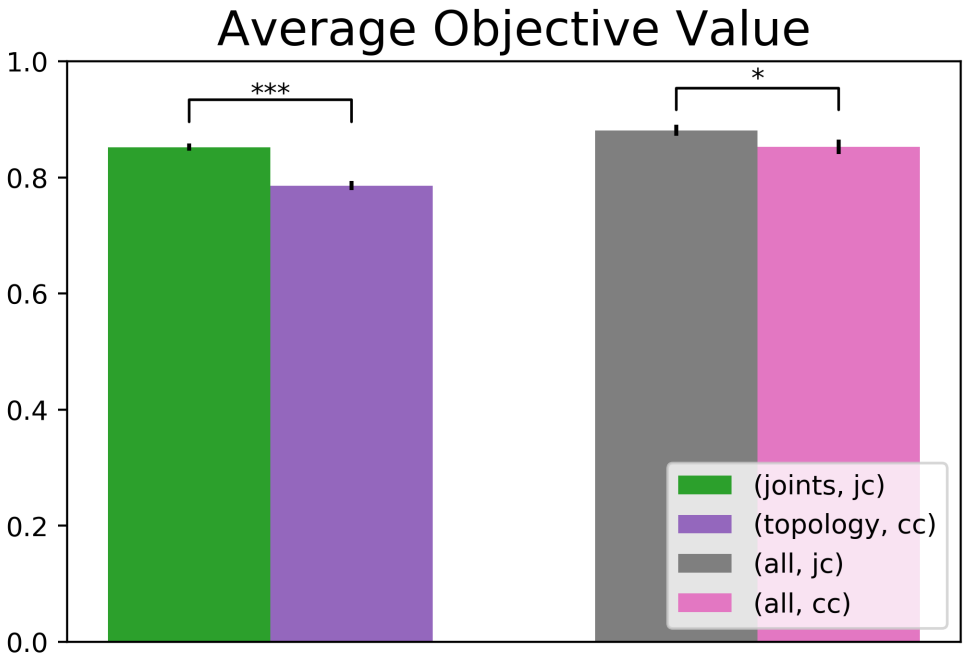



*Figure 5.4: The average ending maximum objective value for the selected configurations comparing joint cost to connection cost. (joints, JC) was compared to (topology, CC) and (all, JC) was compared to (all, CC). Comparisons were performed using the Mann-Whitney U Test. *** indicates a p-value* $< 0.0005$ *and* $*$ *indicates a p-value* $< 0.05$.

cost and connection cost configurations average performance of the highest performing individual after evolution. Both joint cost configurations yield significantly higher performance than the connection cost alternatives. Average performance over generational time is shown in Figure 5.5.

When evaluating a robot in multiple environments, there are many ways to aggregate performance across each environment. Figure 5.6 shows the trial average environmental performance $o_{\text{environment}}$ of the most fit robot in its worst environment. This helps to indicate if the evolved robots exhibit generalism across all environments or if they sacrifice performance in one environment to specialize on other environments.

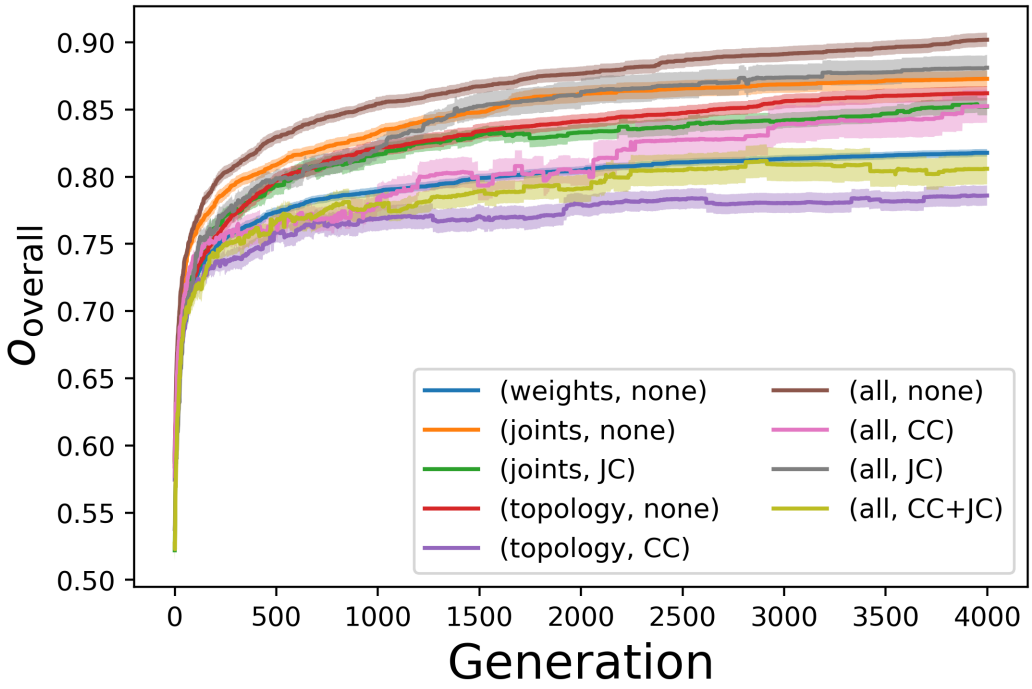



*Figure 5.5: Performance over generational time for each configuration.*

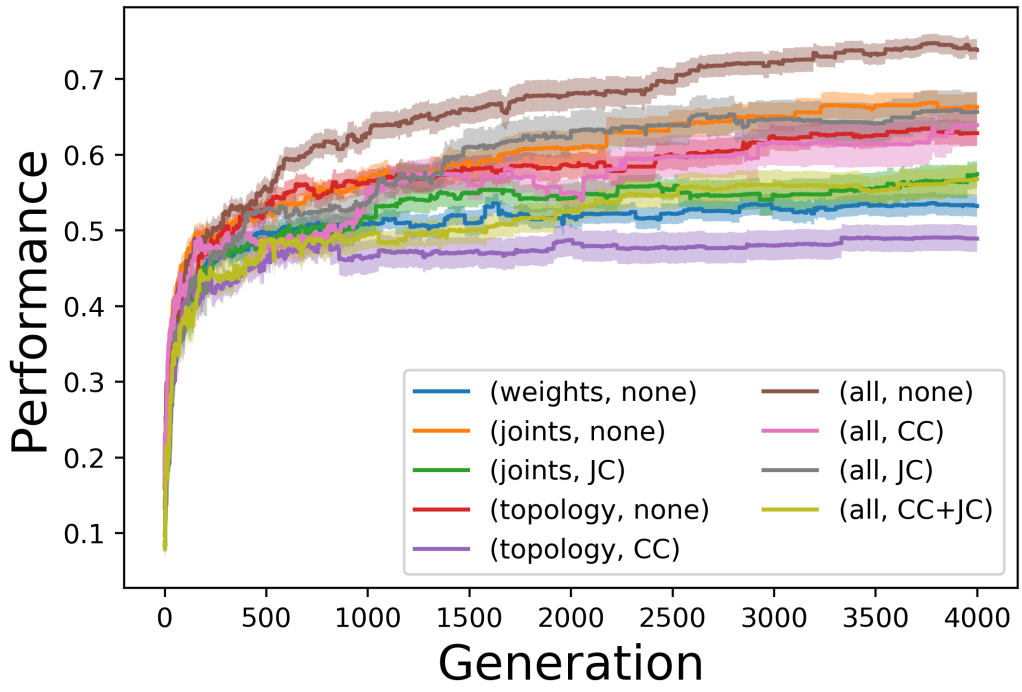



*Figure 5.6: The average value of the best robot's worst environment over generational time.*

Figure 5.7 reports the percentage of trials in which the highest performing robot in the population's lowest performing environment is above the specified threshold value. In ten percent of evolutionary runs, the (all,JC) configuration created a robot with above 0.9 performance in every environment. 6.6% of (all,none) and (all,CC) configurations were able to create a robot with above 0.9 performance in every environment. All other trials of every other configuration failed to create a robot with this high performance in every environment.

The average sub-goal interference $I$ of the highest performing robots was calculated and the results for various comparisons between configurations are presented in Figures 5.8, 5.9, and 5.10. Figure 5.8 reports the $I$ value of (joints, JC) to be

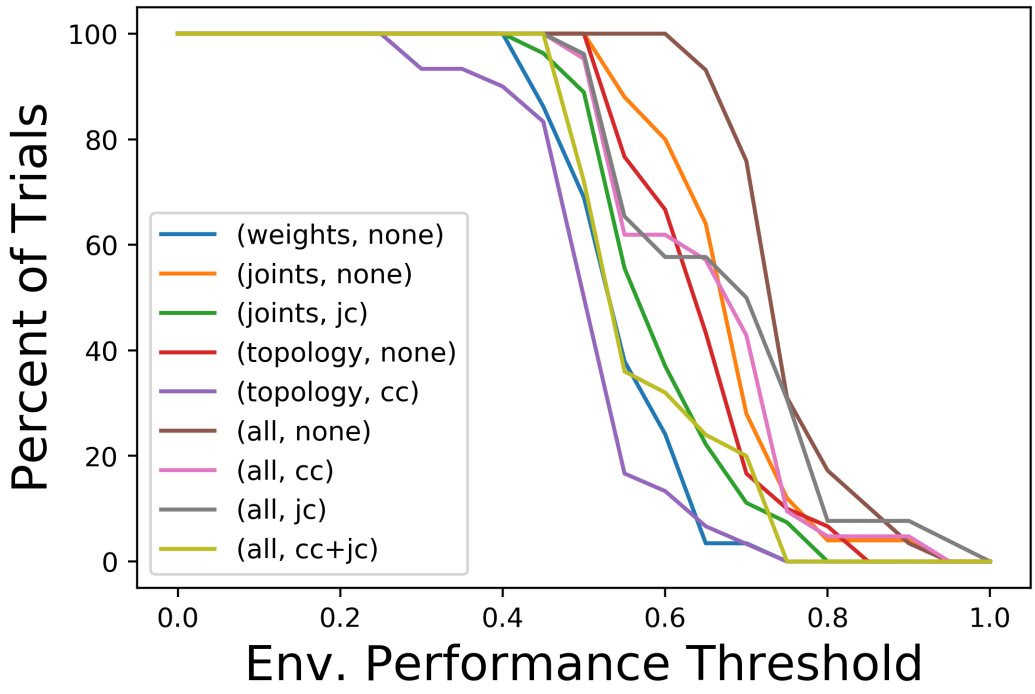



*Figure 5.7: Percentage of trials in which the best robot after 4000 generations achieved a performance greater than or equal to the threshold value in every environment.*

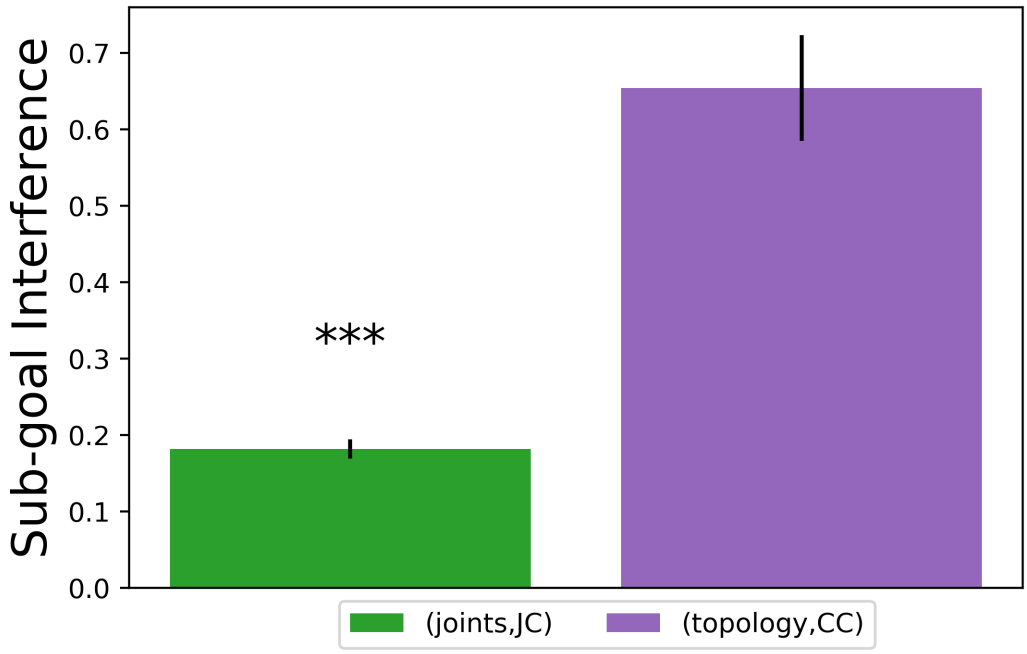



*Figure 5.8: The calculated interference values of the (joint,JC) and (topology,CC) configurations. The asterisks indicates a p-value of* $< 0.0001$ *according to the Mann-Whitney U Test.*

significantly lower than (topology,CC). Figure 5.9 compares the (all,none), (all, CC), and (all,JC) configurations. The $I$ value of the (all,JC) configuration is significantly less than that of the (all,none) configuration, however the other pairwise comparisons are not significant (Fig. 5.10).

Figures 5.11, 5.12, and 5.13 report the movement patterns of the highest performing individual of chosen configurations in each of the sixteen environments. The movement pattern was the position of each limb through simulation time.

In the second experiment, we test how well each configuration performs over the whole environment space when evolved only in a subset of environments. Based

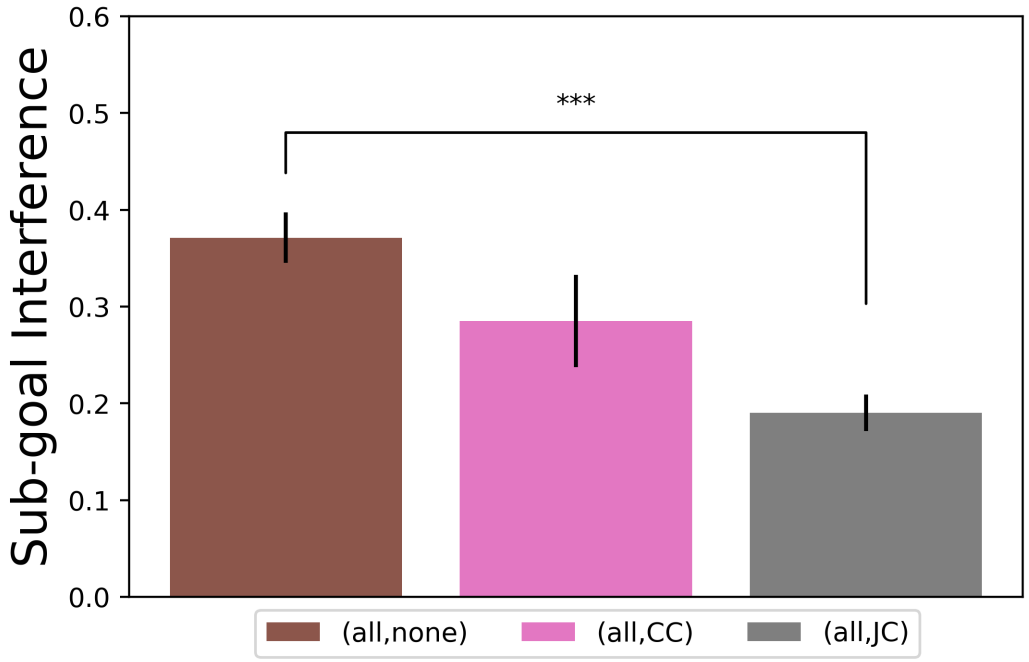



*Figure 5.9: The calculated interference values of the (all,none), (all,CC), and (all,JC) configurations. The (all,JC) configuration is significantly lower than (all,none) configuration with a p-value* $< 0.0001$*. No other comparison is significant.*

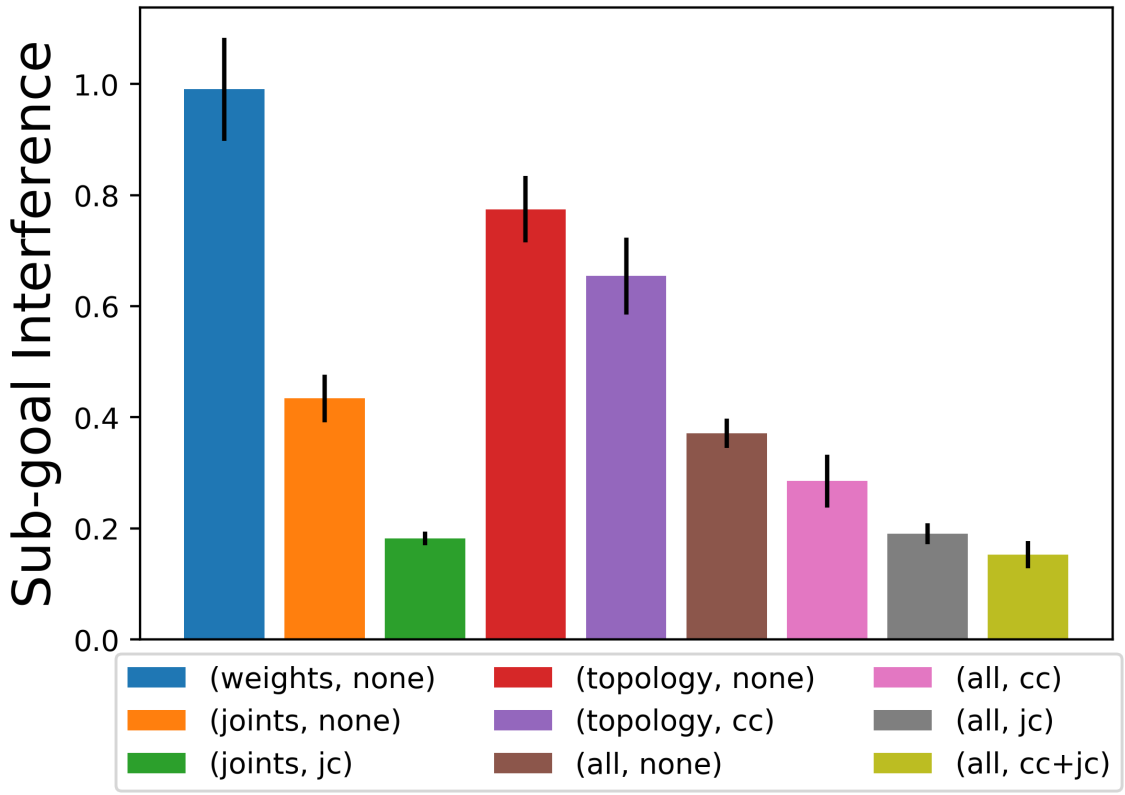



*Figure 5.10: The average calculated sub-goal interference $I$ value for each configuration when evolved in all sixteen environments. Error bars indicate $\pm$ 1 standard error.*

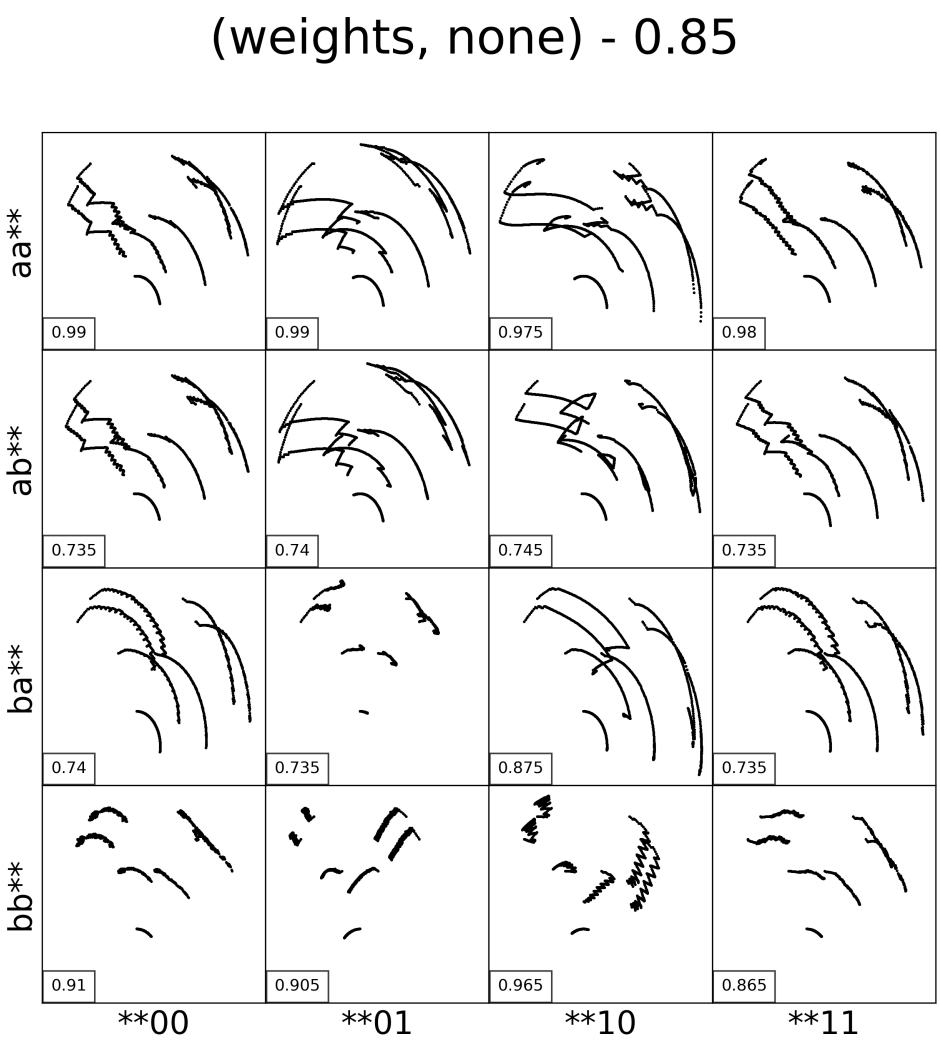



*Figure 5.11: Movement pattern of a the best robot evolved in all environments using experiment (weights, none). The environmental performance is reported in the lower left corner of each environment.*

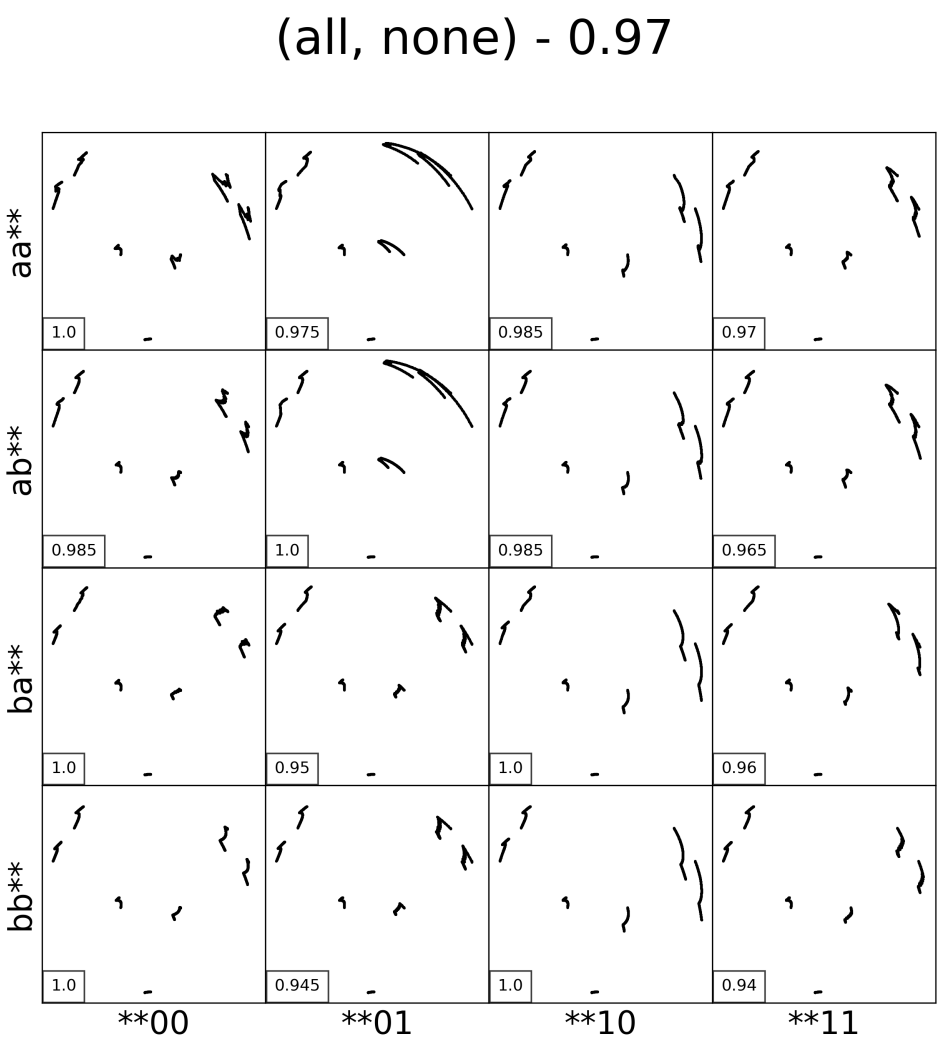



*Figure 5.12: Movement pattern of a the best robot evolved in all environments using experiment (all, none). The environmental performance is reported in the lower left corner of each environment.*

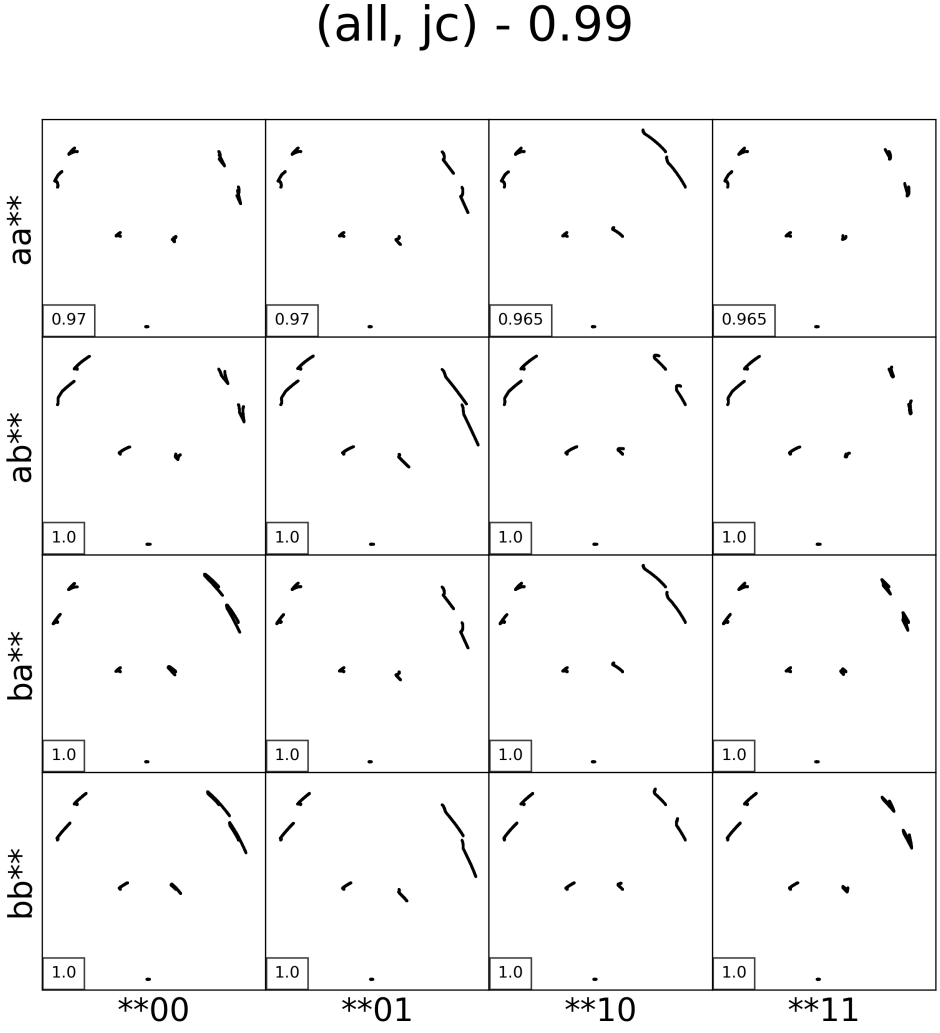



*Figure 5.13: Movement pattern of a the best robot evolved in all environments using experiment (all, jc). The environmental performance is reported in the lower left corner of each environment.*

on the results of the first experiment, only the four configurations which could mutate the morphology and topology of the network were tested: (all,none), (all,JC), (all,CC), and (all,CC+JC). Robots were evolved for 8000 generations only experiencing $\{00aa, 01ba, 10ab, 11bb\}$. This corresponds to the environments numbered $\{0, 6, 9, 15\}$ in Figure 5.2. These environments were chosen because they represent a Latin Hypercube Sample [103] the minimal set allowing the robot to experience all variations of every sub-goal. We ran thirty independent trials of each configuration. After evolution on these four environments, the best performing robot was tested in the remaining twelve environments. The average performance in these unseen environments is shown in Figure 5.14. The (all, JC) and (all, CC+JC) configurations are significantly better than the (all, none) configuration. There was no other significance between configurations.

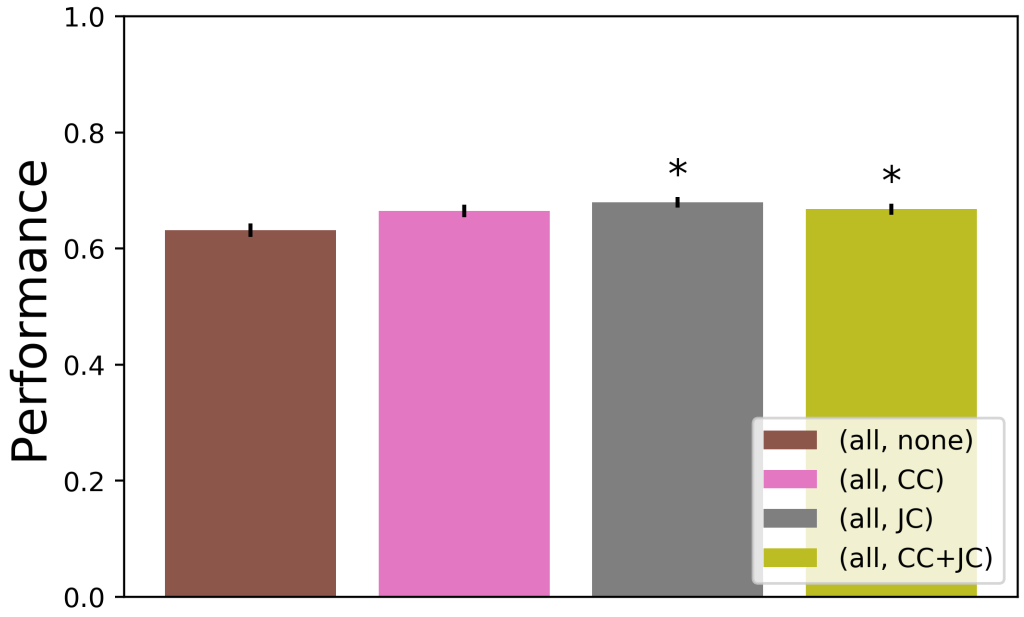



*Figure 5.14: Average performance of the best robot evolved with a training set of* $\{00aa, 01ba, 10ab, 11bb\}$ *in all unseen test environments for each cost configuration with mutation parameter set to 'all'. Error bars show* $\pm$ *1 standard error. First a Kruskal-Wallis test was performed indicating significant difference between each configurations with* $p < 0.01$*. Asterisks indicate pairwise significance as determined by the Mann-Whitney U-Test between the selected configuration and (all,none) with a p-value of* $0.01$*. There was no significance between any other pair of configurations.*

## 5.5 Discussion

The results of the first experiment, comparing all nine configurations, indicate that using joint cost produces robots which are better adapted to the task at hand compared to connection cost alternatives while still maintaining low interference behavior. The first experiment further shows that giving evolution the ability to control both morphology and controller structure leads to better overall fitness in this task: Figure 5.5 shows that average performance of a robot evolved with the (all, none) configuration is better than any other configuration when the robot is evolved in all environments.

When evaluating which configuration is *'best'* it is important to note that the objective value is only one metric to consider. Because the performance is an average over all environments, there can be cases when a robot with relatively high overall performance will sacrifice performance of a single environment to attain higher performance in the remaining environments. This is why it is important to consider the lowest performing environment as well (Fig. 5.6). This figure further shows that the (all, none) configuration will produce robots better generalized to all environments.

One can also evaluate a configuration's ability to evolve desirable robots by how likely it is that a high performing individual will be created. Figure 5.7 shows that only four configurations were able to produce robots with greater than or equal to 0.9 performance in all 16 environments: (joints, none), (all, none), (all, CC), and (all, JC). Further, only (all, JC) was able to create an individual with over 0.95% performance in every environment. This indicates that these configurations more easily find high quality solutions over alternative configurations.

One possible explanation for the relatively poor fitness performance of the configuration (all, CC+JC) compared to the non-cost and single cost options could be the structure of how cost is implemented. Because the cost is multiplied in the fitness function, it is possible that the selection pressure placed on the robots to be inexpensive is greater than the pressure to complete the objective function.

The other factor considered was how much interference there was between the sub-goals in the environment. Figure 5.10 shows the $I$ value of all configurations and gives a general trend that configurations which placed morphology under evolutionary control evolved more independent behaviors than those which did not. Further, this figure shows that adding joint cost when only the morphology can be mutated can decrease $I$ compared to the non-cost option. While the (all,CC+JC) configuration achieved the lowest interference, its low performance values indicate that the robot is simply performing the same action in multiple environments rather than evolving specialized motion of each limb in accordance with each sub-task. In contrast, (all,JC) achieved high performance and low $I$ values, indicating it was able to specialize limb movement to environmental changes.

Figures 5.11, 5.12, and 5.13 give insight into the different movement patterns produced by the different mutation and cost configurations. First consider the (weights,none) configuration. Without the ability to change morphology or neural topology, the robot attempts to solve each environment with a different movement pattern, regardless of the fact that sub-variants of the environment remain the same. This leads to behavior with a high amount of interference. In contrast, robots which can adapt their morphology and neural topology can more easily create movement patterns which are similar across multiple environments (Figs. 5.12, 5.13). This means differences

in movement across environments are the result of specialized behavior in a subset of the limbs.

These specialized behaviors can allow the robot to behave similarly when introduced to new, unseen, environments: altering behavior only in the portion of the body related to the portion of the environment that varied. Figure 5.14 shows that joint cost functions select for better performing individuals in unseen when evolution occurs on a subset of environments when compared with the non-cost configuration. This may be because limiting joint ranges is the easiest way for the robot to separate its body in accordance with the sub-goals in the environment. For example, fixing the root partitions a robot's movement into independent movement in the right and left halves which is in accordance with the modularity of the environmental sub-goals. Joint cost may be a factor in discovering modularity sooner during evolution.

## 5.6 Conclusion

In this paper we presented sub-goal interference as a novel metric for quantifying embodied modularity with respect to multiple environments. We showed that robots with evolved body plans exhibited more specialized behaviors than those with non-modular body plans. This replicates the well known biological phenomena of evolution producing bodies which are well adapted and reflect the environment in which the organism lives [31]. Our work is a first step towards the artificial evolution of embodied agents which reflect the latent structure in the environments they are to which they are exposed.

Continued focus examining the *why* of morphological design in biological organ-

isms could answer remaining questions regarding intelligence. Having an overly complex controller in a simple morphology may prohibit the automatic creation of modules and specialization in the controller due to limitations in possible movement as described by the morphology. Similarly, a simple controller may not be sufficient to adequately control a complex, highly independent morphology. The co-evolution of body and controller may be necessary to understand the evolution of intelligence in order to balance both control and morphology with respect to one another. Animals which have more freedom of movement may be required to be more intelligent in order to adequately control their actions.

Even though the task we present is a simple categorization task, it provides insight into how simple evolutionary pressures may cause more or less modularity in an organism's body plan and nervous system. This task can be made more complex by introducing tree-based body plans with more depth, new variations of sub-goals, or even completely new sub-goals themselves. In future work, we would like to explore a variety of tasks beyond basic categorization to those which are not as cleanly split into independent sub-goals. One such task is grasping. We can imagine a similar tree morphology being tasked to recognize objects in the environment, grasp objects of a certain type, and ignore the others. It would be worthwhile to see how the structure of the tree adapts in order to achieve this more complicated task.

Because most vertebrate skeletal structures can be described as a tree, we believe the use of a tree morphology enables the answering of general enough framework for which to explore modularity while still answering important questions regarding the evolution of mechanical independence, or dexterity. Our work indicates that when morphology is able to be adapted, dexterity arises in due to the independence

present in the sub-goals. Further, morphological cost can place evolutionary pressure on populations to only exhibit the most necessary components for overall success. Thus, when future new environments are encountered, it may become easier for an agent to understand environments which are simply novel combinations of previously experienced stimulus.

Morphological cost is present in many forms in biological organisms. In general, the more mass a body has, the more energy is required for movement. Similarly, there may be a structural cost associated with having more independence between joints. The joints of an animal are more prone to injury than the bones. Having a larger number of independent joints may result in a body which is more frail and subject to more severe forms of damage. In this work we present a simple morphological cost implemented by summing up the total amount of possible movement the robot could experience. A more realistic approach may be to place higher cost on the joints which move the most mass. In the case of the tree morphology presented in this work, when the root joint actuates, the position of seven total branches changes. In contrast when the joint of a leaf actuates, only one total branch position is change. The amount of energy used to actuate the root joint is therefore greater than the amount used by the leaf joint. Future work could focus on finding various other biologically plausible implementations of morphological cost.

Similar to morphological cost, the connection cost implemented here is fairly simple. A more sophisticated cost function could incorporate the embodied position of each neuron, mimicking costs associated with synaptic path length. This could lead to the hierarchical control seen in vertebrates where response to inputs from local sensation can be shortcut by being computed locally in the spine. Similarly, in the tree,

local computation can occur at the leaf level without the need to send information to upstream to the root branch. However, more complex task may require the robot to perform global computation in which many sensors will be used in computation in the root.

Lastly, this work examined the behavior of robots given incomplete combinations of the total environment space. Our results indicate that robots who experience some sort of cost over evolution are better able break down previously experienced environments into its sub-goals and reconstruct the sensation of new, unseen environments based on recognition of these previously experienced sub-goals. Taken together, this work is of import for understanding the long term scalability of evolution of embodied agents. Organisms can operate in new environments which they experience as novel combinations of familiar percepts rather than completely new sensory data. This work contributes to an understanding of such abilities evolve, and how they might be instantiated in embodied machines.

# References for chapter 5


[1] Joshua E Auerbach and Josh C Bongard. "Dynamic resolution in the co-evolution of morphology and control". In: *Artificial Life XII: Proceedings of the Twelfth International Conference on the Synthesis and Simulation of Living Systems*. CONF. MIT Press. 2010.

[2] James P Bagrow. "Communities and bottlenecks: Trees and treelike networks have high modularity". In: *Physical Review E* 85.6 (2012), p. 066118.

[4] Anton Bernatskiy and Joshua C Bongard. “Exploiting the relationship between structural modularity and sparsity for faster network evolution”. In: *Proceedings of the companion publication of the 2015 annual conference on genetic and evolutionary computation*. ACM. 2015, pp. 1173–1176.

[5] Jesper Blynel and Dario Floreano. “Exploring the T-maze: Evolving learning-like robot behaviors using CTRNNs”. In: *Workshops on Applications of Evolutionary Computation*. Springer. 2003, pp. 593–604.

[6] Jessica A Bolker. “Modularity in development and why it matters to evo-devo”. In: *American Zoologist* 40.5 (2000), pp. 770–776.

[9] Josh Bongard. “Morphological change in machines accelerates the evolution of robust behavior”. In: *Proceedings of the National Academy of Sciences* 108.4 (2011), pp. 1234–1239.

[13] Josh C Bongard et al. “Evolving robot morphology facilitates the evolution of neural modularity and evolvability”. In: *Proceedings of the 2015 annual conference on genetic and evolutionary computation*. ACM. 2015, pp. 129–136.

[16] Werner Callebaut, Diego Rasskin-Gutman, and Herbert A Simon. *Modularity: understanding the development and evolution of natural complex systems*. MIT press, 2005.

[17] Collin K Cappelle et al. “Morphological modularity can enable the evolution of robot behavior to scale linearly with the number of environmental features”. In: *Frontiers in Robotics and AI* 3 (2016), p. 59.

[18] Collin Cappelle, Anton Bernatskiy, and Josh Bongard. “Reducing Training Environments in Evolutionary Robotics Through Ecological Modularity”. In: *Conference on Biomimetic and Biohybrid Systems*. Springer. 2017, pp. 95–106.

[23] Jeff Clune, Jean-Baptiste Mouret, and Hod Lipson. “The evolutionary origins of modularity”. In: *Proc. R. Soc. B* 280.1755 (2013), p. 20122863.

[27] Jeff Clune et al. “Investigating whether HyperNEAT produces modular neural networks”. In: *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM. 2010, pp. 635–642.

[31] Charles Darwin. *On the origin of species, 1859*. Routledge, 2004.

[32] Stephane Doncieux et al. “Evolutionary robotics: what, why, and where to”. In: *Frontiers in Robotics and AI* 2 (2015), p. 4.

[37] Carlos Espinosa-Soto. “On the role of sparseness in the evolution of modularity in gene regulatory networks”. In: *PLoS computational biology* 14.5 (2018), e1006172.

[38] Carlos Espinosa-Soto and Andreas Wagner. “Specialization can drive the evolution of modularity”. In: *PLoS computational biology* 6.3 (2010), e1000719.

[42] Santo Fortunato and Marc Barthelemy. “Resolution limit in community detection”. In: *Proceedings of the National Academy of Sciences* 104.1 (2007), pp. 36–41.

[45] Tamar Friedlander et al. “Mutation rules and the evolution of sparseness and modularity in biological systems”. In: *PloS one* 8.8 (2013), e70444.

[49] Leland H Hartwell et al. “From molecular to modular cell biology”. In: *Nature* 402.6761supp (1999), p. C47.

[55] Pankaj Jalote. “ICSE 2014”. In: *Communications of the ACM* (2013).

[57] Nadav Kashtan and Uri Alon. “Spontaneous evolution of modularity and network motifs”. In: *Proceedings of the National Academy of Sciences* 102.39 (2005), pp. 13773–13778.

[58] Nadav Kashtan, Elad Noor, and Uri Alon. “Varying environments can speed up evolution”. In: *Proceedings of the National Academy of Sciences* 104.34 (2007), pp. 13711–13716.

[62] Sam Kriegman et al. “Simulating the evolution of soft and rigid-body robots”. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM. 2017, pp. 1117–1120.

[65] Lars Kunze, Tobias Roehm, and Michael Beetz. “Towards semantic robot description languages”. In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 5589–5595.

[69] Elizabeth A Leicht and Mark EJ Newman. “Community structure in directed networks”. In: *Physical review letters* 100.11 (2008), p. 118703.

[70] Hod Lipson. “Principles of modularity, regularity, and hierarchy for scalable systems”. In: *Journal of Biological Physics and Chemistry* 7.4 (2007), p. 125.

[73] Daniel Marbach and Auke Jan Ijspeert. “Co-evolution of configuration and control for homogenous modular robots”. In: *Proceedings of the eighth conference on intelligent autonomous systems (IAS8)*. CONF. IOS Press. 2004, pp. 712–719.

[74] Maja Matarić and Dave Cliff. “Challenges in evolving controllers for physical robots”. In: *Robotics and autonomous systems* 19.1 (1996), pp. 67–83.

[76] Garrett W Milliken, Jeannette P Ward, and Carl J Erickson. “Independent digit control in foraging by the aye-aye (Daubentonia madagascariensis)”. In: *Folia Primatol* 56 (1991), pp. 219–224.

[78] John Napier. “The evolution of the hand”. In: *Scientific American* 207.6 (1962), pp. 56–65.

[79] Mark EJ Newman. “Modularity and community structure in networks”. In: *Proceedings of the national academy of sciences* 103.23 (2006), pp. 8577–8582.

[84] Merav Parter, Nadav Kashtan, and Uri Alon. “Environmental variability and modularity of bacterial metabolic networks”. In: *BMC evolutionary biology* 7.1 (2007), p. 169.

[85] JEAN-JACQUES Petter. “The aye-aye”. In: *Primate conservation* (1977), pp. 37–57.

[95] Michael Schmidt and Hod Lipson. “Age-fitness pareto optimization”. In: *Genetic Programming Theory and Practice VIII*. Springer, 2011, pp. 129–146.

[98] Karl Sims. “Evolving virtual creatures”. In: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM. 1994, pp. 15–22.

[99] Russell Smith et al. “Open dynamics engine”. In: (2005).

[103] Michael Stein. “Large sample properties of simulations using Latin hypercube sampling”. In: *Technometrics* 29.2 (1987), pp. 143–151.

[104] JGM Thewissen et al. “From land to water: the origin of whales, dolphins, and porpoises”. In: *Evolution: Education and Outreach* 2.2 (2009), p. 272.

[105] Frank Veenstra et al. “Evolution and morphogenesis of simulated modular robots: a comparison between a direct and generative encoding”. In: *European Conference on the Applications of Evolutionary Computation*. Springer. 2017, pp. 870–885.

[106] Phillip Verbancsics and Kenneth O Stanley. “Constraining connectivity to encourage modularity in HyperNEAT”. In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM. 2011, pp. 1483–1490.

[107] George Von Dassow and Ed Munro. “Modularity in animal development and evolution: elements of a conceptual framework for EvoDevo”. In: *Journal of Experimental Zoology* 285.4 (1999), pp. 307–325.

[110] Günter P Wagner, Mihaela Pavlicev, and James M Cheverud. “The road to modularity”. In: *Nature Reviews Genetics* 8.12 (2007), p. 921.

[114] Nathan M Young, Gunter P Wagner, and Benedikt Hallgrimsson. “Development and the evolvability of human limbs”. In: *Proceedings of the National Academy of Sciences* 107.8 (2010), pp. 3400–3405.

# Chapter 6

# Conclusion

In this thesis, I explored the consequences of evolving robots in multiple environments. When robots cannot break down their environment into unfamiliar combinations of familiar percepts they potentially must be exponential number of environments. This leads to an intractable number of necessary simulations for the evolution of non-trivial robots capable of non-trivial behaviors.

In this thesis, I explore the impact of modularity in a robot's morphology and control in order for the robot to better reflect the structure present in the environment. When the design of the robot accurately reflects the environment, it is better able to independently act in response to locally perceived environmental features, thus allowing evolution to occur on a reduced subset of environments while exhibiting high performance in unseen environments. This indicates that modularity can be a path forward to addressing the issue of scalability in evolutionary robotics.

For any system, modularity is notoriously difficult to describe in a way that is both useful and general. In this thesis I explore modularity as it relates to the sensory-motor-environment interactions of robot instead of the common approach of using $Q$

to report the structural modularity present in an agent's neural network. This allows for a more useful definition of modularity in embodied agents as it incorporates all relevant aspects towards a robot's perception and response.

Chapters 2 and 3 consider agents in which morphology and control is hand designed to reflect the structure of the environment. These robots can be compared to agents which do not correctly reflect the environment. These experiments are necessary to show the theoretical benefits of embodied systems which already correctly reflect aspects of their environment.

Chapters 4 and 5 give evolution more control over the qualitative aspects of a robot's understanding of the environment. Whereas the experiments in Chapters 2 and 3 addressed the *why* of modularity, these experiments address the *how*. They provide insight into how modularity may be formed and how the relationship between environment and robot design may be utilized by evolution to create more modular robots.

In Chapter 4, the neural controllers are generated using HyperNEAT, a bio-inspired model for development which maintains spatial relationships between neurons. When the spatial embedding reflected the modular and hierarchical nature of the task environments presented to the robot, it was more evolvable than robots which possessed other embeddings which did not adequately capture the structure of the environment.

In Chapter 5, I show the complete evolution of a robot's morphology and neural topology. By exposing the robot to an environment which exhibits both modular and non-modular sub-goals, I show the evolution of varying degrees of dexterity which reflect the goals of the environment. Further, I show morphological cost can be used

to pressure evolution to find populations of robots which experience low interference.

## 6.1 Significance

The work presented in this thesis has implications with regards to evolution in both biology as well as an optimization tool for artificial agents.

By focusing on the fundamental process by which artificial evolution occurs and examining a theoretical agent, this thesis can be applied to give insight into biological questions regarding the evolution of modularity and the varying degree of mechanical independence shown within organisms. This work confirmed what biologists have long know: environment can guide evolution [31]. I also give support to the possibility of morphological cost applying pressure towards morphologies which accurately reflect the structure of the environment. While the conclusions of this thesis for the field of biology are more narrow, for evolutionary robotics and evolutionary computation the results are more general.

The use of a tree robot makes this work general to many other types of morphologies and to other hierarchical systems. Most robots can be described as trees in which appendages branch off from a central control body, therefore even though a robot's morphology may not look like a tree, from a theoretical aspect, it behaves similarly to the tree robot presented in this thesis. Research concerning robots which cannot be described as trees such as soft robots may still benefit from this thesis because this work generalizes to systems which must react to disparate local sensation which potentially requires immediate response through local computation or slower more collective response through the use of hierarchical control. While this thesis does

not specifically explore non-tree morphologies, the results here can be applied to any hierarchical system with distributed local sensation as described above.

This thesis provides an important novel framework for discussing the relationship between morphology, control, and environment by first providing definitions for morphological, neurological, and ecological modularity and then tying them all together with the sub-goal interference metric $I$. When $I$ is used in conjunction with fitness in a set of environments, it indicates the level of specialization of perception and response within the robot.

The use of cost has been previously implemented to constrain complexity of design [91] but, to my knowledge, it has not been used to help guide evolution towards modular solutions. Connection cost has proven effective for the formation of neural modularity [23], however I show it may be insufficient in an embodied context. This is important to help shift the focus of modularity research away from the $Q$ metric towards something more grounded in movement which is an important component of active embodied agents. Because $Q$ inherently only describes structural modularity of networks, it cannot be used to adequately describe the independence of sensations due to motion.

This thesis implies cost functions may also be part of the catalyst for the evolution of independent control and motion within biological organisms. By incorporating cost, evolutionary methods will find which components of the genome are most essential essential towards achieving high fitness. Cost can aid evolution to isolate the independent components of the agent to accurately reflect the independent aspects of the environment.

Outside of using cost functions, this thesis also explored the concept of embodied

embeddings which can be used to describe better neural embeddings for the popular evolutionary algorithm HyperNEAT. These embeddings create relationships within the agents neural controller and have previously been based on grid patterns or chosen arbitrarily by the researcher [26]. This thesis extends work which uses the body as the starting point for the location of neurons in the embedding [25] by using embodied concepts such as hierarchy and locality when determining the embedding. This work can be used to help future researchers choose how to structure the spatial relationship between neurons given a morphology and set of task environments.

To summarize, this thesis first defines what modularity is in an embodied context with relation to environment, second shows why modularity is important to the long term scalability of evolutionary robotics, and third examines a variety of ways in which the structure and modularity of the environment can become reflected in an evolved agent.

## 6.2 Future Work

Many questions remain due to the theoretical nature of this thesis. This section describes the various avenues future work should explore in order to contribute to this area of research.

Future implementations using morphological cost should strive to use more general multi-objective methods to receive the full benefit of what morphological cost can provide. In this thesis I deliberately used naïve methods such as simply combining the cost penalty term in the fitness function in order to compare different implementations of cost. However, now that morphological cost has been shown to be effective even

in this simple use case, it would be fruitful to utilize the complete potential given by more explicit multi-objective techniques of having a spectrum of individuals in the population exhibiting different cost values.

Along with a different multi-objective implementation, the use of different cost functions could be an area of worthwhile research. Embodied agents provide a greater array of feasible constraints than their disembodied counterparts. One biologically inspired cost function is the use of energy. Here morphological cost was used as a simple penalty to the maximum range of joints in the phenotype but, instead, penalizing for energy usage could accomplish the same goal while leading towards more conservative behavior. Energy is also an attractive cost option because of its biological reality.

However energy may be too indirect to accomplish the intended goal and specific robots may have obvious forms of cost which may allow faster evolution. Some examples of potentially useful morphological cost functions can be:

- *Mass and volume* - Selecting for smaller robots may increase the specialization by reducing the amount of material connecting different areas of the robot. This would be useful for robots in which the entire body plan is placed under evolutionary control.

- *Number of joints* - By placing a cost on how many joints the robot has, pressure is directly placed on the mechanical independence. In this case, only the joints which are found to be most necessary to complete the task will remain. This type of cost could be useful when the pieces of the body are all pre-defined but not how they are connected.

- *Joint torque and speed* - This type of cost is similar to the one presented in this thesis. By inhibiting how much a joint can lift or how fast it can move, different forms of relationships between the bodies may arise. It might be found that joints which are higher up on the hierarchy (and therefore control more of the body) may have a higher torque so as to move the entirety of the body but may require lower speed so as to not disrupt the local computation of sensors lower down in the hierarchy. This type of cost would probably be most beneficial in a similar case to the tree robot where evolution controls how the joints move.

- *Material type* - In reality robots are not made up of only one material. By implementing cost to favor cheaper types of materials to create the morphology, there may be impacts to the mechanical relationships within the body of the robot. This type of cost will most likely be beneficial to soft robots where the material varies widely and has a larger effect on the overall movement and control of the robot.

Finally, future work should try to incorporate performance and sub-goal interference more directly. In this work we introduce $I$ as a separate metric from fitness, but for actual analysis it seems useful to evaluate both performance and $I$ to completely describe, in one metric, the quality of the robot. This thesis shows that it may be difficult to do in general or for larger systems because it requires a complete knowledge of the association between the portions of the robot and the independent sub-goals. However, it seems fruitful to continue exploring a general relationship between robot and environment in order for evolutionary robotics to continue progressing.

Aside from additional implementation techniques, this thesis raises questions regarding the fundamental nature of modularity. Can modularity form when there is no

perceivable presence of modularity? Most experiments which approach modularity do so by exposing the agent to stimulus which contains some level of modularity already present. What happens when the human researcher does not perceive modularity in the task they assign to the agent? Consider the evolution of the human hand. Although many other species exist in the same or similar environments, the human hand is unique because it exhibits a high level of mechanical independence between digits. What in our environment led to the formation of the hand? Are there better, or more modular, versions of the hand that we as humans do not recognize? Can artificial evolution generate agents with forms of modularity not recognizable to the human observer? This may be important to the scalability of evolutionary approaches when it is not known how many free parameters are present in the environment and how the parameters of the environment relate to each other. It could be that by evolving robots which minimize various forms of cost discover novel modularity in the environment.

# Bibliography


[1] Joshua E Auerbach and Josh C Bongard. "Dynamic resolution in the co-evolution of morphology and control". In: *Artificial Life XII: Proceedings of the Twelfth International Conference on the Synthesis and Simulation of Living Systems*. CONF. MIT Press. 2010.

[2] James P Bagrow. "Communities and bottlenecks: Trees and treelike networks have high modularity". In: *Physical Review E* 85.6 (2012), p. 066118.

[3] Wolfgang Banzhaf et al. *Genetic programming: an introduction*. Vol. 1. Morgan Kaufmann San Francisco, 1998.

[4] Anton Bernatskiy and Joshua C Bongard. "Exploiting the relationship between structural modularity and sparsity for faster network evolution". In: *Proceedings of the companion publication of the 2015 annual conference on genetic and evolutionary computation*. ACM. 2015, pp. 1173–1176.

[5] Jesper Blynel and Dario Floreano. "Exploring the T-maze: Evolving learning-like robot behaviors using CTRNNs". In: *Workshops on Applications of Evolutionary Computation*. Springer. 2003, pp. 593–604.

[6] Jessica A Bolker. "Modularity in development and why it matters to evo-devo". In: *American Zoologist* 40.5 (2000), pp. 770–776.

[7] Josh Bongard. "Evolving modular genetic regulatory networks". In: *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*. Vol. 2. IEEE. 2002, pp. 1872–1877.

[8] Josh Bongard. "Morphological change in machines accelerates the evolution of robust behavior". In: *Proceedings of the National Academy of Sciences* 108.4 (2011), pp. 1234–1239.

[9] Josh Bongard. "Morphological change in machines accelerates the evolution of robust behavior". In: *Proceedings of the National Academy of Sciences* 108.4 (2011), pp. 1234–1239.

[10] Josh C Bongard. "Spontaneous evolution of structural modularity in robot neural network controllers". In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM. 2011, pp. 251–258.

[11] Josh C Bongard. "Spontaneous evolution of structural modularity in robot neural network controllers". In: *Proceedings of the 2011 Genetic and Evolutionary Computation Conference*. Dublin: ACM, 2011, pp. 251–258.

[12] Josh C Bongard and Rolf Pfeifer. "Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny". In: *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc. 2001, pp. 829–836.

[13] Josh C Bongard et al. "Evolving robot morphology facilitates the evolution of neural modularity and evolvability". In: *Proceedings of the 2015 annual conference on genetic and evolutionary computation*. ACM. 2015, pp. 129–136.

[14] Rodney A Brooks. "Elephants don't play chess". In: *Robotics and autonomous systems* 6.1 (1990), pp. 3–15.

[15] Duncan S Callaway et al. "Network robustness and fragility: Percolation on random graphs". In: *Physical review letters* 85.25 (2000), p. 5468.

[16] Werner Callebaut, Diego Rasskin-Gutman, and Herbert A Simon. *Modularity: understanding the development and evolution of natural complex systems*. MIT press, 2005.

[17] Collin K Cappelle et al. "Morphological modularity can enable the evolution of robot behavior to scale linearly with the number of environmental features". In: *Frontiers in Robotics and AI* 3 (2016), p. 59.

[18] Collin Cappelle, Anton Bernatskiy, and Josh Bongard. "Reducing Training Environments in Evolutionary Robotics Through Ecological Modularity". In: *Conference on Biomimetic and Biohybrid Systems*. Springer. 2017, pp. 95–106.

[19] Sean B Carroll. "Chance and necessity: the evolution of morphological complexity and diversity". In: *Nature* 409.6823 (2001), p. 1102.

[20] Nick Cheney et al. "Scalable co-optimization of morphology and control in embodied machines". In: *Journal of The Royal Society Interface* 15.143 (2018), p. 20170937.

[21] Andy Clark. *Being there: Putting brain, body, and world together again*. MIT press, 1998.

[22] Dave Cliff, Phil Husbands, and Inman Harvey. "Explorations in evolutionary robotics". In: *Adaptive behavior* 2.1 (1993), pp. 73–110.

[23] Jeff Clune, Jean-Baptiste Mouret, and Hod Lipson. “The evolutionary origins of modularity”. In: *Proc. R. Soc. B* 280.1755 (2013), p. 20122863.

[24] Jeff Clune, Jean-Baptiste Mouret, and Hod Lipson. “The evolutionary origins of modularity”. In: *Procs of the Royal Society B: Biological sciences* 280.1755 (2013), p. 20122863.

[25] Jeff Clune, Charles Ofria, and Robert T Pennock. “The sensitivity of HyperNEAT to different geometric representations of a problem”. In: *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. ACM. 2009, pp. 675–682.

[26] Jeff Clune et al. “Evolving coordinated quadruped gaits with the HyperNEAT generative encoding”. In: *2009 iEEE congress on evolutionary computation*. IEEE. 2009, pp. 2764–2771.

[27] Jeff Clune et al. “Investigating whether HyperNEAT produces modular neural networks”. In: *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM. 2010, pp. 635–642.

[28] Jeff Clune et al. “On the performance of indirect encoding across the continuum of regularity”. In: *IEEE Transactions on Evolutionary Computation* 15.3 (2011), pp. 346–367.

[29] Paolo Crucitti, Vito Latora, and Massimo Marchiori. “A topological analysis of the Italian electric power grid”. In: *Physica A: Statistical mechanics and its applications* 338.1-2 (2004), pp. 92–97.

[30] David B D’Ambrosio, Jason Gauci, and Kenneth O Stanley. “HyperNEAT: The first five years”. In: *Growing adaptive machines*. Springer, 2014, pp. 159–185.

[31] Charles Darwin. *On the origin of species, 1859*. Routledge, 2004.

[32] Stephane Doncieux et al. “Evolutionary robotics: what, why, and where to”. In: *Frontiers in Robotics and AI* 2 (2015), p. 4.

[33] Denis Duboule. “Vertebrate hox gene regulation: clustering and/or colinearity?” In: *Current opinion in genetics & development* 8.5 (1998), pp. 514–518.

[34] Agoston E Eiben, James E Smith, et al. *Introduction to evolutionary computing*. Vol. 53. Springer, 2003.

[35] Kai Olav Ellefsen, Jean-Baptiste Mouret, and Jeff Clune. “Neural modularity helps organisms evolve to learn new skills without forgetting old skills”. In: *PLoS Comput Biol* 11.4 (2015), e1004128.

[36] C. Espinosa-Soto and A. Wagner. “Specialization can drive the evolution of modularity”. In: *PLoS Comp Biol* 6.3 (2010), e 1000719.

[37] Carlos Espinosa-Soto. “On the role of sparseness in the evolution of modularity in gene regulatory networks”. In: *PLoS computational biology* 14.5 (2018), e1006172.

[38] Carlos Espinosa-Soto and Andreas Wagner. “Specialization can drive the evolution of modularity”. In: *PLoS computational biology* 6.3 (2010), e1000719.

[39] Luca Ferrarini et al. “Hierarchical functional modularity in the resting-state human brain”. In: *Human brain mapping* 30.7 (2009), pp. 2220–2231.

[40] Robert Fitch et al. “Reconfigurable modular robotics”. In: *Robotics and Autonomous Systems* 7.62 (2014), pp. 943–944.

[41] Dario Floreano and Francesco Mondada. “Evolution of homing navigation in a real mobile robot”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.3 (1996), pp. 396–407.

[42] Santo Fortunato and Marc Barthelemy. “Resolution limit in community detection”. In: *Proceedings of the National Academy of Sciences* 104.1 (2007), pp. 36–41.

[43] James A Foster. “Computational genetics: Evolutionary computation”. In: *Nature Reviews Genetics* 2.6 (2001), p. 428.

[44] Robert M French. “Catastrophic forgetting in connectionist networks”. In: *Trends in cognitive sciences* 3.4 (1999), pp. 128–135.

[45] Tamar Friedlander et al. “Mutation rules and the evolution of sparseness and modularity in biological systems”. In: *PloS one* 8.8 (2013), e70444.

[46] Nicolás García-Pedrajas, Domingo Ortiz-Boyer, and César Hervás-Martínez. “An alternative approach for neural network evolution with a genetic algorithm: Crossover by combinatorial optimization”. In: *Neural Networks* 19.4 (2006), pp. 514–528.

[47] Jason Gauci and Kenneth O Stanley. “Indirect encoding of neural networks for scalable go”. In: *International Conference on Parallel Problem Solving from Nature*. Springer. 2010, pp. 354–363.

[48] F. Gruau. “Automatic definition of modular neural networks”. In: *Adaptive Behaviour* 3 (1994), pp. 151–183.

[49] Leland H Hartwell et al. “From molecular to modular cell biology”. In: *Nature* 402.6761supp (1999), p. C47.

[50] Inman Harvey et al. “Evolutionary robotics: the Sussex approach”. In: *Robotics and autonomous systems* 20.2-4 (1997), pp. 205–224.

[51] Arend Hintze and Christoph Adami. “Evolution of complex modular biological networks”. In: *PLoS computational biology* 4.2 (2008), e23.

[52] Gregory Hornby et al. “Automated antenna design with evolutionary algorithms”. In: *Space 2006*. 2006, p. 7242.

[53] Joost Huizinga, Jeff Clune, and Jean-Baptiste Mouret. “Evolving neural networks that are both modular and regular: HyperNEAT plus the connection cost technique”. In: *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. ACM. 2014, pp. 697–704.

[54] Nick Jakobi, Phil Husbands, and Inman Harvey. “Noise and the reality gap: The use of simulation in evolutionary robotics”. In: *Advances in artificial life*. Springer, 1995, pp. 704–720. DOI: 10.1007/3-540-59496-5_337.

[55] Pankaj Jalote. “ICSE 2014”. In: *Communications of the ACM* (2013).

[56] N. Kashtan and U. Alon. “Spontaneous evolution of modularity and network motifs”. In: *PNAS* 102.39 (2005), p. 13773.

[57] Nadav Kashtan and Uri Alon. “Spontaneous evolution of modularity and network motifs”. In: *Proceedings of the National Academy of Sciences* 102.39 (2005), pp. 13773–13778.

[58] Nadav Kashtan, Elad Noor, and Uri Alon. “Varying environments can speed up evolution”. In: *Proceedings of the National Academy of Sciences* 104.34 (2007), pp. 13711–13716.

[59] Sylvain Koos, Jean-Baptiste Mouret, and Stéphane Doncieux. “The transferability approach: Crossing the reality gap in evolutionary robotics”. In: *IEEE Transactions on Evolutionary Computation* 17.1 (2012), pp. 122–145.

[60] Kostas Kouvaris et al. “How Evolution Learns to Generalise: Principles of under-fitting, over-fitting and induction in the evolution of developmental organisation”. In: *arXiv preprint arXiv:1508.06854* (2015).

[61] Anat Kreimer et al. “The evolution of modularity in bacterial metabolic networks”. In: *Proceedings of the National Academy of Sciences* 105.19 (2008), pp. 6976–6981.

[62] Sam Kriegman et al. “Simulating the evolution of soft and rigid-body robots”. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM. 2017, pp. 1117–1120.

[63] Robb Krumlauf. “Hox genes in vertebrate development”. In: *Cell* 78.2 (1994), pp. 191–201.

[64] Manoj Kumar et al. “Genetic algorithm: Review and application”. In: *International Journal of Information Technology and Knowledge Management* 2.2 (2010), pp. 451–454.

[65] Lars Kunze, Tobias Roehm, and Michael Beetz. "Towards semantic robot description languages". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 5589–5595.

[66] Quoc V Le et al. "Building high-level features using large scale unsupervised learning". In: *arXiv preprint arXiv:1112.6209* (2011).

[67] Joel Lehman and Kenneth O Stanley. "Exploiting open-endedness to solve problems through the search for novelty." In: *ALIFE*. 2008, pp. 329–336.

[68] Joel Lehman et al. "Encouraging Reactivity to Create Robust Machines". In: *Adaptive Behavior* (2013).

[69] Elizabeth A Leicht and Mark EJ Newman. "Community structure in directed networks". In: *Physical review letters* 100.11 (2008), p. 118703.

[70] Hod Lipson. "Principles of modularity, regularity, and hierarchy for scalable systems". In: *Journal of Biological Physics and Chemistry* 7.4 (2007), p. 125.

[71] Hod Lipson and Jordan B Pollack. "Automatic design and manufacture of robotic lifeforms". In: *Nature* 406.6799 (2000), p. 974.

[72] Hod Lipson et al. "On the origin of modular variation". In: *Evolution* 56.8 (2002), pp. 1549–1556.

[73] Daniel Marbach and Auke Jan Ijspeert. "Co-evolution of configuration and control for homogenous modular robots". In: *Proceedings of the eighth conference on intelligent autonomous systems (IAS8)*. CONF. IOS Press. 2004, pp. 712–719.

[74] Maja Matarić and Dave Cliff. "Challenges in evolving controllers for physical robots". In: *Robotics and autonomous systems* 19.1 (1996), pp. 67–83.

[75] David Meunier et al. "Hierarchical modularity in human brain functional networks". In: *Frontiers in neuroinformatics* 3 (2009), p. 37.

[76] Garrett W Milliken, Jeannette P Ward, and Carl J Erickson. "Independent digit control in foraging by the aye-aye (Daubentonia madagascariensis)". In: *Folia Primatol* 56 (1991), pp. 219–224.

[77] Francesco Mondada, Edoardo Franzi, and Andre Guignard. "The development of khepera". In: *Experiments with the Mini-Robot Khepera, Proceedings of the First International Khepera Workshop*. CONF. 1999, pp. 7–14.

[78] John Napier. "The evolution of the hand". In: *Scientific American* 207.6 (1962), pp. 56–65.

[79] Mark EJ Newman. "Modularity and community structure in networks". In: *Proceedings of the national academy of sciences* 103.23 (2006), pp. 8577–8582.

[80] Mark EJ Newman and Michelle Girvan. "Finding and evaluating community structure in networks". In: *Physical review E* 69.2 (2004), p. 026113.

[81] Stefano Nolfi and Dario Floreano. "Evolutionary Robotics: The Biology". In: *Intelligence, and Technology of Self-Organizing Machines, Bradford Company, Scituate, MA* (2004).

[82] Stefano Nolfi, Dario Floreano, and Director Dario Floreano. *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT press, 2000.

[83] Lawrence Page et al. *The PageRank citation ranking: Bringing order to the web.* Tech. rep. Stanford InfoLab, 1999.

[84] Merav Parter, Nadav Kashtan, and Uri Alon. "Environmental variability and modularity of bacterial metabolic networks". In: *BMC evolutionary biology* 7.1 (2007), p. 169.

[85] JEAN-JACQUES Petter. "The aye-aye". In: *Primate conservation* (1977), pp. 37–57.

[86] Rolf Pfeifer and Josh Bongard. *How the body shapes the way we think: a new view of intelligence*. MIT press, 2006.

[87] Massimo Pigliucci. "Is evolvability evolvable?" In: *Nature Reviews Genetics* 9.1 (2008), p. 75.

[88] Tony Pinville et al. "How to promote generalisation in evolutionary robotics: The ProGAb approach". In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM. 2011, pp. 259–266.

[89] Nataša Pržulj. "Biological network comparison using graphlet degree distribution". In: *Bioinformatics* 23.2 (2007), e177–e183.

[90] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. "Quality diversity: A new frontier for evolutionary computation". In: *Frontiers in Robotics and AI* 3 (2016), p. 40.

[91] Timothy E Revello and Robert McCartney. "A cost term in an evolutionary robotics fitness function". In: *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512)*. Vol. 1. IEEE. 2000, pp. 125–132.

[92] Sebastian Risi, Joel Lehman, and Kenneth O Stanley. "Evolving the placement and density of neurons in the hyperneat substrate". In: *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM. 2010, pp. 563–570.

[93] Sebastian Risi and Kenneth O Stanley. "Enhancing es-hyperneat to evolve more complex regular neural networks". In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM. 2011, pp. 1539–1546.

[94] Gerhard Schlosser and Günter P Wagner. *Modularity in development and evolution*. University of Chicago Press, 2004.

[95] Michael Schmidt and Hod Lipson. "Age-fitness pareto optimization". In: *Genetic Programming Theory and Practice VIII*. Springer, 2011, pp. 129–146.

[96] John Scott and Peter J Carrington. *The SAGE handbook of social network analysis*. SAGE publications, 2011.

[97] Jimmy Secretan et al. "Picbreeder: evolving pictures collaboratively online". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2008, pp. 1759–1768.

[98] Karl Sims. "Evolving virtual creatures". In: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM. 1994, pp. 15–22.

[99] Russell Smith et al. "Open dynamics engine". In: (2005).

[100] Ricard V Solé and Sergi Valverde. "Spontaneous emergence of modularity in cellular networks". In: *Journal of The Royal Society Interface* 5.18 (2007), pp. 129–133.

[101] Kenneth O Stanley, David B D'Ambrosio, and Jason Gauci. "A hypercube-based encoding for evolving large-scale neural networks". In: *Artificial life* 15.2 (2009), pp. 185–212.

[102] Kenneth O Stanley and Risto Miikkulainen. "Evolving neural networks through augmenting topologies". In: *Evolutionary computation* 10.2 (2002), pp. 99–127.

[103] Michael Stein. "Large sample properties of simulations using Latin hypercube sampling". In: *Technometrics* 29.2 (1987), pp. 143–151.

[104] JGM Thewissen et al. "From land to water: the origin of whales, dolphins, and porpoises". In: *Evolution: Education and Outreach* 2.2 (2009), p. 272.

[105] Frank Veenstra et al. "Evolution and morphogenesis of simulated modular robots: a comparison between a direct and generative encoding". In: *European Conference on the Applications of Evolutionary Computation*. Springer. 2017, pp. 870–885.

[106] Phillip Verbancsics and Kenneth O Stanley. "Constraining connectivity to encourage modularity in HyperNEAT". In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM. 2011, pp. 1483–1490.

[107] George Von Dassow and Ed Munro. "Modularity in animal development and evolution: elements of a conceptual framework for EvoDevo". In: *Journal of Experimental Zoology* 285.4 (1999), pp. 307–325.

[108] G.P. Wagner, M. Pavlicev, and J.M. Cheverud. "The road to modularity". In: *Nature Reviews Genetics* 8.12 (2007), pp. 921–931. ISSN: 1471-0056.

[109] Günter P Wagner. "Homologues, natural kinds and the evolution of modularity". In: *American Zoologist* 36.1 (1996), pp. 36–43.

[110] Günter P Wagner, Mihaela Pavlicev, and James M Cheverud. "The road to modularity". In: *Nature Reviews Genetics* 8.12 (2007), p. 921.

[111] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*. Vol. 8. Cambridge university press, 1994.

[112] Yuichi Yamashita and Jun Tani. "Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment". In: *PLoS Comp Bio* 4.11 (2008), e1000220.

[113] Mark Yim et al. "Modular self-reconfigurable robot systems [grand challenges of robotics]". In: *Robotics & Automation Magazine, IEEE* 14.1 (2007), pp. 43–52.

[114] Nathan M Young, Gunter P Wagner, and Benedikt Hallgrimsson. "Development and the evolvability of human limbs". In: *Proceedings of the National Academy of Sciences* 107.8 (2010), pp. 3400–3405.